14 NLP Research Breakthroughs You Can Apply To Your Business

These papers provide a breadth of information about NLP [**Natural language processing**] that is generally useful and interesting from a data science perspective.

Contents

12. A Hierarchical Multi-task Approach for Learning Embeddings from Semantic Tasks

13. Sequence classification with human attention

14. Improving Language Understanding by Generative Pre-Training

Universal Language Model Fine-tuning for Text Classification

Jeremy Howard*
fast.ai
University of San Francisco
j@fast.ai

Sebastian Ruder*
Insight Centre, NUI Galway
Aylien Ltd., Dublin
sebastian@ruder.io

Abstract

Inductive transfer learning has greatly impacted computer vision, but existing approaches in NLP still require task-specific modifications and training from scratch. We propose Universal Language Model Fine-tuning (ULMFiT), an effective transfer learning method that can be applied to any task in NLP, and introduce techniques that are key for fine-tuning a language model. Our method significantly outperforms the state-of-the-art on six text classification tasks, reducing the error by 18-24% on the majority of datasets. Furthermore, with only 100 labeled examples, it matches the performance of training from scratch on 100× more data. We open-source our pretrained models and code[1].

1 Introduction

Inductive transfer learning has had a large impact on computer vision (CV). Applied CV models (including object detection, classification, and segmentation) are rarely trained from scratch, but instead are fine-tuned from models that have been pretrained on ImageNet, MS-COCO, and other datasets (Sharif Razavian et al., 2014; Long et al., 2015a; He et al., 2016; Huang et al., 2017).

Text classification is a category of Natural Language Processing (NLP) tasks with real-world applications such as spam, fraud, and bot detection (Jindal and Liu, 2007; Ngai et al., 2011; Chu et al., 2012), emergency response (Caragea et al., 2011), and commercial document classification, such as for legal discovery (Roitblat et al., 2010).

While Deep Learning models have achieved state-of-the-art on many NLP tasks, these models are trained from scratch, requiring large datasets, and days to converge. Research in NLP focused mostly on *transductive* transfer (Blitzer et al., 2007). For *inductive* transfer, fine-tuning pretrained word embeddings (Mikolov et al., 2013), a simple transfer technique that only targets a model's first layer, has had a large impact in practice and is used in most state-of-the-art models. Recent approaches that concatenate embeddings derived from other tasks with the input at different layers (Peters et al., 2017; McCann et al., 2017; Peters et al., 2018) still train the main task model from scratch and treat pretrained embeddings as fixed parameters, limiting their usefulness.

In light of the benefits of pretraining (Erhan et al., 2010), we should be able to do better than *randomly initializing* the remaining parameters of our models. However, inductive transfer via fine-tuning has been unsuccessful for NLP (Mou et al., 2016). Dai and Le (2015) first proposed fine-tuning a language model (LM) but require millions of in-domain documents to achieve good performance, which severely limits its applicability.

We show that not the idea of LM fine-tuning but our lack of knowledge of how to train them effectively has been hindering wider adoption. LMs overfit to small datasets and suffered catastrophic forgetting when fine-tuned with a classifier. Compared to CV, NLP models are typically more shallow and thus require different fine-tuning methods.

We propose a new method, Universal Language Model Fine-tuning (ULMFiT) that addresses these issues and enables robust inductive transfer learning for any NLP task, akin to fine-tuning ImageNet models: The same 3-layer LSTM architecture—with the same hyperparameters and no additions other than tuned dropout hyperparameters—outperforms highly engineered models and trans-

[1] http://nlp.fast.ai/ulmfit.

*Equal contribution. Jeremy focused on the algorithm development and implementation, Sebastian focused on the experiments and writing.

fer learning approaches on six widely studied text classification tasks. On IMDb, with 100 labeled examples, ULMFiT matches the performance of training from scratch with $10\times$ and—given 50k unlabeled examples—with $100\times$ more data.

Contributions Our contributions are the following: 1) We propose Universal Language Model Fine-tuning (ULMFiT), a method that can be used to achieve CV-like transfer learning for any task for NLP. 2) We propose *discriminative fine-tuning*, *slanted triangular learning rates*, and *gradual unfreezing*, novel techniques to retain previous knowledge and avoid catastrophic forgetting during fine-tuning. 3) We significantly outperform the state-of-the-art on six representative text classification datasets, with an error reduction of 18-24% on the majority of datasets. 4) We show that our method enables extremely sample-efficient transfer learning and perform an extensive ablation analysis. 5) We make the pretrained models and our code available to enable wider adoption.

2 Related work

Transfer learning in CV Features in deep neural networks in CV have been observed to transition from *general* to task-*specific* from the first to the last layer (Yosinski et al., 2014). For this reason, most work in CV focuses on transferring the first layers of the model (Long et al., 2015b). Sharif Razavian et al. (2014) achieve state-of-the-art results using features of an ImageNet model as input to a simple classifier. In recent years, this approach has been superseded by fine-tuning either the last (Donahue et al., 2014) or several of the last layers of a pretrained model and leaving the remaining layers frozen (Long et al., 2015a).

Hypercolumns In NLP, only recently have methods been proposed that go beyond transferring word embeddings. The prevailing approach is to pretrain embeddings that capture additional context via other tasks. Embeddings at different levels are then used as features, concatenated either with the word embeddings or with the inputs at intermediate layers. This method is known as hypercolumns (Hariharan et al., 2015) in CV[2] and is used by Peters et al. (2017), Peters et al. (2018), Wieting and Gimpel (2017), Conneau et al. (2017), and McCann et al. (2017) who use language modeling, paraphrasing, entailment, and Machine Translation (MT) respectively for pretraining. Specifically, Peters et al. (2018) require engineered custom architectures, while we show state-of-the-art performance with the same basic architecture across a range of tasks. In CV, hypercolumns have been nearly entirely superseded by end-to-end fine-tuning (Long et al., 2015a).

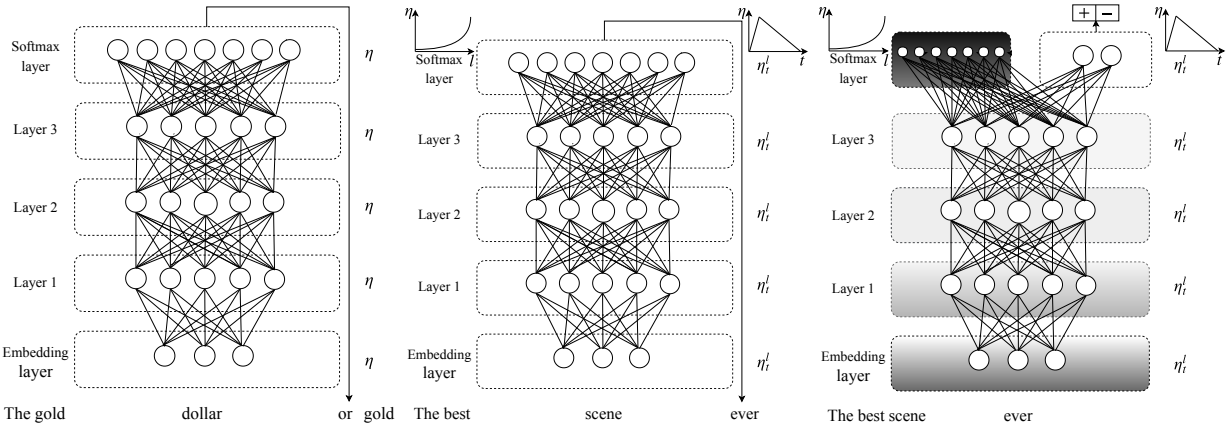
Multi-task learning A related direction is multi-task learning (MTL) (Caruana, 1993). This is the approach taken by Rei (2017) and Liu et al. (2018) who add a language modeling objective to the model that is trained jointly with the main task model. MTL requires the tasks to be trained from scratch every time, which makes it inefficient and often requires careful weighting of the task-specific objective functions (Chen et al., 2017).

Fine-tuning Fine-tuning has been used successfully to transfer between similar tasks, e.g. in QA (Min et al., 2017), for distantly supervised sentiment analysis (Severyn and Moschitti, 2015), or MT domains (Sennrich et al., 2015) but has been shown to fail between unrelated ones (Mou et al., 2016). Dai and Le (2015) also fine-tune a language model, but overfit with 10k labeled examples and require millions of in-domain documents for good performance. In contrast, ULMFiT leverages general-domain pretraining and novel fine-tuning techniques to prevent overfitting even with only 100 labeled examples and achieves state-of-the-art results also on small datasets.

3 Universal Language Model Fine-tuning

We are interested in the most general *inductive* transfer learning setting for NLP (Pan and Yang, 2010): Given a static source task $\mathcal{T}_S$ and *any* target task $\mathcal{T}_T$ with $\mathcal{T}_S \neq \mathcal{T}_T$, we would like to improve performance on $\mathcal{T}_T$. Language modeling can be seen as the ideal source task and a counterpart of ImageNet for NLP: It captures many facets of language relevant for downstream tasks, such as long-term dependencies (Linzen et al., 2016), hierarchical relations (Gulordava et al., 2018), and sentiment (Radford et al., 2017). In contrast to tasks like MT (McCann et al., 2017) and entailment (Conneau et al., 2017), it provides data in near-unlimited quantities for most domains and languages. Additionally, a pretrained LM can be easily adapted to the idiosyncrasies of a target

[2] A hypercolumn at a pixel in CV is the vector of activations of all CNN units above that pixel. In analogy, a hypercolumn for a word or sentence in NLP is the concatenation of embeddings at different layers in a pretrained model.



(a) LM pre-training (b) LM fine-tuning (c) Classifier fine-tuning

Figure 1: ULMFiT consists of three stages: a) The LM is trained on a general-domain corpus to capture general features of the language in different layers. b) The full LM is fine-tuned on target task data using discriminative fine-tuning ('*Discr*') and slanted triangular learning rates (STLR) to learn task-specific features. c) The classifier is fine-tuned on the target task using gradual unfreezing, '*Discr*', and STLR to preserve low-level representations and adapt high-level ones (shaded: unfreezing stages; black: frozen).

task, which we show significantly improves performance (see Section 5). Moreover, language modeling already is a key component of existing tasks such as MT and dialogue modeling. Formally, language modeling induces a hypothesis space $\mathcal{H}$ that should be useful for many other NLP tasks (Vapnik and Kotz, 1982; Baxter, 2000).

We propose Universal Language Model Fine-tuning (ULMFiT), which pretrains a language model (LM) on a large general-domain corpus and fine-tunes it on the target task using novel techniques. The method is *universal* in the sense that it meets these practical criteria: 1) It works across tasks varying in document size, number, and label type; 2) it uses a single architecture and training process; 3) it requires no custom feature engineering or preprocessing; and 4) it does not require additional in-domain documents or labels.

In our experiments, we use the state-of-the-art language model AWD-LSTM (Merity et al., 2017a), a regular LSTM (with no attention, short-cut connections, or other sophisticated additions) with various tuned dropout hyperparameters. Analogous to CV, we expect that downstream performance can be improved by using higher-performance language models in the future.

ULMFiT consists of the following steps, which we show in Figure 1: a) General-domain LM pretraining (§3.1); b) target task LM fine-tuning (§3.2); and c) target task classifier fine-tuning (§3.3). We discuss these in the following sections.

3.1 General-domain LM pretraining

An ImageNet-like corpus for language should be large and capture general properties of language. We pretrain the language model on Wikitext-103 (Merity et al., 2017b) consisting of 28,595 preprocessed Wikipedia articles and 103 million words. Pretraining is most beneficial for tasks with small datasets and enables generalization even with 100 labeled examples. We leave the exploration of more diverse pretraining corpora to future work, but expect that they would boost performance. While this stage is the most expensive, it only needs to be performed once and improves performance and convergence of downstream models.

3.2 Target task LM fine-tuning

No matter how diverse the general-domain data used for pretraining is, the data of the target task will likely come from a different distribution. We thus fine-tune the LM on data of the target task. Given a pretrained general-domain LM, this stage converges faster as it only needs to adapt to the idiosyncrasies of the target data, and it allows us to train a robust LM even for small datasets. We propose *discriminative fine-tuning* and *slanted triangular learning rates* for fine-tuning the LM, which we introduce in the following.

Discriminative fine-tuning As different layers capture *different types of information* (Yosinski et al., 2014), they should be fine-tuned to *different extents*. To this end, we propose a novel fine-

tuning method, *discriminative fine-tuning*[3].

Instead of using the same learning rate for *all* layers of the model, discriminative fine-tuning allows us to tune *each* layer with different learning rates. For context, the regular stochastic gradient descent (SGD) update of a model's parameters θ at time step t looks like the following (Ruder, 2016):

$$\theta_t = \theta_{t-1} - \eta \cdot \nabla_\theta J(\theta) \tag{1}$$

where η is the learning rate and $\nabla_\theta J(\theta)$ is the gradient with regard to the model's objective function. For discriminative fine-tuning, we split the parameters θ into $\{\theta^1, \ldots, \theta^L\}$ where θ^l contains the parameters of the model at the l-th layer and L is the number of layers of the model. Similarly, we obtain $\{\eta^1, \ldots, \eta^L\}$ where η^l is the learning rate of the l-th layer.

The SGD update with discriminative fine-tuning is then the following:

$$\theta_t^l = \theta_{t-1}^l - \eta^l \cdot \nabla_{\theta^l} J(\theta) \tag{2}$$

We empirically found it to work well to first choose the learning rate η^L of the last layer by fine-tuning only the last layer and using $\eta^{l-1} = \eta^l / 2.6$ as the learning rate for lower layers.

Slanted triangular learning rates For adapting its parameters to task-specific features, we would like the model to quickly converge to a suitable region of the parameter space in the beginning of training and then refine its parameters. Using the same learning rate (LR) or an annealed learning rate throughout training is not the best way to achieve this behaviour. Instead, we propose *slanted triangular learning rates* (STLR), which first linearly increases the learning rate and then linearly decays it according to the following update schedule, which can be seen in Figure 2:

$$\begin{aligned} cut &= \lfloor T \cdot cut\_frac \rfloor \\ p &= \begin{cases} t/cut, & \text{if } t < cut \\ 1 - \frac{t - cut}{cut \cdot (1/cut\_frac - 1)}, & \text{otherwise} \end{cases} \\ \eta_t &= \eta_{max} \cdot \frac{1 + p \cdot (ratio - 1)}{ratio} \end{aligned} \tag{3}$$

where T is the number of training iterations[4], cut_frac is the fraction of iterations we increase the LR, cut is the iteration when we switch from increasing to decreasing the LR, p is the fraction of the number of iterations we have increased or will decrease the LR respectively, $ratio$ specifies how much smaller the lowest LR is from the maximum LR η_{max}, and η_t is the learning rate at iteration t. We generally use $cut\_frac = 0.1$, $ratio = 32$ and $\eta_{max} = 0.01$.

STLR modifies triangular learning rates (Smith, 2017) with a short increase and a long decay period, which we found key for good performance.[5] In Section 5, we compare against aggressive cosine annealing, a similar schedule that has recently been used to achieve state-of-the-art performance in CV (Loshchilov and Hutter, 2017).[6]

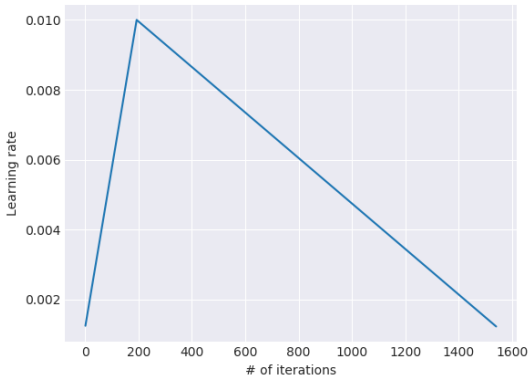


Figure 2: The slanted triangular learning rate schedule used for ULMFiT as a function of the number of training iterations.

3.3 Target task classifier fine-tuning

Finally, for fine-tuning the classifier, we augment the pretrained language model with two additional linear blocks. Following standard practice for CV classifiers, each block uses batch normalization (Ioffe and Szegedy, 2015) and dropout, with ReLU activations for the intermediate layer and a softmax activation that outputs a probability distribution over target classes at the last layer. Note that the parameters in these task-specific classifier layers are the only ones that are learned from scratch. The first linear layer takes as the input the pooled last hidden layer states.

Concat pooling The signal in text classification tasks is often contained in a few words, which may

[3] An unrelated method of the same name exists for deep Boltzmann machines (Salakhutdinov and Hinton, 2009).

[4] In other words, the number of epochs times the number of updates per epoch.

[5] We also credit personal communication with the author.

[6] While Loshchilov and Hutter (2017) use multiple annealing cycles, we generally found one cycle to work best.

occur anywhere in the document. As input documents can consist of hundreds of words, information may get lost if we only consider the last hidden state of the model. For this reason, we concatenate the hidden state at the last time step $\mathbf{h}_T$ of the document with both the max-pooled and the mean-pooled representation of the hidden states over as many time steps as fit in GPU memory $\mathbf{H} = \{\mathbf{h}_1, \ldots, \mathbf{h}_T\}$:

$$\mathbf{h}_c = [\mathbf{h}_T, \texttt{maxpool}(\mathbf{H}), \texttt{meanpool}(\mathbf{H})] \quad (4)$$

where [] is concatenation.

Fine-tuning the target classifier is the most critical part of the transfer learning method. Overly aggressive fine-tuning will cause catastrophic forgetting, eliminating the benefit of the information captured through language modeling; too cautious fine-tuning will lead to slow convergence (and resultant overfitting). Besides discriminative fine-tuning and triangular learning rates, we propose *gradual unfreezing* for fine-tuning the classifier.

Gradual unfreezing Rather than fine-tuning all layers at once, which risks catastrophic forgetting, we propose to gradually unfreeze the model starting from the last layer as this contains the *least general* knowledge (Yosinski et al., 2014): We first unfreeze the last layer and fine-tune all unfrozen layers for one epoch. We then unfreeze the next lower frozen layer and repeat, until we fine-tune all layers until convergence at the last iteration. This is similar to '*chain-thaw*' (Felbo et al., 2017), except that we add a layer at a time to the set of 'thawed' layers, rather than only training a single layer at a time.

While discriminative fine-tuning, slanted triangular learning rates, and gradual unfreezing all are beneficial on their own, we show in Section 5 that they complement each other and enable our method to perform well across diverse datasets.

BPTT for Text Classification (BPT3C) Language models are trained with backpropagation through time (BPTT) to enable gradient propagation for large input sequences. In order to make fine-tuning a classifier for large documents feasible, we propose BPTT for Text Classification (BPT3C): We divide the document into fixed-length batches of size b. At the beginning of each batch, the model is initialized with the final state of the previous batch; we keep track of the hidden states for mean and max-pooling; gradients are back-propagated to the batches whose hidden states contributed to the final prediction. In practice, we use variable length backpropagation sequences (Merity et al., 2017a).

Dataset	Type	# classes	# examples
TREC-6	Question	6	5.5k
IMDb	Sentiment	2	25k
Yelp-bi	Sentiment	2	560k
Yelp-full	Sentiment	5	650k
AG	Topic	4	120k
DBpedia	Topic	14	560k

Table 1: Text classification datasets and tasks with number of classes and training examples.

Bidirectional language model Similar to existing work (Peters et al., 2017, 2018), we are not limited to fine-tuning a unidirectional language model. For all our experiments, we pretrain both a forward and a backward LM. We fine-tune a classifier for each LM independently using BPT3C and average the classifier predictions.

4 Experiments

While our approach is equally applicable to sequence labeling tasks, we focus on text classification tasks in this work due to their important real-world applications.

4.1 Experimental setup

Datasets and tasks We evaluate our method on six widely-studied datasets, with varying numbers of documents and varying document length, used by state-of-the-art text classification and transfer learning approaches (Johnson and Zhang, 2017; McCann et al., 2017) as instances of three common text classification tasks: sentiment analysis, question classification, and topic classification. We show the statistics for each dataset and task in Table 1.

Sentiment Analysis For sentiment analysis, we evaluate our approach on the binary movie review IMDb dataset (Maas et al., 2011) and on the binary and five-class version of the Yelp review dataset compiled by Zhang et al. (2015).

Question Classification We use the six-class version of the small TREC dataset (Voorhees and Tice, 1999) dataset of open-domain, fact-based questions divided into broad semantic categories.

	Model	Test		Model	Test
IMDb	CoVe (McCann et al., 2017)	8.2	TREC-6	CoVe (McCann et al., 2017)	4.2
	oh-LSTM (Johnson and Zhang, 2016)	5.9		TBCNN (Mou et al., 2015)	4.0
	Virtual (Miyato et al., 2016)	5.9		LSTM-CNN (Zhou et al., 2016)	3.9
	ULMFiT (ours)	**4.6**		ULMFiT (ours)	**3.6**

Table 2: Test error rates (%) on two text classification datasets used by McCann et al. (2017).

	AG	DBpedia	Yelp-bi	Yelp-full
Char-level CNN (Zhang et al., 2015)	9.51	1.55	4.88	37.95
CNN (Johnson and Zhang, 2016)	6.57	0.84	2.90	32.39
DPCNN (Johnson and Zhang, 2017)	6.87	0.88	2.64	30.58
ULMFiT (ours)	**5.01**	**0.80**	**2.16**	**29.98**

Table 3: Test error rates (%) on text classification datasets used by Johnson and Zhang (2017).

Topic classification For topic classification, we evaluate on the large-scale AG news and DBpedia ontology datasets created by Zhang et al. (2015).

Pre-processing We use the same pre-processing as in earlier work (Johnson and Zhang, 2017; McCann et al., 2017). In addition, to allow the language model to capture aspects that might be relevant for classification, we add special tokens for upper-case words, elongation, and repetition.

Hyperparameters We are interested in a model that performs robustly across a diverse set of tasks. To this end, if not mentioned otherwise, we use the same set of hyperparameters across tasks, which we tune on the IMDb validation set. We use the AWD-LSTM language model (Merity et al., 2017a) with an embedding size of 400, 3 layers, 1150 hidden activations per layer, and a BPTT batch size of 70. We apply dropout of 0.4 to layers, 0.3 to RNN layers, 0.4 to input embedding layers, 0.05 to embedding layers, and weight dropout of 0.5 to the RNN hidden-to-hidden matrix. The classifier has a hidden layer of size 50. We use Adam with $\beta_1 = 0.7$ instead of the default $\beta_1 = 0.9$ and $\beta_2 = 0.99$, similar to (Dozat and Manning, 2017). We use a batch size of 64, a base learning rate of 0.004 and 0.01 for fine-tuning the LM and the classifier respectively, and tune the number of epochs on the validation set of each task[7]. We otherwise use the same practices used in (Merity et al., 2017a).

Baselines and comparison models For each task, we compare against the current state-of-the-art. For the IMDb and TREC-6 datasets, we compare against CoVe (McCann et al., 2017), a state-of-the-art transfer learning method for NLP. For the AG, Yelp, and DBpedia datasets, we compare against the state-of-the-art text categorization method by Johnson and Zhang (2017).

4.2 Results

For consistency, we report all results as error rates (lower is better). We show the test error rates on the IMDb and TREC-6 datasets used by McCann et al. (2017) in Table 2. Our method outperforms both CoVe, a state-of-the-art transfer learning method based on hypercolumns, as well as the state-of-the-art on both datasets. On IMDb, we reduce the error dramatically by 43.9% and 22% with regard to CoVe and the state-of-the-art respectively. This is promising as the existing state-of-the-art requires complex architectures (Peters et al., 2018), multiple forms of attention (McCann et al., 2017) and sophisticated embedding schemes (Johnson and Zhang, 2016), while our method employs a regular LSTM with dropout. We note that the language model fine-tuning approach of Dai and Le (2015) only achieves an error of 7.64 vs. 4.6 for our method on IMDb, demonstrating the benefit of transferring knowledge from a large ImageNet-like corpus using our fine-tuning techniques. IMDb in particular is reflective of real-world datasets: Its documents are generally a few

[7] On small datasets such as TREC-6, we fine-tune the LM only for 15 epochs without overfitting, while we can fine-tune longer on larger datasets. We found 50 epochs to be a good default for fine-tuning the classifier.

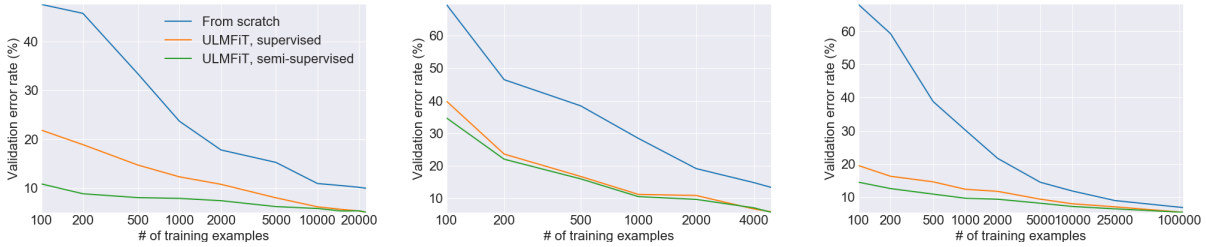


Figure 3: Validation error rates for supervised and semi-supervised ULMFiT vs. training from scratch with different numbers of training examples on IMDb, TREC-6, and AG (from left to right).

paragraphs long—similar to emails (e.g for legal discovery) and online comments (e.g for community management); and sentiment analysis is similar to many commercial applications, e.g. product response tracking and support email routing.

On TREC-6, our improvement—similar as the improvements of state-of-the-art approaches—is not statistically significant, due to the small size of the 500-examples test set. Nevertheless, the competitive performance on TREC-6 demonstrates that our model performs well across different dataset sizes and can deal with examples that range from single sentences—in the case of TREC-6—to several paragraphs for IMDb. Note that despite pretraining on more than two orders of magnitude less data than the 7 million sentence pairs used by McCann et al. (2017), we consistently outperform their approach on both datasets.

We show the test error rates on the larger AG, DBpedia, Yelp-bi, and Yelp-full datasets in Table 3. Our method again outperforms the state-of-the-art significantly. On AG, we observe a similarly dramatic error reduction by 23.7% compared to the state-of-the-art. On DBpedia, Yelp-bi, and Yelp-full, we reduce the error by 4.8%, 18.2%, 2.0% respectively.

5 Analysis

In order to assess the impact of each contribution, we perform a series of analyses and ablations. We run experiments on three corpora, IMDb, TREC-6, and AG that are representative of different tasks, genres, and sizes. For all experiments, we split off 10% of the training set and report error rates on this validation set with unidirectional LMs. We fine-tune the classifier for 50 epochs and train all methods but ULMFiT with early stopping.

Low-shot learning One of the main benefits of transfer learning is being able to train a model for a task with a small number of labels. We evaluate ULMFiT on different numbers of labeled examples in two settings: only labeled examples are used for LM fine-tuning ('*supervised*'); and all task data is available and can be used to fine-tune the LM ('*semi-supervised*'). We compare ULMFiT to training from scratch—which is necessary for hypercolumn-based approaches. We split off balanced fractions of the training data, keep the validation set fixed, and use the same hyperparameters as before. We show the results in Figure 3.

On IMDb and AG, supervised ULMFiT with only 100 labeled examples matches the performance of training from scratch with 10× and 20× more data respectively, clearly demonstrating the benefit of general-domain LM pretraining. If we allow ULMFiT to also utilize unlabeled examples (50k for IMDb, 100k for AG), at 100 labeled examples, we match the performance of training from scratch with 50× and 100× more data on AG and IMDb respectively. On TREC-6, ULMFiT significantly improves upon training from scratch; as examples are shorter and fewer, supervised and semi-supervised ULMFiT achieve similar results.

Pretraining	IMDb	TREC-6	AG
Without pretraining	5.63	10.67	5.52
With pretraining	**5.00**	**5.69**	**5.38**

Table 4: Validation error rates for ULMFiT with and without pretraining.

Impact of pretraining We compare using no pretraining with pretraining on WikiText-103 (Merity et al., 2017b) in Table 4. Pretraining is most useful for small and medium-sized datasets, which are most common in commercial applications. However, even for large datasets, pretraining improves performance.

LM	IMDb	TREC-6	AG
Vanilla LM	5.98	7.41	5.76
AWD-LSTM LM	**5.00**	**5.69**	**5.38**

Table 5: Validation error rates for ULMFiT with a vanilla LM and the AWD-LSTM LM.

LM fine-tuning	IMDb	TREC-6	AG
No LM fine-tuning	6.99	6.38	6.09
Full	5.86	6.54	5.61
Full + discr	5.55	6.36	5.47
Full + discr + stlr	**5.00**	**5.69**	**5.38**

Table 6: Validation error rates for ULMFiT with different variations of LM fine-tuning.

Impact of LM quality In order to gauge the importance of choosing an appropriate LM, we compare a vanilla LM with the same hyperparameters without any dropout[8] with the AWD-LSTM LM with tuned dropout parameters in Table 5. Using our fine-tuning techniques, even a regular LM reaches surprisingly good performance on the larger datasets. On the smaller TREC-6, a vanilla LM without dropout runs the risk of overfitting, which decreases performance.

Impact of LM fine-tuning We compare no fine-tuning against fine-tuning the full model (Erhan et al., 2010) ('*Full*'), the most commonly used fine-tuning method, with and without discriminative fine-tuning ('*Discr*') and slanted triangular learning rates ('*Stlr*') in Table 6. Fine-tuning the LM is most beneficial for larger datasets. '*Discr*' and '*Stlr*' improve performance across all three datasets and are necessary on the smaller TREC-6, where regular fine-tuning is not beneficial.

Impact of classifier fine-tuning We compare training from scratch, fine-tuning the full model ('*Full*'), only fine-tuning the last layer ('*Last*') (Donahue et al., 2014), '*Chain-thaw*' (Felbo et al., 2017), and gradual unfreezing ('*Freez*'). We furthermore assess the importance of discriminative fine-tuning ('*Discr*') and slanted triangular learning rates ('*Stlr*'). We compare the latter to an alternative, aggressive cosine annealing schedule ('*Cos*') (Loshchilov and Hutter, 2017). We use a learning rate $\eta^L = 0.01$ for '*Discr*', learning rates of 0.001 and 0.0001 for the last and all other layers respectively for '*Chain-thaw*' as in (Felbo et al., 2017), and a learning rate of 0.001 otherwise. We show the results in Table 7.

[8]To avoid overfitting, we only train the vanilla LM classifier for 5 epochs and keep dropout of 0.4 in the classifier.

Classifier fine-tuning	IMDb	TREC-6	AG
From scratch	9.93	13.36	6.81
Full	6.87	6.86	5.81
Full + discr	5.57	6.21	5.62
Last	6.49	16.09	8.38
Chain-thaw	5.39	6.71	5.90
Freez	6.37	6.86	5.81
Freez + discr	5.39	5.86	6.04
Freez + stlr	5.04	6.02	5.35
Freez + cos	5.70	6.38	**5.29**
Freez + discr + stlr	**5.00**	**5.69**	5.38

Table 7: Validation error rates for ULMFiT with different methods to fine-tune the classifier.

Fine-tuning the classifier significantly improves over training from scratch, particularly on the small TREC-6. '*Last*', the standard fine-tuning method in CV, severely underfits and is never able to lower the training error to 0. '*Chain-thaw*' achieves competitive performance on the smaller datasets, but is outperformed significantly on the large AG. '*Freez*' provides similar performance as '*Full*'. '*Discr*' consistently boosts the performance of '*Full*' and '*Freez*', except for the large AG. Cosine annealing is competitive with slanted triangular learning rates on large data, but under-performs on smaller datasets. Finally, full ULMFiT classifier fine-tuning (bottom row) achieves the best performance on IMDB and TREC-6 and competitive performance on AG. Importantly, ULMFiT is the only method that shows excellent performance across the board—and is therefore the only *universal* method.

Classifier fine-tuning behavior While our results demonstrate that *how* we fine-tune the classifier makes a significant difference, fine-tuning for inductive transfer is currently under-explored in NLP as it mostly has been thought to be unhelpful (Mou et al., 2016). To better understand the fine-tuning behavior of our model, we compare the validation error of the classifier fine-tuned with ULMFiT and '*Full*' during training in Figure 4.

On all datasets, fine-tuning the full model leads to the lowest error comparatively early in training, e.g. already after the first epoch on IMDb.

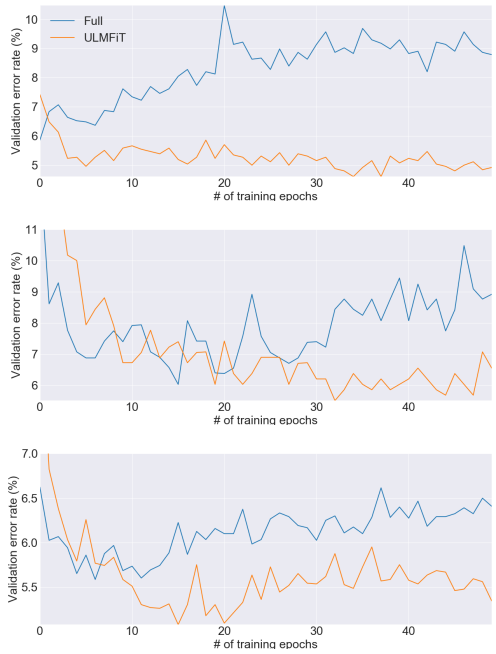


Figure 4: Validation error rate curves for fine-tuning the classifier with ULMFiT and '*Full*' on IMDb, TREC-6, and AG (top to bottom).

The error then increases as the model starts to overfit and knowledge captured through pretraining is lost. In contrast, ULMFiT is more stable and suffers from no such catastrophic forgetting; performance remains similar or improves until late epochs, which shows the positive effect of the learning rate schedule.

Impact of bidirectionality At the cost of training a second model, ensembling the predictions of a forward and backwards LM-classifier brings a performance boost of around 0.5–0.7. On IMDb we lower the test error from 5.30 of a single model to 4.58 for the bidirectional model.

6 Discussion and future directions

While we have shown that ULMFiT can achieve state-of-the-art performance on widely used text classification tasks, we believe that language model fine-tuning will be particularly useful in the following settings compared to existing transfer learning approaches (Conneau et al., 2017; McCann et al., 2017; Peters et al., 2018): a) NLP for non-English languages, where training data for supervised pretraining tasks is scarce; b) new NLP tasks where no state-of-the-art architecture exists; and c) tasks with limited amounts of labeled data (and some amounts of unlabeled data).

Given that transfer learning and particularly fine-tuning for NLP is under-explored, many future directions are possible. One possible direction is to improve language model pretraining and fine-tuning and make them more scalable: for ImageNet, predicting far fewer classes only incurs a small performance drop (Huh et al., 2016), while recent work shows that an alignment between source and target task label sets is important (Mahajan et al., 2018)—focusing on predicting a subset of words such as the most frequent ones might retain most of the performance while speeding up training. Language modeling can also be augmented with additional tasks in a multi-task learning fashion (Caruana, 1993) or enriched with additional supervision, e.g. syntax-sensitive dependencies (Linzen et al., 2016) to create a model that is more general or better suited for certain downstream tasks, ideally in a weakly-supervised manner to retain its universal properties.

Another direction is to apply the method to novel tasks and models. While an extension to sequence labeling is straightforward, other tasks with more complex interactions such as entailment or question answering may require novel ways to pretrain and fine-tune. Finally, while we have provided a series of analyses and ablations, more studies are required to better understand what knowledge a pretrained language model captures, how this changes during fine-tuning, and what information different tasks require.

7 Conclusion

We have proposed ULMFiT, an effective and extremely sample-efficient transfer learning method that can be applied to any NLP task. We have also proposed several novel fine-tuning techniques that in conjunction prevent catastrophic forgetting and enable robust learning across a diverse range of tasks. Our method significantly outperformed existing transfer learning techniques and the state-of-the-art on six representative text classification tasks. We hope that our results will catalyze new developments in transfer learning for NLP.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. Sebastian is supported by Irish Research Council Grant Number EBPPG/2014/30 and Science Foundation Ireland Grant Number SFI/12/RC/2289.

References

Jonathan Baxter. 2000. A Model of Inductive Bias Learning. *Journal of Artificial Intelligence Research* 12:149–198.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. *Annual Meeting-Association for Computational Linguistics* 45(1):440. https://doi.org/10.1109/IRPS.2011.5784441.

Cornelia Caragea, Nathan McNeese, Anuj Jaiswal, Greg Traylor, Hyun-Woo Kim, Prasenjit Mitra, Dinghao Wu, Andrea H Tapia, Lee Giles, Bernard J Jansen, et al. 2011. Classifying text messages for the haiti earthquake. In *Proceedings of the 8th international conference on information systems for crisis response and management (ISCRAM2011)*. Citeseer.

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2017. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks pages 1–10.

Zi Chu, Steven Gianvecchio, Haining Wang, and Sushil Jajodia. 2012. Detecting automation of twitter accounts: Are you a human, bot, or cyborg? *IEEE Transactions on Dependable and Secure Computing* 9(6):811–824.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised Sequence Learning. *Advances in Neural Information Processing Systems (NIPS '15)* http://arxiv.org/abs/1511.01432.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*. pages 647–655.

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *Proceedings of ICLR 2017*.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research* 11(Feb):625–660.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of NAACL-HLT 2018*.

Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. 2015. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 447–456.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Gao Huang, Zhuang Liu, Kilian Q. Weinberger, and Laurens van der Maaten. 2017. Densely Connected Convolutional Networks. In *Proceedings of CVPR 2017*.

Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. 2016. What makes ImageNet good for transfer learning? *arXiv preprint arXiv:1608.08614* .

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*. pages 448–456.

Nitin Jindal and Bing Liu. 2007. Review spam detection. In *Proceedings of the 16th international conference on World Wide Web*. ACM, pages 1189–1190.

Rie Johnson and Tong Zhang. 2016. Supervised and semi-supervised text categorization using lstm for region embeddings. In *International Conference on Machine Learning*. pages 526–534.

Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 562–570.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *arXiv preprint arXiv:1611.01368* .

Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. In *Proceedings of AAAI 2018*.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015a. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 3431–3440.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. 2015b. Learning Transferable Features with Deep Adaptation Networks. In *Proceedings of the 32nd International Conference on Machine learning (ICML '15)*. volume 37.

Ilya Loshchilov and Frank Hutter. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. In *Proceedings of the Internal Conference on Learning Representations 2017*.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 142–150.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. 2018. Exploring the Limits of Weakly Supervised Pretraining .

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in Translation: Contextualized Word Vectors. In *Advances in Neural Information Processing Systems*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017a. Regularizing and Optimizing LSTM Language Models. *arXiv preprint arXiv:1708.02182* .

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017b. Pointer Sentinel Mixture Models. In *Proceedings of the International Conference on Learning Representations 2017*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*.

Sewon Min, Minjoon Seo, and Hannaneh Hajishirzi. 2017. Question Answering through Transfer Learning from Large Fine-grained Supervision Data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Short Papers)*.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725* .

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How Transferable are Neural Networks in NLP Applications? *Proceedings of 2016 Conference on Empirical Methods in Natural Language Processing* .

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

EWT Ngai, Yong Hu, YH Wong, Yijun Chen, and Xin Sun. 2011. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems* 50(3):559–569.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 22(10):1345–1359.

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of ACL 2017*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL 2018*.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444* .

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of ACL 2017*.

Herbert L Roitblat, Anne Kershaw, and Patrick Oot. 2010. Document categorization in legal electronic discovery: computer classification vs. manual review. *Journal of the Association for Information Science and Technology* 61(1):70–80.

Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747* .

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Deep boltzmann machines. In *Artificial Intelligence and Statistics*. pages 448–455.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709* .

Aliaksei Severyn and Alessandro Moschitti. 2015. UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification. *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* pages 464–469.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 806–813.

Leslie N Smith. 2017. Cyclical learning rates for training neural networks. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*. IEEE, pages 464–472.

Vladimir Naumovich Vapnik and Samuel Kotz. 1982. *Estimation of dependences based on empirical data*, volume 40. Springer-Verlag New York.

Ellen M Voorhees and Dawn M Tice. 1999. The trec-8 question answering track evaluation. In *TREC*. volume 1999, page 82.

John Wieting and Kevin Gimpel. 2017. Revisiting Recurrent Networks for Paraphrastic Sentence Embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in neural information processing systems*. pages 3320–3328.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*. pages 649–657.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *Proceedings of COLING 2016*.

Deep contextualized word representations

Matthew E. Peters[†], **Mark Neumann**[†], **Mohit Iyyer**[†], **Matt Gardner**[†],
{matthewp,markn,mohiti,mattg}@allenai.org

Christopher Clark[*], **Kenton Lee**[*], **Luke Zettlemoyer**[†*]
{csquared,kentonl,lsz}@cs.washington.edu

[†]Allen Institute for Artificial Intelligence
[*]Paul G. Allen School of Computer Science & Engineering, University of Washington

Abstract

We introduce a new type of *deep contextualized* word representation that models both (1) complex characteristics of word use (e.g., syntax and semantics), and (2) how these uses vary across linguistic contexts (i.e., to model polysemy). Our word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large text corpus. We show that these representations can be easily added to existing models and significantly improve the state of the art across six challenging NLP problems, including question answering, textual entailment and sentiment analysis. We also present an analysis showing that exposing the deep internals of the pre-trained network is crucial, allowing downstream models to mix different types of semi-supervision signals.

1 Introduction

Pre-trained word representations (Mikolov et al., 2013; Pennington et al., 2014) are a key component in many neural language understanding models. However, learning high quality representations can be challenging. They should ideally model both (1) complex characteristics of word use (e.g., syntax and semantics), and (2) how these uses vary across linguistic contexts (i.e., to model polysemy). In this paper, we introduce a new type of *deep contextualized* word representation that directly addresses both challenges, can be easily integrated into existing models, and significantly improves the state of the art in every considered case across a range of challenging language understanding problems.

Our representations differ from traditional word type embeddings in that each token is assigned a representation that is a function of the entire input sentence. We use vectors derived from a bidirectional LSTM that is trained with a coupled language model (LM) objective on a large text corpus. For this reason, we call them ELMo (Embeddings from Language Models) representations. Unlike previous approaches for learning contextualized word vectors (Peters et al., 2017; McCann et al., 2017), ELMo representations are deep, in the sense that they are a function of all of the internal layers of the biLM. More specifically, we learn a linear combination of the vectors stacked above each input word for each end task, which markedly improves performance over just using the top LSTM layer.

Combining the internal states in this manner allows for very rich word representations. Using intrinsic evaluations, we show that the higher-level LSTM states capture context-dependent aspects of word meaning (e.g., they can be used without modification to perform well on supervised word sense disambiguation tasks) while lower-level states model aspects of syntax (e.g., they can be used to do part-of-speech tagging). Simultaneously exposing all of these signals is highly beneficial, allowing the learned models select the types of semi-supervision that are most useful for each end task.

Extensive experiments demonstrate that ELMo representations work extremely well in practice. We first show that they can be easily added to existing models for six diverse and challenging language understanding problems, including textual entailment, question answering and sentiment analysis. The addition of ELMo representations alone significantly improves the state of the art in every case, including up to 20% relative error reductions. For tasks where direct comparisons are possible, ELMo outperforms CoVe (McCann et al., 2017), which computes contextualized representations using a neural machine translation encoder. Finally, an analysis of both ELMo and CoVe reveals that deep representations outperform

those derived from just the top layer of an LSTM. Our trained models and code are publicly available, and we expect that ELMo will provide similar gains for many other NLP problems.[1]

2 Related work

Due to their ability to capture syntactic and semantic information of words from large scale unlabeled text, pretrained word vectors (Turian et al., 2010; Mikolov et al., 2013; Pennington et al., 2014) are a standard component of most state-of-the-art NLP architectures, including for question answering (Liu et al., 2017), textual entailment (Chen et al., 2017) and semantic role labeling (He et al., 2017). However, these approaches for learning word vectors only allow a single context-independent representation for each word.

Previously proposed methods overcome some of the shortcomings of traditional word vectors by either enriching them with subword information (e.g., Wieting et al., 2016; Bojanowski et al., 2017) or learning separate vectors for each word sense (e.g., Neelakantan et al., 2014). Our approach also benefits from subword units through the use of character convolutions, and we seamlessly incorporate multi-sense information into downstream tasks without explicitly training to predict predefined sense classes.

Other recent work has also focused on learning context-dependent representations. `context2vec` (Melamud et al., 2016) uses a bidirectional Long Short Term Memory (LSTM; Hochreiter and Schmidhuber, 1997) to encode the context around a pivot word. Other approaches for learning contextual embeddings include the pivot word itself in the representation and are computed with the encoder of either a supervised neural machine translation (MT) system (CoVe; McCann et al., 2017) or an unsupervised language model (Peters et al., 2017). Both of these approaches benefit from large datasets, although the MT approach is limited by the size of parallel corpora. In this paper, we take full advantage of access to plentiful monolingual data, and train our biLM on a corpus with approximately 30 million sentences (Chelba et al., 2014). We also generalize these approaches to deep contextual representations, which we show work well across a broad range of diverse NLP tasks.

[1] `http://allennlp.org/elmo`

Previous work has also shown that different layers of deep biRNNs encode different types of information. For example, introducing multi-task syntactic supervision (e.g., part-of-speech tags) at the lower levels of a deep LSTM can improve overall performance of higher level tasks such as dependency parsing (Hashimoto et al., 2017) or CCG super tagging (Søgaard and Goldberg, 2016). In an RNN-based encoder-decoder machine translation system, Belinkov et al. (2017) showed that the representations learned at the first layer in a 2-layer LSTM encoder are better at predicting POS tags then second layer. Finally, the top layer of an LSTM for encoding word context (Melamud et al., 2016) has been shown to learn representations of word sense. We show that similar signals are also induced by the modified language model objective of our ELMo representations, and it can be very beneficial to learn models for downstream tasks that mix these different types of semi-supervision.

Dai and Le (2015) and Ramachandran et al. (2017) pretrain encoder-decoder pairs using language models and sequence autoencoders and then fine tune with task specific supervision. In contrast, after pretraining the biLM with unlabeled data, we fix the weights and add additional task-specific model capacity, allowing us to leverage large, rich and universal biLM representations for cases where downstream training data size dictates a smaller supervised model.

3 ELMo: Embeddings from Language Models

Unlike most widely used word embeddings (Pennington et al., 2014), ELMo word representations are functions of the entire input sentence, as described in this section. They are computed on top of two-layer biLMs with character convolutions (Sec. 3.1), as a linear function of the internal network states (Sec. 3.2). This setup allows us to do semi-supervised learning, where the biLM is pretrained at a large scale (Sec. 3.4) and easily incorporated into a wide range of existing neural NLP architectures (Sec. 3.3).

3.1 Bidirectional language models

Given a sequence of N tokens, $(t_1, t_2, ..., t_N)$, a forward language model computes the probability of the sequence by modeling the probability of to-

ken t_k given the history $(t_1, ..., t_{k-1})$:

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_1, t_2, \ldots, t_{k-1}).$$

Recent state-of-the-art neural language models (Józefowicz et al., 2016; Melis et al., 2017; Merity et al., 2017) compute a context-independent token representation $\mathbf{x}_k^{LM}$ (via token embeddings or a CNN over characters) then pass it through L layers of forward LSTMs. At each position k, each LSTM layer outputs a context-dependent representation $\overrightarrow{\mathbf{h}}_{k,j}^{LM}$ where $j = 1, \ldots, L$. The top layer LSTM output, $\overrightarrow{\mathbf{h}}_{k,L}^{LM}$, is used to predict the next token t_{k+1} with a Softmax layer.

A backward LM is similar to a forward LM, except it runs over the sequence in reverse, predicting the previous token given the future context:

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_{k+1}, t_{k+2}, \ldots, t_N).$$

It can be implemented in an analogous way to a forward LM, with each backward LSTM layer j in a L layer deep model producing representations $\overleftarrow{\mathbf{h}}_{k,j}^{LM}$ of t_k given $(t_{k+1}, \ldots, t_N)$.

A biLM combines both a forward and backward LM. Our formulation jointly maximizes the log likelihood of the forward and backward directions:

$$\sum_{k=1}^{N} \Big(\log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \Big).$$

We tie the parameters for both the token representation (Θ_x) and Softmax layer (Θ_s) in the forward and backward direction while maintaining separate parameters for the LSTMs in each direction. Overall, this formulation is similar to the approach of Peters et al. (2017), with the exception that we share some weights between directions instead of using completely independent parameters. In the next section, we depart from previous work by introducing a new approach for learning word representations that are a linear combination of the biLM layers.

3.2 ELMo

ELMo is a task specific combination of the intermediate layer representations in the biLM. For each token t_k, a L-layer biLM computes a set of $2L + 1$ representations

$$\begin{aligned} R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \ldots, L\} \\ &= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\}, \end{aligned}$$

where $\mathbf{h}_{k,0}^{LM}$ is the token layer and $\mathbf{h}_{k,j}^{LM} = [\overrightarrow{\mathbf{h}}_{k,j}^{LM}; \overleftarrow{\mathbf{h}}_{k,j}^{LM}]$, for each biLSTM layer.

For inclusion in a downstream model, ELMo collapses all layers in R into a single vector, $\mathbf{ELMo}_k = E(R_k; \mathbf{\Theta}_e)$. In the simplest case, ELMo just selects the top layer, $E(R_k) = \mathbf{h}_{k,L}^{LM}$, as in TagLM (Peters et al., 2017) and CoVe (McCann et al., 2017). More generally, we compute a task specific weighting of all biLM layers:

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}. \quad (1)$$

In (1), $\mathbf{s}^{task}$ are softmax-normalized weights and the scalar parameter γ^{task} allows the task model to scale the entire ELMo vector. γ is of practical importance to aid the optimization process (see supplemental material for details). Considering that the activations of each biLM layer have a different distribution, in some cases it also helped to apply layer normalization (Ba et al., 2016) to each biLM layer before weighting.

3.3 Using biLMs for supervised NLP tasks

Given a pre-trained biLM and a supervised architecture for a target NLP task, it is a simple process to use the biLM to improve the task model. We simply run the biLM and record all of the layer representations for each word. Then, we let the end task model learn a linear combination of these representations, as described below.

First consider the lowest layers of the supervised model without the biLM. Most supervised NLP models share a common architecture at the lowest layers, allowing us to add ELMo in a consistent, unified manner. Given a sequence of tokens $(t_1, \ldots, t_N)$, it is standard to form a context-independent token representation $\mathbf{x}_k$ for each token position using pre-trained word embeddings and optionally character-based representations. Then, the model forms a context-sensitive representation $\mathbf{h}_k$, typically using either bidirectional RNNs, CNNs, or feed forward networks.

To add ELMo to the supervised model, we first freeze the weights of the biLM and then

concatenate the ELMo vector $\mathbf{ELMo}_k^{task}$ with $\mathbf{x}_k$ and pass the ELMo enhanced representation $[\mathbf{x}_k; \mathbf{ELMo}_k^{task}]$ into the task RNN. For some tasks (e.g., SNLI, SQuAD), we observe further improvements by also including ELMo at the output of the task RNN by introducing another set of output specific linear weights and replacing $\mathbf{h}_k$ with $[\mathbf{h}_k; \mathbf{ELMo}_k^{task}]$. As the remainder of the supervised model remains unchanged, these additions can happen within the context of more complex neural models. For example, see the SNLI experiments in Sec. 4 where a bi-attention layer follows the biLSTMs, or the coreference resolution experiments where a clustering model is layered on top of the biLSTMs.

Finally, we found it beneficial to add a moderate amount of dropout to ELMo (Srivastava et al., 2014) and in some cases to regularize the ELMo weights by adding $\lambda\|\mathbf{w}\|_2^2$ to the loss. This imposes an inductive bias on the ELMo weights to stay close to an average of all biLM layers.

3.4 Pre-trained bidirectional language model architecture

The pre-trained biLMs in this paper are similar to the architectures in Józefowicz et al. (2016) and Kim et al. (2015), but modified to support joint training of both directions and add a residual connection between LSTM layers. We focus on large scale biLMs in this work, as Peters et al. (2017) highlighted the importance of using biLMs over forward-only LMs and large scale training.

To balance overall language model perplexity with model size and computational requirements for downstream tasks while maintaining a purely character-based input representation, we halved all embedding and hidden dimensions from the single best model `CNN-BIG-LSTM` in Józefowicz et al. (2016). The final model uses $L = 2$ biLSTM layers with 4096 units and 512 dimension projections and a residual connection from the first to second layer. The context insensitive type representation uses 2048 character n-gram convolutional filters followed by two highway layers (Srivastava et al., 2015) and a linear projection down to a 512 representation. As a result, the biLM provides three layers of representations for each input token, including those outside the training set due to the purely character input. In contrast, traditional word embedding methods only provide one layer of representation for tokens in a fixed vocabulary.

After training for 10 epochs on the 1B Word Benchmark (Chelba et al., 2014), the average forward and backward perplexities is 39.7, compared to 30.0 for the forward `CNN-BIG-LSTM`. Generally, we found the forward and backward perplexities to be approximately equal, with the backward value slightly lower.

Once pretrained, the biLM can compute representations for any task. In some cases, fine tuning the biLM on domain specific data leads to significant drops in perplexity and an increase in downstream task performance. This can be seen as a type of domain transfer for the biLM. As a result, in most cases we used a fine-tuned biLM in the downstream task. See supplemental material for details.

4 Evaluation

Table 1 shows the performance of ELMo across a diverse set of six benchmark NLP tasks. In every task considered, simply adding ELMo establishes a new state-of-the-art result, with relative error reductions ranging from 6 - 20% over strong base models. This is a very general result across a diverse set model architectures and language understanding tasks. In the remainder of this section we provide high-level sketches of the individual task results; see the supplemental material for full experimental details.

Question answering The Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) contains 100K+ crowd sourced question-answer pairs where the answer is a span in a given Wikipedia paragraph. Our baseline model (Clark and Gardner, 2017) is an improved version of the Bidirectional Attention Flow model in Seo et al. (BiDAF; 2017). It adds a self-attention layer after the bidirectional attention component, simplifies some of the pooling operations and substitutes the LSTMs for gated recurrent units (GRUs; Cho et al., 2014). After adding ELMo to the baseline model, test set F_1 improved by 4.7% from 81.1% to 85.8%, a 24.9% relative error reduction over the baseline, and improving the overall single model state-of-the-art by 1.4%. A 11 member ensemble pushes F_1 to 87.4, the overall state-of-the-art at time of submission to the leaderboard.[2] The increase of 4.7% with ELMo is also significantly larger then the 1.8% improvement from adding CoVe to a baseline model (McCann et al., 2017).

[2] As of November 17, 2017.

Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/ relative)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

Table 1: Test set comparison of ELMo enhanced neural models with state-of-the-art single model baselines across six benchmark NLP tasks. The performance metric varies across tasks – accuracy for SNLI and SST-5; F_1 for SQuAD, SRL and NER; average F_1 for Coref. Due to the small test sizes for NER and SST-5, we report the mean and standard deviation across five runs with different random seeds. The "increase" column lists both the absolute and relative improvements over our baseline.

Textual entailment Textual entailment is the task of determining whether a "hypothesis" is true, given a "premise". The Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015) provides approximately 550K hypothesis/premise pairs. Our baseline, the ESIM sequence model from Chen et al. (2017), uses a biLSTM to encode the premise and hypothesis, followed by a matrix attention layer, a local inference layer, another biLSTM inference composition layer, and finally a pooling operation before the output layer. Overall, adding ELMo to the ESIM model improves accuracy by an average of 0.7% across five random seeds. A five member ensemble pushes the overall accuracy to 89.3%, exceeding the previous ensemble best of 88.9% (Gong et al., 2018).

Semantic role labeling A semantic role labeling (SRL) system models the predicate-argument structure of a sentence, and is often described as answering "Who did what to whom". He et al. (2017) modeled SRL as a BIO tagging problem and used an 8-layer deep biLSTM with forward and backward directions interleaved, following Zhou and Xu (2015). As shown in Table 1, when adding ELMo to a re-implementation of He et al. (2017) the single model test set F_1 jumped 3.2% from 81.4% to 84.6% – a new state-of-the-art on the OntoNotes benchmark (Pradhan et al., 2013), even improving over the previous best ensemble result by 1.2%.

Coreference resolution Coreference resolution is the task of clustering mentions in text that refer to the same underlying real world entities. Our baseline model is the end-to-end span-based neural model of Lee et al. (2017). It uses a biLSTM and attention mechanism to first compute span representations and then applies a softmax mention ranking model to find coreference chains. In our experiments with the OntoNotes coreference annotations from the CoNLL 2012 shared task (Pradhan et al., 2012), adding ELMo improved the average F_1 by 3.2% from 67.2 to 70.4, establishing a new state of the art, again improving over the previous best ensemble result by 1.6% F_1.

Named entity extraction The CoNLL 2003 NER task (Sang and Meulder, 2003) consists of newswire from the Reuters RCV1 corpus tagged with four different entity types (`PER`, `LOC`, `ORG`, `MISC`). Following recent state-of-the-art systems (Lample et al., 2016; Peters et al., 2017), the baseline model uses pre-trained word embeddings, a character-based CNN representation, two biLSTM layers and a conditional random field (CRF) loss (Lafferty et al., 2001), similar to Collobert et al. (2011). As shown in Table 1, our ELMo enhanced biLSTM-CRF achieves 92.22% F_1 averaged over five runs. The key difference between our system and the previous state of the art from Peters et al. (2017) is that we allowed the task model to learn a weighted average of all biLM layers, whereas Peters et al. (2017) only use the top biLM layer. As shown in Sec. 5.1, using all layers instead of just the last layer improves performance across multiple tasks.

Sentiment analysis The fine-grained sentiment classification task in the Stanford Sentiment Treebank (SST-5; Socher et al., 2013) involves selecting one of five labels (from very negative to very positive) to describe a sentence from a movie review. The sentences contain diverse linguistic phenomena such as idioms and complex syntac-

Task	Baseline	Last Only	All layers	
			λ=1	λ=0.001
SQuAD	80.8	84.7	85.0	**85.2**
SNLI	88.1	89.1	89.3	**89.5**
SRL	81.6	84.1	84.6	**84.8**

Table 2: Development set performance for SQuAD, SNLI and SRL comparing using all layers of the biLM (with different choices of regularization strength λ) to just the top layer.

Task	Input Only	Input & Output	Output Only
SQuAD	85.1	**85.6**	84.8
SNLI	88.9	**89.5**	88.7
SRL	**84.7**	84.3	80.9

Table 3: Development set performance for SQuAD, SNLI and SRL when including ELMo at different locations in the supervised model.

tic constructions such as negations that are difficult for models to learn. Our baseline model is the biattentive classification network (BCN) from McCann et al. (2017), which also held the prior state-of-the-art result when augmented with CoVe embeddings. Replacing CoVe with ELMo in the BCN model results in a 1.0% absolute accuracy improvement over the state of the art.

5 Analysis

This section provides an ablation analysis to validate our chief claims and to elucidate some interesting aspects of ELMo representations. Sec. 5.1 shows that using deep contextual representations in downstream tasks improves performance over previous work that uses just the top layer, regardless of whether they are produced from a biLM or MT encoder, and that ELMo representations provide the best overall performance. Sec. 5.3 explores the different types of contextual information captured in biLMs and uses two intrinsic evaluations to show that syntactic information is better represented at lower layers while semantic information is captured a higher layers, consistent with MT encoders. It also shows that our biLM consistently provides richer representations then CoVe. Additionally, we analyze the sensitivity to where ELMo is included in the task model (Sec. 5.2), training set size (Sec. 5.4), and visualize the ELMo learned weights across the tasks (Sec. 5.5).

5.1 Alternate layer weighting schemes

There are many alternatives to Equation 1 for combining the biLM layers. Previous work on contextual representations used only the last layer, whether it be from a biLM (Peters et al., 2017) or an MT encoder (CoVe; McCann et al., 2017). The choice of the regularization parameter λ is also important, as large values such as $\lambda = 1$ effectively reduce the weighting function to a simple average over the layers, while smaller values (e.g., $\lambda = 0.001$) allow the layer weights to vary.

Table 2 compares these alternatives for SQuAD, SNLI and SRL. Including representations from all layers improves overall performance over just using the last layer, and including contextual representations from the last layer improves performance over the baseline. For example, in the case of SQuAD, using just the last biLM layer improves development F_1 by 3.9% over the baseline. Averaging all biLM layers instead of using just the last layer improves F_1 another 0.3% (comparing "Last Only" to λ=1 columns), and allowing the task model to learn individual layer weights improves F_1 another 0.2% (λ=1 vs. λ=0.001). A small λ is preferred in most cases with ELMo, although for NER, a task with a smaller training set, the results are insensitive to λ (not shown).

The overall trend is similar with CoVe but with smaller increases over the baseline. For SNLI, averaging all layers with λ=1 improves development accuracy from 88.2 to 88.7% over using just the last layer. SRL F_1 increased a marginal 0.1% to 82.2 for the λ=1 case compared to using the last layer only.

5.2 Where to include ELMo?

All of the task architectures in this paper include word embeddings only as input to the lowest layer biRNN. However, we find that including ELMo at the output of the biRNN in task-specific architectures improves overall results for some tasks. As shown in Table 3, including ELMo at both the input and output layers for SNLI and SQuAD improves over just the input layer, but for SRL (and coreference resolution, not shown) performance is highest when it is included at just the input layer. One possible explanation for this result is that both the SNLI and SQuAD architectures use attention layers after the biRNN, so introducing ELMo at this layer allows the model to attend directly to the biLM's internal representations. In the SRL case,

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

Table 4: Nearest neighbors to "play" using GloVe and the context embeddings from a biLM.

Model	**F_1**
WordNet 1st Sense Baseline	65.9
Raganato et al. (2017a)	69.9
Iacobacci et al. (2016)	**70.1**
CoVe, First Layer	59.4
CoVe, Second Layer	*64.7*
biLM, First layer	67.4
biLM, Second layer	*69.0*

Table 5: All-words fine grained WSD F_1. For CoVe and the biLM, we report scores for both the first and second layer biLSTMs.

Model	**Acc.**
Collobert et al. (2011)	97.3
Ma and Hovy (2016)	97.6
Ling et al. (2015)	**97.8**
CoVe, First Layer	*93.3*
CoVe, Second Layer	92.8
biLM, First Layer	*97.3*
biLM, Second Layer	96.8

Table 6: Test set POS tagging accuracies for PTB. For CoVe and the biLM, we report scores for both the first and second layer biLSTMs.

the task-specific context representations are likely more important than those from the biLM.

5.3 What information is captured by the biLM's representations?

Since adding ELMo improves task performance over word vectors alone, the biLM's contextual representations must encode information generally useful for NLP tasks that is not captured in word vectors. Intuitively, the biLM must be disambiguating the meaning of words using their context. Consider "play", a highly polysemous word. The top of Table 4 lists nearest neighbors to "play" using GloVe vectors. They are spread across several parts of speech (e.g., "played", "playing" as verbs, and "player", "game" as nouns) but concentrated in the sports-related senses of "play". In contrast, the bottom two rows show nearest neighbor sentences from the SemCor dataset (see below) using the biLM's context representation of "play" in the source sentence. In these cases, the biLM is able to disambiguate both the part of speech and word sense in the source sentence.

These observations can be quantified using an intrinsic evaluation of the contextual representations similar to Belinkov et al. (2017). To isolate the information encoded by the biLM, the representations are used to directly make predictions for a fine grained word sense disambiguation (WSD) task and a POS tagging task. Using this approach, it is also possible to compare to CoVe, and across each of the individual layers.

Word sense disambiguation Given a sentence, we can use the biLM representations to predict the sense of a target word using a simple 1-nearest neighbor approach, similar to Melamud et al. (2016). To do so, we first use the biLM to compute representations for all words in SemCor 3.0, our training corpus (Miller et al., 1994), and then take the average representation for each sense. At test time, we again use the biLM to compute representations for a given target word and take the nearest neighbor sense from the training set, falling back to the first sense from WordNet for lemmas not observed during training.

Table 5 compares WSD results using the evaluation framework from Raganato et al. (2017b) across the same suite of four test sets in Raganato et al. (2017a). Overall, the biLM top layer rep-

resentations have F_1 of 69.0 and are better at WSD then the first layer. This is competitive with a state-of-the-art WSD-specific supervised model using hand crafted features (Iacobacci et al., 2016) and a task specific biLSTM that is also trained with auxiliary coarse-grained semantic labels and POS tags (Raganato et al., 2017a). The CoVe biLSTM layers follow a similar pattern to those from the biLM (higher overall performance at the second layer compared to the first); however, our biLM outperforms the CoVe biLSTM, which trails the WordNet first sense baseline.

POS tagging To examine whether the biLM captures basic syntax, we used the context representations as input to a linear classifier that predicts POS tags with the Wall Street Journal portion of the Penn Treebank (PTB) (Marcus et al., 1993). As the linear classifier adds only a small amount of model capacity, this is direct test of the biLM's representations. Similar to WSD, the biLM representations are competitive with carefully tuned, task specific biLSTMs (Ling et al., 2015; Ma and Hovy, 2016). However, unlike WSD, accuracies using the first biLM layer are higher than the top layer, consistent with results from deep biLSTMs in multi-task training (Søgaard and Goldberg, 2016; Hashimoto et al., 2017) and MT (Belinkov et al., 2017). CoVe POS tagging accuracies follow the same pattern as those from the biLM, and just like for WSD, the biLM achieves higher accuracies than the CoVe encoder.

Implications for supervised tasks Taken together, these experiments confirm different layers in the biLM represent different types of information and explain why including all biLM layers is important for the highest performance in downstream tasks. In addition, the biLM's representations are more transferable to WSD and POS tagging than those in CoVe, helping to illustrate why ELMo outperforms CoVe in downstream tasks.

5.4 Sample efficiency

Adding ELMo to a model increases the sample efficiency considerably, both in terms of number of parameter updates to reach state-of-the-art performance and the overall training set size. For example, the SRL model reaches a maximum development F_1 after 486 epochs of training without ELMo. After adding ELMo, the model exceeds the baseline maximum at epoch 10, a 98% relative decrease in the number of updates needed to reach the same level of performance.

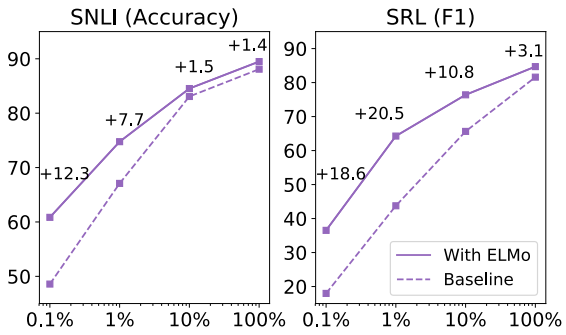


Figure 1: Comparison of baseline vs. ELMo performance for SNLI and SRL as the training set size is varied from 0.1% to 100%.

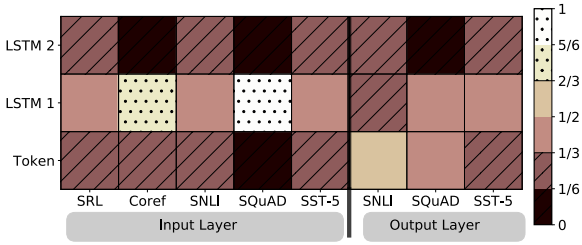


Figure 2: Visualization of softmax normalized biLM layer weights across tasks and ELMo locations. Normalized weights less then $1/3$ are hatched with horizontal lines and those greater then $2/3$ are speckled.

In addition, ELMo-enhanced models use smaller training sets more efficiently than models without ELMo. Figure 1 compares the performance of baselines models with and without ELMo as the percentage of the full training set is varied from 0.1% to 100%. Improvements with ELMo are largest for smaller training sets and significantly reduce the amount of training data needed to reach a given level of performance. In the SRL case, the ELMo model with 1% of the training set has about the same F_1 as the baseline model with 10% of the training set.

5.5 Visualization of learned weights

Figure 2 visualizes the softmax-normalized learned layer weights. At the input layer, the task model favors the first biLSTM layer. For coreference and SQuAD, the this is strongly favored, but the distribution is less peaked for the other tasks. The output layer weights are relatively balanced, with a slight preference for the lower layers.

6 Conclusion

We have introduced a general approach for learning high-quality deep context-dependent representations from biLMs, and shown large improvements when applying ELMo to a broad range of NLP tasks. Through ablations and other controlled experiments, we have also confirmed that the biLM layers efficiently encode different types of syntactic and semantic information about words-in-context, and that using all layers improves overall task performance.

References

Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR* abs/1607.06450.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James R. Glass. 2017. What do neural machine translation models learn about morphology? In *ACL*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL* 5:135–146.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH*.

Qian Chen, Xiao-Dan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *ACL*.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. In *TACL*.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *SSST@EMNLP*.

Christopher Clark and Matthew Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *CoRR* abs/1710.10723.

Kevin Clark and Christopher D. Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. In *EMNLP*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. In *JMLR*.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *NIPS*.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *EMNLP*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *NIPS*.

Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural language inference over interaction space. In *ICLR*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *EMNLP 2017*.

Luheng He, Kenton Lee, Mike Lewis, and Luke S. Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *ACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *ACL*.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR* abs/1602.02410.

Rafal Józefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *ICML*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. In *AAAI 2016*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, Ishaan Gulrajani James Bradbury, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*.

John D. Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL-HLT*.

Kenton Lee, Luheng He, Mike Lewis, and Luke S. Zettlemoyer. 2017. End-to-end neural coreference resolution. In *EMNLP*.

Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *EMNLP*.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2017. Stochastic answer networks for machine reading comprehension. *arXiv preprint arXiv:1712.03556* .

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *ACL*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics* 19:313–330.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS 2017*.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *CoNLL*.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. On the state of the art of evaluation in neural language models. *CoRR* abs/1707.05589.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing lstm language models. *CoRR* abs/1708.02182.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

George A. Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G. Thomas. 1994. Using a semantic concordance for sense identification. In *HLT*.

Tsendsuren Munkhdalai and Hong Yu. 2017. Neural tree indexers for text understanding. In *EACL*.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *EMNLP*.

Martha Palmer, Paul Kingsbury, and Daniel Gildea. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics* 31:71–106.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *CoNLL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *EMNLP-CoNLL Shared Task*.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017a. Neural sequence learning models for word sense disambiguation. In *EMNLP*.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017b. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *EACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*.

Prajit Ramachandran, Peter Liu, and Quoc Le. 2017. Improving sequence to sequence learning with unlabeled data. In *EMNLP*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL 2016*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *NIPS*.

Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *ACL*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *ACL*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *EMNLP*.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning global features for coreference resolution. In *HLT-NAACL*.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *CoRR* abs/1212.5701.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *ACL*.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *COLING*.

A Supplemental Material to accompany *Deep contextualized word representations*

This supplement contains details of the model architectures, training routines and hyper-parameter choices for the state-of-the-art models in Section 4.

All of the individual models share a common architecture in the lowest layers with a context independent token representation below several layers of stacked RNNs – LSTMs in every case except the SQuAD model that uses GRUs.

A.1 Fine tuning biLM

As noted in Sec. 3.4, fine tuning the biLM on task specific data typically resulted in significant drops in perplexity. To fine tune on a given task, the supervised labels were temporarily ignored, the biLM fine tuned for one epoch on the training split and evaluated on the development split. Once fine tuned, the biLM weights were fixed during task training.

Table 7 lists the development set perplexities for the considered tasks. In every case except CoNLL 2012, fine tuning results in a large improvement in perplexity, e.g., from 72.1 to 16.8 for SNLI.

The impact of fine tuning on supervised performance is task dependent. In the case of SNLI, fine tuning the biLM increased development accuracy 0.6% from 88.9% to 89.5% for our single best model. However, for sentiment classification development set accuracy is approximately the same regardless whether a fine tuned biLM was used.

A.2 Importance of γ in Eqn. (1)

The γ parameter in Eqn. (1) was of practical importance to aid optimization, due to the different distributions between the biLM internal representations and the task specific representations. It is especially important in the last-only case in Sec. 5.1. Without this parameter, the last-only case performed poorly (well below the baseline) for SNLI and training failed completely for SRL.

A.3 Textual Entailment

Our baseline SNLI model is the ESIM sequence model from Chen et al. (2017). Following the original implementation, we used 300 dimensions for all LSTM and feed forward layers and pre-trained 300 dimensional GloVe embeddings that were fixed during training. For regularization, we added 50% variational dropout (Gal and Ghahramani, 2016) to the input of each LSTM layer and 50% dropout (Srivastava et al., 2014) at the input to the final two fully connected layers. All feed forward layers use ReLU activations. Parameters were optimized using Adam (Kingma and Ba, 2015) with gradient norms clipped at 5.0 and initial learning rate 0.0004, decreasing by half each time accuracy on the development set did not increase in subsequent epochs. The batch size was 32.

Dataset		Before tuning	After tuning
SNLI		72.1	16.8
CoNLL 2012 (coref/SRL)		92.3	-
CoNLL 2003 (NER)		103.2	46.3
SQuAD	Context	99.1	43.5
	Questions	158.2	52.0
SST		131.5	78.6

Table 7: Development set perplexity before and after fine tuning for one epoch on the training set for various datasets (lower is better). Reported values are the average of the forward and backward perplexities.

The best ELMo configuration added ELMo vectors to both the input and output of the lowest layer LSTM, using (1) with layer normalization and $\lambda = 0.001$. Due to the increased number of parameters in the ELMo model, we added ℓ^2 regularization with regularization coefficient 0.0001 to all recurrent and feed forward weight matrices and 50% dropout after the attention layer.

Table 8 compares test set accuracy of our system to previously published systems. Overall, adding ELMo to the ESIM model improved accuracy by 0.7% establishing a new single model state-of-the-art of 88.7%, and a five member ensemble pushes the overall accuracy to 89.3%.

A.4 Question Answering

Our QA model is a simplified version of the model from Clark and Gardner (2017). It embeds tokens by concatenating each token's case-sensitive 300 dimensional GloVe word vector (Pennington et al., 2014) with a character-derived embedding produced using a convolutional neural network followed by max-pooling on learned character embeddings. The token embeddings are passed through a shared bi-directional GRU, and then the bi-directional attention mechanism from BiDAF (Seo et al., 2017). The augmented con-

Model	Acc.
Feature based (Bowman et al., 2015)	78.2
DIIN (Gong et al., 2018)	88.0
BCN+Char+CoVe (McCann et al., 2017)	88.1
ESIM (Chen et al., 2017)	88.0
ESIM+TreeLSTM (Chen et al., 2017)	88.6
ESIM+ELMo	**88.7** $\pm$ 0.17
DIIN ensemble (Gong et al., 2018)	88.9
ESIM+ELMo ensemble	**89.3**

Table 8: SNLI test set accuracy.[3] Single model results occupy the portion, with ensemble results at the bottom.

text vectors are then passed through a linear layer with ReLU activations, a residual self-attention layer that uses a GRU followed by the same attention mechanism applied context-to-context, and another linear layer with ReLU activations. Finally, the results are fed through linear layers to predict the start and end token of the answer.

Variational dropout is used before the input to the GRUs and the linear layers at a rate of 0.2. A dimensionality of 90 is used for the GRUs, and 180 for the linear layers. We optimize the model using Adadelta with a batch size of 45. At test time we use an exponential moving average of the weights and limit the output span to be of at most size 17. We do not update the word vectors during training.

Performance was highest when adding ELMo without layer normalization to both the input and output of the contextual GRU layer and leaving the ELMo weights unregularized ($\lambda = 0$).

Table 9 compares test set results from the SQuAD leaderboard as of November 17, 2017 when we submitted our system. Overall, our submission had the highest single model and ensemble results, improving the previous single model result (SAN) by 1.4% F_1 and our baseline by 4.2%. A 11 member ensemble pushes F_1 to 87.4%, 1.0% increase over the previous ensemble best.

A.5 Semantic Role Labeling

Our baseline SRL model is an exact reimplementation of (He et al., 2017). Words are represented using a concatenation of 100 dimensional vector representations, initialized using GloVe (Pennington et al., 2014) and a binary, per-word predicate feature, represented using an 100 dimensional embedding. This 200 dimensional token representation is then passed through an 8 layer "interleaved" biLSTM with a 300 dimensional hidden size, in which the directions of the LSTM layers alternate per layer. This deep LSTM uses Highway connections (Srivastava et al., 2015) between layers and variational recurrent dropout (Gal and Ghahramani, 2016). This deep representation is then projected using a final dense layer followed by a softmax activation to form a distribution over all possible tags. Labels consist of semantic roles from PropBank (Palmer et al., 2005) augmented with a BIO labeling scheme to represent argument spans. During training, we minimize the negative log likelihood of the tag sequence using Adadelta with a learning rate of 1.0 and $\rho = 0.95$ (Zeiler, 2012). At test time, we perform Viterbi decoding to enforce valid spans using BIO constraints. Variational dropout of 10% is added to all LSTM hidden layers. Gradients are clipped if their value exceeds 1.0. Models are trained for 500 epochs or until validation F1 does not improve for 200 epochs, whichever is sooner. The pretrained GloVe vectors are fine-tuned during training. The final dense layer and all cells of all LSTMs are initialized to be orthogonal. The forget gate bias is initialized to 1 for all LSTMs, with all other gates initialized to 0, as per (Józefowicz et al., 2015).

Table 10 compares test set F1 scores of our ELMo augmented implementation of (He et al., 2017) with previous results. Our single model score of 84.6 F1 represents a new state-of-the-art result on the CONLL 2012 Semantic Role Labeling task, surpassing the previous single model result by 2.9 F1 and a 5-model ensemble by 1.2 F1.

A.6 Coreference resolution

Our baseline coreference model is the end-to-end neural model from Lee et al. (2017) with all hy-

[3] A comprehensive comparison can be found at https://nlp.stanford.edu/projects/snli/

Model	EM	F_1
BiDAF (Seo et al., 2017)	68.0	77.3
BiDAF + Self Attention	72.1	81.1
DCN+	75.1	83.1
Reg-RaSoR	75.8	83.3
FusionNet	76.0	83.9
r-net (Wang et al., 2017)	76.5	84.3
SAN (Liu et al., 2017)	76.8	84.4
BiDAF + Self Attention + ELMo	**78.6**	**85.8**
DCN+ Ensemble	78.9	86.0
FusionNet Ensemble	79.0	86.0
Interactive AoA Reader+ Ensemble	79.1	86.5
BiDAF + Self Attention + ELMo Ensemble	**81.0**	**87.4**

Table 9: Test set results for SQuAD, showing both Exact Match (EM) and F_1. The top half of the table contains single model results with ensembles at the bottom. References provided where available.

Model	F_1
Pradhan et al. (2013)	77.5
Zhou and Xu (2015)	81.3
He et al. (2017), single	81.7
He et al. (2017), ensemble	83.4
He et al. (2017), our impl.	81.4
He et al. (2017) + ELMo	**84.6**

Table 10: SRL CoNLL 2012 test set F_1.

Model	Average F_1
Durrett and Klein (2013)	60.3
Wiseman et al. (2016)	64.2
Clark and Manning (2016)	65.7
Lee et al. (2017) (single)	67.2
Lee et al. (2017) (ensemble)	68.8
Lee et al. (2017) + ELMo	**70.4**

Table 11: Coreference resolution average F_1 on the test set from the CoNLL 2012 shared task.

perparameters exactly following the original implementation.

The best configuration added ELMo to the input of the lowest layer biLSTM and weighted the biLM layers using (1) without any regularization ($\lambda = 0$) or layer normalization. 50% dropout was added to the ELMo representations.

Table 11 compares our results with previously published results. Overall, we improve the single model state-of-the-art by 3.2% average F_1, and our single model result improves the previous ensemble best by 1.6% F_1. Adding ELMo to the output from the biLSTM in addition to the biLSTM input reduced F_1 by approximately 0.7% (not shown).

A.7 Named Entity Recognition

Our baseline NER model concatenates 50 dimensional pre-trained Senna vectors (Collobert et al., 2011) with a CNN character based representation. The character representation uses 16 dimensional character embeddings and 128 convolutional filters of width three characters, a ReLU activation and by max pooling. The token representation is passed through two biLSTM layers, the first with 200 hidden units and the second with 100 hidden units before a final dense layer and softmax layer. During training, we use a CRF loss and at test time perform decoding using the Viterbi algorithm while ensuring that the output tag sequence is valid.

Variational dropout is added to the input of both biLSTM layers. During training the gradients are rescaled if their ℓ^2 norm exceeds 5.0 and parameters updated using Adam with constant learning rate of 0.001. The pre-trained Senna embeddings are fine tuned during training. We employ early stopping on the development set and report the averaged test set score across five runs with different random seeds.

ELMo was added to the input of the lowest layer task biLSTM. As the CoNLL 2003 NER data set is relatively small, we found the best performance by constraining the trainable layer weights to be effectively constant by setting $\lambda = 0.1$ with (1).

Table 12 compares test set F_1 scores of our ELMo enhanced biLSTM-CRF tagger with previous results. Overall, the 92.22% F_1 from our system establishes a new state-of-the-art. When compared to Peters et al. (2017), using representations

Model	$F_1 \pm$ std.
Collobert et al. (2011)♣	89.59
Lample et al. (2016)	90.94
Ma and Hovy (2016)	91.2
Chiu and Nichols (2016)♣,♢	91.62 ± 0.33
Peters et al. (2017)♢	91.93 ± 0.19
biLSTM-CRF + ELMo	**92.22** ± **0.10**

Table 12: Test set F_1 for CoNLL 2003 NER task. Models with ♣ included gazetteers and those with ♢ used both the train and development splits for training.

Model	Acc.
DMN (Kumar et al., 2016)	52.1
LSTM-CNN (Zhou et al., 2016)	52.4
NTI (Munkhdalai and Yu, 2017)	53.1
BCN+Char+CoVe (McCann et al., 2017)	53.7
BCN+ELMo	**54.7**

Table 13: Test set accuracy for SST-5.

from all layers of the biLM provides a modest improvement.

A.8 Sentiment classification

We use almost the same biattention classification network architecture described in McCann et al. (2017), with the exception of replacing the final maxout network with a simpler feedforward network composed of two ReLu layers with dropout. A BCN model with a batch-normalized maxout network reached significantly lower validation accuracies in our experiments, although there may be discrepancies between our implementation and that of McCann et al. (2017). To match the CoVe training setup, we only train on phrases that contain four or more tokens. We use 300-d hidden states for the biLSTM and optimize the model parameters with Adam (Kingma and Ba, 2015) using a learning rate of 0.0001. The trainable biLM layer weights are regularized by $\lambda = 0.001$, and we add ELMo to both the input and output of the biLSTM; the output ELMo vectors are computed with a second biLSTM and concatenated to the input.

An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling

Shaojie Bai [1] **J. Zico Kolter** [2] **Vladlen Koltun** [3]

Abstract

For most deep learning practitioners, sequence modeling is synonymous with recurrent networks. Yet recent results indicate that convolutional architectures can outperform recurrent networks on tasks such as audio synthesis and machine translation. Given a new sequence modeling task or dataset, which architecture should one use? We conduct a systematic evaluation of generic convolutional and recurrent architectures for sequence modeling. The models are evaluated across a broad range of standard tasks that are commonly used to benchmark recurrent networks. Our results indicate that a simple convolutional architecture outperforms canonical recurrent networks such as LSTMs across a diverse range of tasks and datasets, while demonstrating longer effective memory. We conclude that the common association between sequence modeling and recurrent networks should be reconsidered, and convolutional networks should be regarded as a natural starting point for sequence modeling tasks. To assist related work, we have made code available at http://github.com/locuslab/TCN.

1. Introduction

Deep learning practitioners commonly regard recurrent architectures as the default starting point for sequence modeling tasks. The sequence modeling chapter in the canonical textbook on deep learning is titled "Sequence Modeling: Recurrent and Recursive Nets" (Goodfellow et al., 2016), capturing the common association of sequence modeling and recurrent architectures. A well-regarded recent online course on "Sequence Models" focuses exclusively on recurrent architectures (Ng, 2018).

[1]Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA, USA [2]Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA [3]Intel Labs, Santa Clara, CA, USA. Correspondence to: Shaojie Bai <shaojieb@cs.cmu.edu>, J. Zico Kolter <zkolter@cs.cmu.edu>, Vladlen Koltun <vkoltun@gmail.edu>.

On the other hand, recent research indicates that certain convolutional architectures can reach state-of-the-art accuracy in audio synthesis, word-level language modeling, and machine translation (van den Oord et al., 2016; Kalchbrenner et al., 2016; Dauphin et al., 2017; Gehring et al., 2017a;b). This raises the question of whether these successes of convolutional sequence modeling are confined to specific application domains or whether a broader reconsideration of the association between sequence processing and recurrent networks is in order.

We address this question by conducting a systematic empirical evaluation of convolutional and recurrent architectures on a broad range of sequence modeling tasks. We specifically target a comprehensive set of tasks that have been repeatedly used to compare the effectiveness of different recurrent network architectures. These tasks include polyphonic music modeling, word- and character-level language modeling, as well as synthetic stress tests that had been deliberately designed and frequently used to benchmark RNNs. Our evaluation is thus set up to compare convolutional and recurrent approaches to sequence modeling on the recurrent networks' "home turf".

To represent convolutional networks, we describe a generic temporal convolutional network (TCN) architecture that is applied across all tasks. This architecture is informed by recent research, but is deliberately kept simple, combining some of the best practices of modern convolutional architectures. It is compared to canonical recurrent architectures such as LSTMs and GRUs.

The results suggest that TCNs convincingly outperform baseline recurrent architectures across a broad range of sequence modeling tasks. This is particularly notable because the tasks include diverse benchmarks that have commonly been used to evaluate recurrent network designs (Chung et al., 2014; Pascanu et al., 2014; Jozefowicz et al., 2015; Zhang et al., 2016). This indicates that the recent successes of convolutional architectures in applications such as audio processing are not confined to these domains.

To further understand these results, we analyze more deeply the memory retention characteristics of recurrent networks. We show that despite the theoretical ability of recurrent architectures to capture infinitely long history, TCNs exhibit

substantially longer memory, and are thus more suitable for domains where a long history is required.

To our knowledge, the presented study is the most extensive systematic comparison of convolutional and recurrent architectures on sequence modeling tasks. The results suggest that the common association between sequence modeling and recurrent networks should be reconsidered. The TCN architecture appears not only more accurate than canonical recurrent networks such as LSTMs and GRUs, but also simpler and clearer. It may therefore be a more appropriate starting point in the application of deep networks to sequences.

2. Background

Convolutional networks (LeCun et al., 1989) have been applied to sequences for decades (Sejnowski & Rosenberg, 1987; Hinton, 1989). They were used prominently for speech recognition in the 80s and 90s (Waibel et al., 1989; Bottou et al., 1990). ConvNets were subsequently applied to NLP tasks such as part-of-speech tagging and semantic role labelling (Collobert & Weston, 2008; Collobert et al., 2011; dos Santos & Zadrozny, 2014). More recently, convolutional networks were applied to sentence classification (Kalchbrenner et al., 2014; Kim, 2014) and document classification (Zhang et al., 2015; Conneau et al., 2017; Johnson & Zhang, 2015; 2017). Particularly inspiring for our work are the recent applications of convolutional architectures to machine translation (Kalchbrenner et al., 2016; Gehring et al., 2017a;b), audio synthesis (van den Oord et al., 2016), and language modeling (Dauphin et al., 2017).

Recurrent networks are dedicated sequence models that maintain a vector of hidden activations that are propagated through time (Elman, 1990; Werbos, 1990; Graves, 2012). This family of architectures has gained tremendous popularity due to prominent applications to language modeling (Sutskever et al., 2011; Graves, 2013; Hermans & Schrauwen, 2013) and machine translation (Sutskever et al., 2014; Bahdanau et al., 2015). The intuitive appeal of recurrent modeling is that the hidden state can act as a representation of everything that has been seen so far in the sequence. Basic RNN architectures are notoriously difficult to train (Bengio et al., 1994; Pascanu et al., 2013) and more elaborate architectures are commonly used instead, such as the LSTM (Hochreiter & Schmidhuber, 1997) and the GRU (Cho et al., 2014). Many other architectural innovations and training techniques for recurrent networks have been introduced and continue to be actively explored (El Hihi & Bengio, 1995; Schuster & Paliwal, 1997; Gers et al., 2002; Koutnik et al., 2014; Le et al., 2015; Ba et al., 2016; Wu et al., 2016; Krueger et al., 2017; Merity et al., 2017; Campos et al., 2018).

Multiple empirical studies have been conducted to evaluate the effectiveness of different recurrent architectures. These studies have been motivated in part by the many degrees of freedom in the design of such architectures. Chung et al. (2014) compared different types of recurrent units (LSTM vs. GRU) on the task of polyphonic music modeling. Pascanu et al. (2014) explored different ways to construct deep RNNs and evaluated the performance of different architectures on polyphonic music modeling, character-level language modeling, and word-level language modeling. Jozefowicz et al. (2015) searched through more than ten thousand different RNN architectures and evaluated their performance on various tasks. They concluded that if there were "architectures much better than the LSTM", then they were "not trivial to find". Greff et al. (2017) benchmarked the performance of eight LSTM variants on speech recognition, handwriting recognition, and polyphonic music modeling. They also found that "none of the variants can improve upon the standard LSTM architecture significantly". Zhang et al. (2016) systematically analyzed the connecting architectures of RNNs and evaluated different architectures on character-level language modeling and on synthetic stress tests. Melis et al. (2018) benchmarked LSTM-based architectures on word-level and character-level language modeling, and concluded that "LSTMs outperform the more recent models".

Other recent works have aimed to combine aspects of RNN and CNN architectures. This includes the Convolutional LSTM (Shi et al., 2015), which replaces the fully-connected layers in an LSTM with convolutional layers to allow for additional structure in the recurrent layers; the Quasi-RNN model (Bradbury et al., 2017) that interleaves convolutional layers with simple recurrent layers; and the dilated RNN (Chang et al., 2017), which adds dilations to recurrent architectures. While these combinations show promise in combining the desirable aspects of both types of architectures, our study here focuses on a comparison of generic convolutional and recurrent architectures.

While there have been multiple thorough evaluations of RNN architectures on representative sequence modeling tasks, we are not aware of a similarly thorough comparison of convolutional and recurrent approaches to sequence modeling. (Yin et al. (2017) have reported a comparison of convolutional and recurrent networks for sentence-level and document-level classification tasks. In contrast, sequence modeling calls for architectures that can synthesize whole sequences, element by element.) Such comparison is particularly intriguing in light of the aforementioned recent success of convolutional architectures in this domain. Our work aims to compare generic convolutional and recurrent architectures on typical sequence modeling tasks that are commonly used to benchmark RNN variants themselves (Hermans & Schrauwen, 2013; Le et al., 2015; Jozefowicz et al., 2015; Zhang et al., 2016).

3. Temporal Convolutional Networks

We begin by describing a generic architecture for convolutional sequence prediction. Our aim is to distill the best practices in convolutional network design into a simple architecture that can serve as a convenient but powerful starting point. We refer to the presented architecture as a temporal convolutional network (TCN), emphasizing that we adopt this term not as a label for a truly new architecture, but as a simple descriptive term for a family of architectures. (Note that the term has been used before (Lea et al., 2017).) The distinguishing characteristics of TCNs are: 1) the convolutions in the architecture are causal, meaning that there is no information "leakage" from future to past; 2) the architecture can take a sequence of any length and map it to an output sequence of the same length, just as with an RNN. Beyond this, we emphasize how to build very long effective history sizes (i.e., the ability for the networks to look very far into the past to make a prediction) using a combination of very deep networks (augmented with residual layers) and dilated convolutions.

Our architecture is informed by recent convolutional architectures for sequential data (van den Oord et al., 2016; Kalchbrenner et al., 2016; Dauphin et al., 2017; Gehring et al., 2017a;b), but is distinct from all of them and was designed from first principles to combine simplicity, autoregressive prediction, and very long memory. For example, the TCN is much simpler than WaveNet (van den Oord et al., 2016) (no skip connections across layers, conditioning, context stacking, or gated activations).

Compared to the language modeling architecture of Dauphin et al. (2017), TCNs do not use gating mechanisms and have much longer memory.

3.1. Sequence Modeling

Before defining the network structure, we highlight the nature of the sequence modeling task. Suppose that we are given an input sequence $x_0, \ldots, x_T$, and wish to predict some corresponding outputs $y_0, \ldots, y_T$ at each time. The key constraint is that to predict the output y_t for some time t, we are constrained to only use those inputs that have been previously observed: $x_0, \ldots, x_t$. Formally, a sequence modeling network is any function $f : \mathcal{X}^{T+1} \to \mathcal{Y}^{T+1}$ that produces the mapping

$$\hat{y}_0, \ldots, \hat{y}_T = f(x_0, \ldots, x_T) \tag{1}$$

if it satisfies the causal constraint that y_t depends only on $x_0, \ldots, x_t$ and not on any "future" inputs $x_{t+1}, \ldots, x_T$. The goal of learning in the sequence modeling setting is to find a network f that minimizes some expected loss between the actual outputs and the predictions, $L(y_0, \ldots, y_T, f(x_0, \ldots, x_T))$, where the sequences and outputs are drawn according to some distribution.

This formalism encompasses many settings such as autoregressive prediction (where we try to predict some signal given its past) by setting the target output to be simply the input shifted by one time step. It does not, however, directly capture domains such as machine translation, or sequence-to-sequence prediction in general, since in these cases the entire input sequence (including "future" states) can be used to predict each output (though the techniques can naturally be extended to work in such settings).

3.2. Causal Convolutions

As mentioned above, the TCN is based upon two principles: the fact that the network produces an output of the same length as the input, and the fact that there can be no leakage from the future into the past. To accomplish the first point, the TCN uses a 1D fully-convolutional network (FCN) architecture (Long et al., 2015), where each hidden layer is the same length as the input layer, and zero padding of length $(\text{kernel size} - 1)$ is added to keep subsequent layers the same length as previous ones. To achieve the second point, the TCN uses *causal convolutions*, convolutions where an output at time t is convolved only with elements from time t and earlier in the previous layer.

To put it simply: TCN = 1D FCN + causal convolutions.

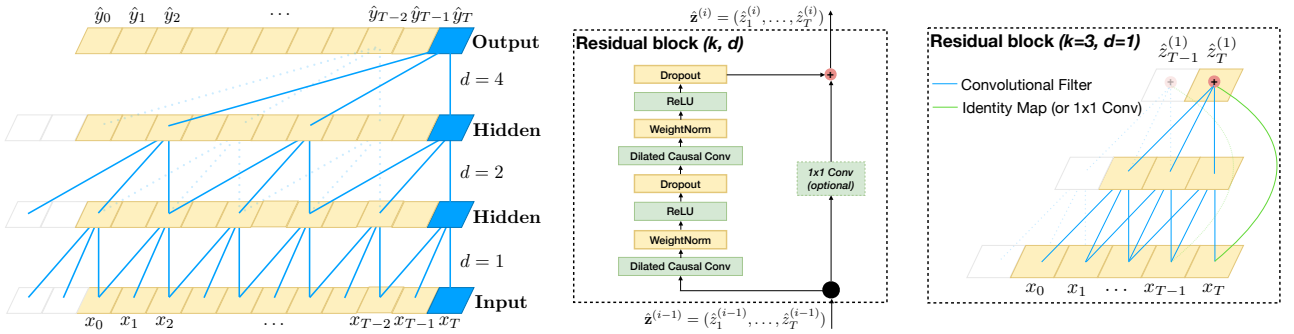
Note that this is essentially the same architecture as the time delay neural network proposed nearly 30 years ago by Waibel et al. (1989), with the sole tweak of zero padding to ensure equal sizes of all layers.

A major disadvantage of this basic design is that in order to achieve a long effective history size, we need an extremely deep network or very large filters, neither of which were particularly feasible when the methods were first introduced. Thus, in the following sections, we describe how techniques from modern convolutional architectures can be integrated into a TCN to allow for both very deep networks and very long effective history.

3.3. Dilated Convolutions

A simple causal convolution is only able to look back at a history with size linear in the depth of the network. This makes it challenging to apply the aforementioned causal convolution on sequence tasks, especially those requiring longer history. Our solution here, following the work of van den Oord et al. (2016), is to employ dilated convolutions that enable an exponentially large receptive field (Yu & Koltun, 2016). More formally, for a 1-D sequence input $\mathbf{x} \in \mathbb{R}^n$ and a filter $f : \{0, \ldots, k-1\} \to \mathbb{R}$, the dilated convolution operation F on element s of the sequence is defined as

$$F(s) = (\mathbf{x} *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot \mathbf{x}_{s-d \cdot i} \tag{2}$$



(a) (b) (c)

Figure 1. Architectural elements in a TCN. (a) A dilated causal convolution with dilation factors $d = 1, 2, 4$ and filter size $k = 3$. The receptive field is able to cover all values from the input sequence. (b) TCN residual block. An 1x1 convolution is added when residual input and output have different dimensions. (c) An example of residual connection in a TCN. The blue lines are filters in the residual function, and the green lines are identity mappings.

where d is the dilation factor, k is the filter size, and $s - d \cdot i$ accounts for the direction of the past. Dilation is thus equivalent to introducing a fixed step between every two adjacent filter taps. When $d = 1$, a dilated convolution reduces to a regular convolution. Using larger dilation enables an output at the top level to represent a wider range of inputs, thus effectively expanding the receptive field of a ConvNet.

This gives us two ways to increase the receptive field of the TCN: choosing larger filter sizes k and increasing the dilation factor d, where the effective history of one such layer is $(k-1)d$. As is common when using dilated convolutions, we increase d exponentially with the depth of the network (i.e., $d = O(2^i)$ at level i of the network). This ensures that there is some filter that hits each input within the effective history, while also allowing for an extremely large effective history using deep networks. We provide an illustration in Figure 1(a).

3.4. Residual Connections

A residual block (He et al., 2016) contains a branch leading out to a series of transformations $\mathcal{F}$, whose outputs are added to the input $\mathbf{x}$ of the block:

$$o = \text{Activation}(\mathbf{x} + \mathcal{F}(\mathbf{x})) \tag{3}$$

This effectively allows layers to learn modifications to the identity mapping rather than the entire transformation, which has repeatedly been shown to benefit very deep networks.

Since a TCN's receptive field depends on the network depth n as well as filter size k and dilation factor d, stabilization of deeper and larger TCNs becomes important. For example, in a case where the prediction could depend on a history of size 2^{12} and a high-dimensional input sequence, a network of up to 12 layers could be needed. Each layer, more specifically, consists of multiple filters for feature extraction. In our design of the generic TCN model, we therefore employ a generic residual module in place of a convolutional layer.

The residual block for our baseline TCN is shown in Figure 1(b). Within a residual block, the TCN has two layers of dilated causal convolution and non-linearity, for which we used the rectified linear unit (ReLU) (Nair & Hinton, 2010). For normalization, we applied weight normalization (Salimans & Kingma, 2016) to the convolutional filters. In addition, a spatial dropout (Srivastava et al., 2014) was added after each dilated convolution for regularization: at each training step, a whole channel is zeroed out.

However, whereas in standard ResNet the input is added directly to the output of the residual function, in TCN (and ConvNets in general) the input and output could have different widths. To account for discrepant input-output widths, we use an additional 1x1 convolution to ensure that element-wise addition $\oplus$ receives tensors of the same shape (see Figure 1(b,c)).

3.5. Discussion

We conclude this section by listing several advantages and disadvantages of using TCNs for sequence modeling.

- **Parallelism**. Unlike in RNNs where the predictions for later timesteps must wait for their predecessors to complete, convolutions can be done in parallel since the same filter is used in each layer. Therefore, in both training and evaluation, a long input sequence can be processed as a whole in TCN, instead of sequentially as in RNN.
- **Flexible receptive field size**. A TCN can change its receptive field size in multiple ways. For instance, stacking more dilated (causal) convolutional layers, using larger dilation factors, or increasing the filter size are all viable options (with possibly different interpretations). TCNs thus afford better control of the model's memory size, and are easy to adapt to different domains.
- **Stable gradients**. Unlike recurrent architectures, TCN has a backpropagation path different from the temporal direction of the sequence. TCN thus avoids the problem

of exploding/vanishing gradients, which is a major issue for RNNs (and which led to the development of LSTM, GRU, HF-RNN (Martens & Sutskever, 2011), etc.).

- **Low memory requirement for training**. Especially in the case of a long input sequence, LSTMs and GRUs can easily use up a lot of memory to store the partial results for their multiple cell gates. However, in a TCN the filters are shared across a layer, with the backpropagation path depending only on network depth. Therefore in practice, we found gated RNNs likely to use up to a multiplicative factor more memory than TCNs.
- **Variable length inputs**. Just like RNNs, which model inputs with variable lengths in a recurrent way, TCNs can also take in inputs of arbitrary lengths by sliding the 1D convolutional kernels. This means that TCNs can be adopted as drop-in replacements for RNNs for sequential data of arbitrary length.

There are also two notable disadvantages to using TCNs.

- **Data storage during evaluation**. In evaluation/testing, RNNs only need to maintain a hidden state and take in a current input x_t in order to generate a prediction. In other words, a "summary" of the entire history is provided by the fixed-length set of vectors h_t, and the actual observed sequence can be discarded. In contrast, TCNs need to take in the raw sequence up to the effective history length, thus possibly requiring more memory during evaluation.
- **Potential parameter change for a transfer of domain**. Different domains can have different requirements on the amount of history the model needs in order to predict. Therefore, when transferring a model from a domain where only little memory is needed (i.e., small k and d) to a domain where much longer memory is required (i.e., much larger k and d), TCN may perform poorly for not having a sufficiently large receptive field.

4. Sequence Modeling Tasks

We evaluate TCNs and RNNs on tasks that have been commonly used to benchmark the performance of different RNN sequence modeling architectures (Hermans & Schrauwen, 2013; Chung et al., 2014; Pascanu et al., 2014; Le et al., 2015; Jozefowicz et al., 2015; Zhang et al., 2016). The intention is to conduct the evaluation on the "home turf" of RNN sequence models. We use a comprehensive set of synthetic stress tests along with real-world datasets from multiple domains.

The adding problem. In this task, each input consists of a length-n sequence of depth 2, with all values randomly chosen in $[0, 1]$, and the second dimension being all zeros except for two elements that are marked by 1. The objective is to sum the two random values whose second dimensions are marked by 1. Simply predicting the sum to be 1 should give an MSE of about 0.1767. First introduced by Hochreiter & Schmidhuber (1997), the adding problem has been used repeatedly as a stress test for sequence models (Martens & Sutskever, 2011; Pascanu et al., 2013; Le et al., 2015; Arjovsky et al., 2016; Zhang et al., 2016).

Sequential MNIST and P-MNIST. Sequential MNIST is frequently used to test a recurrent network's ability to retain information from the distant past (Le et al., 2015; Zhang et al., 2016; Wisdom et al., 2016; Cooijmans et al., 2016; Krueger et al., 2017; Jing et al., 2017). In this task, MNIST images (LeCun et al., 1998) are presented to the model as a 784×1 sequence for digit classification. In the more challenging P-MNIST setting, the order of the sequence is permuted at random (Le et al., 2015; Arjovsky et al., 2016; Wisdom et al., 2016; Krueger et al., 2017).

Copy memory. In this task, each input sequence has length $T + 20$. The first 10 values are chosen randomly among the digits $1, \ldots, 8$, with the rest being all zeros, except for the last 11 entries that are filled with the digit '9' (the first '9' is a delimiter). The goal is to generate an output of the same length that is zero everywhere except the last 10 values after the delimiter, where the model is expected to repeat the 10 values it encountered at the start of the input. This task was used in prior works such as Zhang et al. (2016); Arjovsky et al. (2016); Wisdom et al. (2016); Jing et al. (2017).

JSB Chorales and Nottingham. JSB Chorales (Allan & Williams, 2005) is a polyphonic music dataset consisting of the entire corpus of 382 four-part harmonized chorales by J. S. Bach. Each input is a sequence of elements. Each element is an 88-bit binary code that corresponds to the 88 keys on a piano, with 1 indicating a key that is pressed at a given time. Nottingham is a polyphonic music dataset based on a collection of 1,200 British and American folk tunes, and is much larger than JSB Chorales. JSB Chorales and Nottingham have been used in numerous empirical investigations of recurrent sequence modeling (Chung et al., 2014; Pascanu et al., 2014; Jozefowicz et al., 2015; Greff et al., 2017). The performance on both tasks is measured in terms of negative log-likelihood (NLL).

PennTreebank. We used the PennTreebank (PTB) (Marcus et al., 1993) for both character-level and word-level language modeling. When used as a character-level language corpus, PTB contains 5,059K characters for training, 396K for validation, and 446K for testing, with an alphabet size of 50. When used as a word-level language corpus, PTB contains 888K words for training, 70K for validation, and 79K for testing, with a vocabulary size of 10K. This is a highly studied but relatively small language modeling dataset (Miyamoto & Cho, 2016; Krueger et al., 2017; Merity et al., 2017).

Wikitext-103. Wikitext-103 (Merity et al., 2016) is almost

Table 1. Evaluation of TCNs and recurrent architectures on synthetic stress tests, polyphonic music modeling, character-level language modeling, and word-level language modeling. The generic TCN architecture outperforms canonical recurrent networks across a comprehensive suite of tasks and datasets. Current state-of-the-art results are listed in the supplement. ^{h} means that higher is better. $^\ell$ means that lower is better.

Sequence Modeling Task	Model Size ($\approx$)	Models			
		LSTM	GRU	RNN	**TCN**
Seq. MNIST (accuracy^h)	70K	87.2	96.2	21.5	**99.0**
Permuted MNIST (accuracy)	70K	85.7	87.3	25.3	**97.2**
Adding problem T=600 (loss$^\ell$)	70K	0.164	**5.3e-5**	0.177	**5.8e-5**
Copy memory T=1000 (loss)	16K	0.0204	0.0197	0.0202	**3.5e-5**
Music JSB Chorales (loss)	300K	8.45	8.43	8.91	**8.10**
Music Nottingham (loss)	1M	3.29	3.46	4.05	**3.07**
Word-level PTB (perplexity$^\ell$)	13M	**78.93**	92.48	114.50	88.68
Word-level Wiki-103 (perplexity)	-	48.4	-	-	**45.19**
Word-level LAMBADA (perplexity)	-	4186	-	14725	**1279**
Char-level PTB (bpc$^\ell$)	3M	1.36	1.37	1.48	**1.31**
Char-level text8 (bpc)	5M	1.50	1.53	1.69	**1.45**

110 times as large as PTB, featuring a vocabulary size of about 268K. The dataset contains 28K Wikipedia articles (about 103 million words) for training, 60 articles (about 218K words) for validation, and 60 articles (246K words) for testing. This is a more representative and realistic dataset than PTB, with a much larger vocabulary that includes many rare words, and has been used in Merity et al. (2016); Grave et al. (2017); Dauphin et al. (2017).

LAMBADA. Introduced by Paperno et al. (2016), LAMBADA is a dataset comprising 10K passages extracted from novels, with an average of 4.6 sentences as context, and 1 target sentence the last word of which is to be predicted. This dataset was built so that a person can easily guess the missing word when given the context sentences, but not when given only the target sentence without the context sentences. Most of the existing models fail on LAMBADA (Paperno et al., 2016; Grave et al., 2017). In general, better results on LAMBADA indicate that a model is better at capturing information from longer and broader context. The training data for LAMBADA is the full text of 2,662 novels with more than 200M words. The vocabulary size is about 93K.

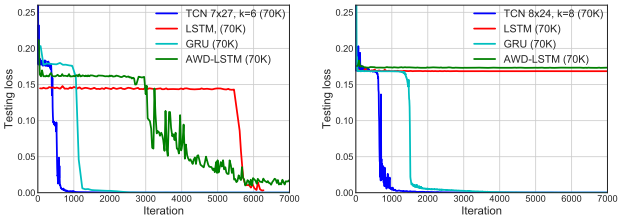
text8. We also used the text8 dataset for character-level language modeling (Mikolov et al., 2012). text8 is about 20 times larger than PTB, with about 100M characters from Wikipedia (90M for training, 5M for validation, and 5M for testing). The corpus contains 27 unique alphabets.

5. Experiments

We compare the generic TCN architecture described in Section 3 to canonical recurrent architectures, namely LSTM, GRU, and vanilla RNN, with standard regularizations. All experiments reported in this section used exactly the same TCN architecture, just varying the depth of the network n and occasionally the kernel size k so that the receptive field covers enough context for predictions. We use an exponential dilation $d = 2^i$ for layer i in the network, and the Adam optimizer (Kingma & Ba, 2015) with learning rate 0.002 for TCN, unless otherwise noted. We also empirically find that gradient clipping helped convergence, and we pick the maximum norm for clipping from $[0.3, 1]$. When training recurrent models, we use grid search to find a good set of hyperparameters (in particular, optimizer, recurrent drop $p \in [0.05, 0.5]$, learning rate, gradient clipping, and initial forget-gate bias), while keeping the network around the same size as TCN. No other architectural elaborations, such as gating mechanisms or skip connections, were added to either TCNs or RNNs. Additional details and controlled experiments are provided in the supplementary material.

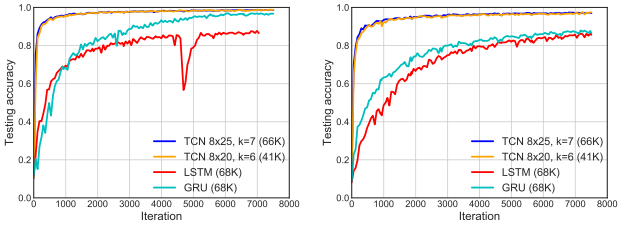
5.1. Synopsis of Results

A synopsis of the results is shown in Table 1. Note that on several of these tasks, the generic, canonical recurrent architectures we study (e.g., LSTM, GRU) are not the state-of-the-art. (See the supplement for more details.) With this caveat, the results strongly suggest that the generic TCN architecture *with minimal tuning* outperforms canonical recurrent architectures across a broad variety of sequence modeling tasks that are commonly used to benchmark the performance of recurrent architectures themselves. We now analyze these results in more detail.



(a) $T = 200$ (b) $T = 600$

Figure 2. Results on the adding problem for different sequence lengths T. TCNs outperform recurrent architectures.



(a) Sequential MNIST (b) P-MNIST

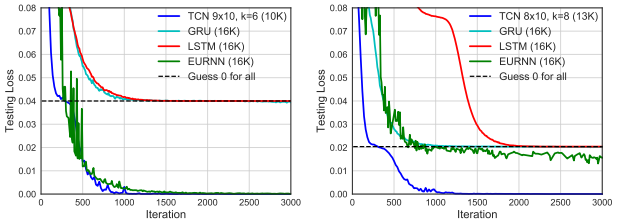
Figure 3. Results on Sequential MNIST and P-MNIST. TCNs outperform recurrent architectures.

5.2. Synthetic Stress Tests

The adding problem. Convergence results for the adding problem, for problem sizes $T = 200$ and 600, are shown in Figure 2. All models were chosen to have roughly 70K parameters. TCNs quickly converged to a virtually perfect solution (i.e., MSE near 0). GRUs also performed quite well, albeit slower to converge than TCNs. LSTMs and vanilla RNNs performed significantly worse.

Sequential MNIST and P-MNIST. Convergence results on sequential and permuted MNIST, run over 10 epochs, are shown in Figure 3. All models were configured to have roughly 70K parameters. For both problems, TCNs substantially outperform the recurrent architectures, both in terms of convergence and in final accuracy on the task. For P-MNIST, TCNs outperform state-of-the-art results (95.9%) based on recurrent networks with Zoneout and Recurrent BatchNorm (Cooijmans et al., 2016; Krueger et al., 2017).

Copy memory. Convergence results on the copy memory task are shown in Figure 4. TCNs quickly converge to correct answers, while LSTMs and GRUs simply converge to the same loss as predicting all zeros. In this case we also compare to the recently-proposed EURNN (Jing et al., 2017), which was highlighted to perform well on this task. While both TCN and EURNN perform well for sequence length $T = 500$, the TCN has a clear advantage for $T = 1000$ and longer (in terms of both loss and rate of convergence).



(a) $T = 500$ (b) $T = 1000$

Figure 4. Result on the copy memory task for different sequence lengths T. TCNs outperform recurrent architectures.

5.3. Polyphonic Music and Language Modeling

We now discuss the results on polyphonic music modeling, character-level language modeling, and word-level language modeling. These domains are dominated by recurrent architectures, with many specialized designs developed for these tasks (Zhang et al., 2016; Ha et al., 2017; Krueger et al., 2017; Grave et al., 2017; Greff et al., 2017; Merity et al., 2017). We mention some of these specialized architectures when useful, but our primary goal is to compare the generic TCN model to similarly generic recurrent architectures, before domain-specific tuning. The results are summarized in Table 1.

Polyphonic music. On Nottingham and JSB Chorales, the TCN with virtually no tuning outperforms the recurrent models by a considerable margin, and even outperforms some enhanced recurrent architectures for this task such as HF-RNN (Boulanger-Lewandowski et al., 2012) and Diagonal RNN (Subakan & Smaragdis, 2017). Note however that other models such as the Deep Belief Net LSTM perform better still (Vohra et al., 2015); we believe this is likely due to the fact that the datasets are relatively small, and thus the right regularization method or generative modeling procedure can improve performance significantly. This is largely orthogonal to the RNN/TCN distinction, as a similar variant of TCN may well be possible.

Word-level language modeling. Language modeling remains one of the primary applications of recurrent networks and many recent works have focused on optimizing LSTMs for this task (Krueger et al., 2017; Merity et al., 2017). Our implementation follows standard practice that ties the weights of encoder and decoder layers for both TCN and RNNs (Press & Wolf, 2016), which significantly reduces the number of parameters in the model. For training, we use SGD and anneal the learning rate by a factor of 0.5 for both TCN and RNNs when validation accuracy plateaus.

On the smaller PTB corpus, an optimized LSTM architecture (with recurrent and embedding dropout, etc.) outperforms the TCN, while the TCN outperforms both GRU and vanilla RNN. However, on the much larger Wikitext-103 corpus and the LAMBADA dataset (Paperno et al., 2016), without any hyperparameter search, the TCN outperforms

the LSTM results of Grave et al. (2017), achieving much lower perplexities.

Character-level language modeling. On character-level language modeling (PTB and text8, accuracy measured in bits per character), the generic TCN outperforms regularized LSTMs and GRUs as well as methods such as Norm-stabilized LSTMs (Krueger & Memisevic, 2015). (Specialized architectures exist that outperform all of these, see the supplement.)

5.4. Memory Size of TCN and RNNs

One of the theoretical advantages of recurrent architectures is their unlimited memory: the theoretical ability to retain information through sequences of unlimited length. We now examine specifically how long the different architectures can retain information in practice. We focus on 1) the copy memory task, which is a stress test designed to evaluate long-term, distant information propagation in recurrent networks, and 2) the LAMBADA task, which tests both local and non-local textual understanding.

The copy memory task is perfectly set up to examine a model's ability to retain information for different lengths of time. The requisite retention time can be controlled by varying the sequence length T. In contrast to Section 5.2, we now focus on the accuracy on the last 10 elements of the output sequence (which are the nontrivial elements that must be recalled). We used models of size 10K for both TCN and RNNs.

The results of this focused study are shown in Figure 5. TCNs consistently converge to 100% accuracy for all sequence lengths, whereas LSTMs and GRUs of the same size quickly degenerate to random guessing as the sequence length T grows. The accuracy of the LSTM falls below 20% for $T < 50$, while the GRU falls below 20% for $T < 200$. These results indicate that TCNs are able to maintain a much longer effective history than their recurrent counterparts.

This observation is backed up on real data by experiments on the large-scale LAMBADA dataset, which is specifically designed to test a model's ability to utilize broad context (Paperno et al., 2016). As shown in Table 1, TCN outperforms LSTMs and vanilla RNNs by a significant margin in perplexity on LAMBADA, with a substantially smaller network and virtually no tuning. (State-of-the-art results on this dataset are even better, but only with the help of additional memory mechanisms (Grave et al., 2017).)

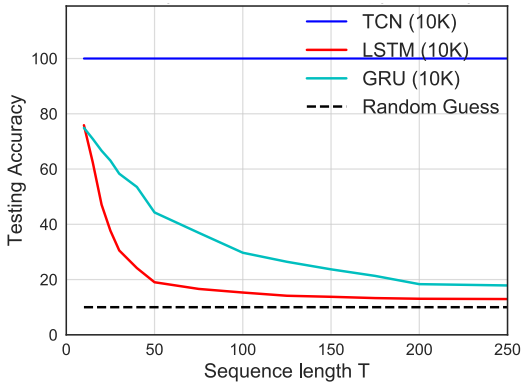


Figure 5. Accuracy on the copy memory task for sequences of different lengths T. While TCN exhibits 100% accuracy for all sequence lengths, the LSTM and GRU degenerate to random guessing as T grows.

6. Conclusion

We have presented an empirical evaluation of generic convolutional and recurrent architectures across a comprehensive suite of sequence modeling tasks. To this end, we have described a simple temporal convolutional network (TCN) that combines best practices such as dilations and residual connections with the causal convolutions needed for autoregressive prediction. The experimental results indicate that TCN models substantially outperform generic recurrent architectures such as LSTMs and GRUs. We further studied long-range information propagation in convolutional and recurrent networks, and showed that the "infinite memory" advantage of RNNs is largely absent in practice. TCNs exhibit longer memory than recurrent architectures with the same capacity.

Numerous advanced schemes for regularizing and optimizing LSTMs have been proposed (Press & Wolf, 2016; Krueger et al., 2017; Merity et al., 2017; Campos et al., 2018). These schemes have significantly advanced the accuracy achieved by LSTM-based architectures on some datasets. The TCN has not yet benefitted from this concerted community-wide investment into architectural and algorithmic elaborations. We see such investment as desirable and expect it to yield advances in TCN performance that are commensurate with the advances seen in recent years in LSTM performance. We will release the code for our project to encourage this exploration.

The preeminence enjoyed by recurrent networks in sequence modeling may be largely a vestige of history. Until recently, before the introduction of architectural elements such as dilated convolutions and residual connections, convolutional architectures were indeed weaker. Our results indicate that with these elements, a simple convolutional architecture is more effective across diverse sequence modeling tasks than recurrent architectures such as LSTMs. Due to the comparable clarity and simplicity of TCNs, we conclude that convolutional networks should be regarded as a natural starting point and a powerful toolkit for sequence modeling.

References

Allan, Moray and Williams, Christopher. Harmonising chorales by probabilistic inference. In *NIPS*, 2005.

Arjovsky, Martin, Shah, Amar, and Bengio, Yoshua. Unitary evolution recurrent neural networks. In *ICML*, 2016.

Ba, Lei Jimmy, Kiros, Ryan, and Hinton, Geoffrey E. Layer normalization. *arXiv:1607.06450*, 2016.

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

Bengio, Yoshua, Simard, Patrice, and Frasconi, Paolo. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2), 1994.

Bottou, Léon, Soulie, F Fogelman, Blanchet, Pascal, and Liénard, Jean-Sylvain. Speaker-independent isolated digit recognition: Multilayer perceptrons vs. dynamic time warping. *Neural Networks*, 3(4), 1990.

Boulanger-Lewandowski, Nicolas, Bengio, Yoshua, and Vincent, Pascal. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv:1206.6392*, 2012.

Bradbury, James, Merity, Stephen, Xiong, Caiming, and Socher, Richard. Quasi-recurrent neural networks. In *ICLR*, 2017.

Campos, Victor, Jou, Brendan, Giró i Nieto, Xavier, Torres, Jordi, and Chang, Shih-Fu. Skip RNN: Learning to skip state updates in recurrent neural networks. In *ICLR*, 2018.

Chang, Shiyu, Zhang, Yang, Han, Wei, Yu, Mo, Guo, Xiaoxiao, Tan, Wei, Cui, Xiaodong, Witbrock, Michael J., Hasegawa-Johnson, Mark A., and Huang, Thomas S. Dilated recurrent neural networks. In *NIPS*, 2017.

Cho, Kyunghyun, Van Merriënboer, Bart, Bahdanau, Dzmitry, and Bengio, Yoshua. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv:1409.1259*, 2014.

Chung, Junyoung, Gulcehre, Caglar, Cho, KyungHyun, and Bengio, Yoshua. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555*, 2014.

Chung, Junyoung, Ahn, Sungjin, and Bengio, Yoshua. Hierarchical multiscale recurrent neural networks. *arXiv:1609.01704*, 2016.

Collobert, Ronan and Weston, Jason. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, 2008.

Collobert, Ronan, Weston, Jason, Bottou, Léon, Karlen, Michael, Kavukcuoglu, Koray, and Kuksa, Pavel P. Natural language processing (almost) from scratch. *JMLR*, 12, 2011.

Conneau, Alexis, Schwenk, Holger, LeCun, Yann, and Barrault, Loïc. Very deep convolutional networks for text classification. In *European Chapter of the Association for Computational Linguistics (EACL)*, 2017.

Cooijmans, Tim, Ballas, Nicolas, Laurent, César, Gülçehre, Çağlar, and Courville, Aaron. Recurrent batch normalization. In *ICLR*, 2016.

Dauphin, Yann N., Fan, Angela, Auli, Michael, and Grangier, David. Language modeling with gated convolutional networks. In *ICML*, 2017.

dos Santos, Cícero Nogueira and Zadrozny, Bianca. Learning character-level representations for part-of-speech tagging. In *ICML*, 2014.

El Hihi, Salah and Bengio, Yoshua. Hierarchical recurrent neural networks for long-term dependencies. In *NIPS*, 1995.

Elman, Jeffrey L. Finding structure in time. *Cognitive Science*, 14 (2), 1990.

Gehring, Jonas, Auli, Michael, Grangier, David, and Dauphin, Yann. A convolutional encoder model for neural machine translation. In *ACL*, 2017a.

Gehring, Jonas, Auli, Michael, Grangier, David, Yarats, Denis, and Dauphin, Yann N. Convolutional sequence to sequence learning. In *ICML*, 2017b.

Gers, Felix A, Schraudolph, Nicol N, and Schmidhuber, Jürgen. Learning precise timing with lstm recurrent networks. *JMLR*, 3, 2002.

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. MIT Press, 2016.

Grave, Edouard, Joulin, Armand, and Usunier, Nicolas. Improving neural language models with a continuous cache. In *ICLR*, 2017.

Graves, Alex. *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2012.

Graves, Alex. Generating sequences with recurrent neural networks. *arXiv:1308.0850*, 2013.

Greff, Klaus, Srivastava, Rupesh Kumar, Koutník, Jan, Steunebrink, Bas R., and Schmidhuber, Jürgen. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10), 2017.

Ha, David, Dai, Andrew, and Le, Quoc V. HyperNetworks. In *ICLR*, 2017.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *CVPR*, 2016.

Hermans, Michiel and Schrauwen, Benjamin. Training and analysing deep recurrent neural networks. In *NIPS*, 2013.

Hinton, Geoffrey E. Connectionist learning procedures. *Artificial Intelligence*, 40(1-3), 1989.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural Computation*, 9(8), 1997.

Jing, Li, Shen, Yichen, Dubcek, Tena, Peurifoy, John, Skirlo, Scott, LeCun, Yann, Tegmark, Max, and Soljačić, Marin. Tunable efficient unitary neural networks (EUNN) and their application to RNNs. In *ICML*, 2017.

Johnson, Rie and Zhang, Tong. Effective use of word order for text categorization with convolutional neural networks. In *HLT-NAACL*, 2015.

Johnson, Rie and Zhang, Tong. Deep pyramid convolutional neural networks for text categorization. In *ACL*, 2017.

Jozefowicz, Rafal, Zaremba, Wojciech, and Sutskever, Ilya. An empirical exploration of recurrent network architectures. In *ICML*, 2015.

Kalchbrenner, Nal, Grefenstette, Edward, and Blunsom, Phil. A convolutional neural network for modelling sentences. In *ACL*, 2014.

Kalchbrenner, Nal, Espeholt, Lasse, Simonyan, Karen, van den Oord, Aäron, Graves, Alex, and Kavukcuoglu, Koray. Neural machine translation in linear time. *arXiv:1610.10099*, 2016.

Kim, Yoon. Convolutional neural networks for sentence classification. In *EMNLP*, 2014.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Koutnik, Jan, Greff, Klaus, Gomez, Faustino, and Schmidhuber, Juergen. A clockwork RNN. In *ICML*, 2014.

Krueger, David and Memisevic, Roland. Regularizing RNNs by stabilizing activations. *arXiv:1511.08400*, 2015.

Krueger, David, Maharaj, Tegan, Kramár, János, Pezeshki, Mohammad, Ballas, Nicolas, Ke, Nan Rosemary, Goyal, Anirudh, Bengio, Yoshua, Larochelle, Hugo, Courville, Aaron C., and Pal, Chris. Zoneout: Regularizing RNNs by randomly preserving hidden activations. In *ICLR*, 2017.

Le, Quoc V, Jaitly, Navdeep, and Hinton, Geoffrey E. A simple way to initialize recurrent networks of rectified linear units. *arXiv:1504.00941*, 2015.

Lea, Colin, Flynn, Michael D., Vidal, René, Reiter, Austin, and Hager, Gregory D. Temporal convolutional networks for action segmentation and detection. In *CVPR*, 2017.

LeCun, Yann, Boser, Bernhard, Denker, John S., Henderson, Donnie, Howard, Richard E., Hubbard, Wayne, and Jackel, Lawrence D. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4), 1989.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998.

Long, Jonathan, Shelhamer, Evan, and Darrell, Trevor. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

Marcus, Mitchell P, Marcinkiewicz, Mary Ann, and Santorini, Beatrice. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2), 1993.

Martens, James and Sutskever, Ilya. Learning recurrent neural networks with Hessian-free optimization. In *ICML*, 2011.

Melis, Gábor, Dyer, Chris, and Blunsom, Phil. On the state of the art of evaluation in neural language models. In *ICLR*, 2018.

Merity, Stephen, Xiong, Caiming, Bradbury, James, and Socher, Richard. Pointer sentinel mixture models. *arXiv:1609.07843*, 2016.

Merity, Stephen, Keskar, Nitish Shirish, and Socher, Richard. Regularizing and optimizing LSTM language models. *arXiv:1708.02182*, 2017.

Mikolov, Tomáš, Sutskever, Ilya, Deoras, Anoop, Le, Hai-Son, Kombrink, Stefan, and Cernocky, Jan. Subword language modeling with neural networks. *Preprint*, 2012.

Miyamoto, Yasumasa and Cho, Kyunghyun. Gated word-character recurrent language model. *arXiv:1606.01700*, 2016.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted Boltzmann machines. In *ICML*, 2010.

Ng, Andrew. Sequence Models (Course 5 of Deep Learning Specialization). *Coursera*, 2018.

Paperno, Denis, Kruszewski, Germán, Lazaridou, Angeliki, Pham, Quan Ngoc, Bernardi, Raffaella, Pezzelle, Sandro, Baroni, Marco, Boleda, Gemma, and Fernández, Raquel. The LAMBADA dataset: Word prediction requiring a broad discourse context. *arXiv:1606.06031*, 2016.

Pascanu, Razvan, Mikolov, Tomas, and Bengio, Yoshua. On the difficulty of training recurrent neural networks. In *ICML*, 2013.

Pascanu, Razvan, Gülçehre, Çaglar, Cho, Kyunghyun, and Bengio, Yoshua. How to construct deep recurrent neural networks. In *ICLR*, 2014.

Press, Ofir and Wolf, Lior. Using the output embedding to improve language models. *arXiv:1608.05859*, 2016.

Salimans, Tim and Kingma, Diederik P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NIPS*, 2016.

Schuster, Mike and Paliwal, Kuldip K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45 (11), 1997.

Sejnowski, Terrence J. and Rosenberg, Charles R. Parallel networks that learn to pronounce English text. *Complex Systems*, 1, 1987.

Shi, Xingjian, Chen, Zhourong, Wang, Hao, Yeung, Dit-Yan, Wong, Wai-Kin, and Woo, Wang-chun. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *NIPS*, 2015.

Srivastava, Nitish, Hinton, Geoffrey E, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1), 2014.

Subakan, Y Cem and Smaragdis, Paris. Diagonal RNNs in symbolic music modeling. *arXiv:1704.05420*, 2017.

Sutskever, Ilya, Martens, James, and Hinton, Geoffrey E. Generating text with recurrent neural networks. In *ICML*, 2011.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

van den Oord, Aäron, Dieleman, Sander, Zen, Heiga, Simonyan, Karen, Vinyals, Oriol, Graves, Alex, Kalchbrenner, Nal, Senior, Andrew W., and Kavukcuoglu, Koray. WaveNet: A generative model for raw audio. *arXiv:1609.03499*, 2016.

Vohra, Raunaq, Goel, Kratarth, and Sahoo, JK. Modeling temporal dependencies in data using a DBN-LSTM. In *Data Science and Advanced Analytics (DSAA)*, 2015.

Waibel, Alex, Hanazawa, Toshiyuki, Hinton, Geoffrey, Shikano, Kiyohiro, and Lang, Kevin J. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3), 1989.

Werbos, Paul J. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78(10), 1990.

Wisdom, Scott, Powers, Thomas, Hershey, John, Le Roux, Jonathan, and Atlas, Les. Full-capacity unitary recurrent neural networks. In *NIPS*, 2016.

Wu, Yuhuai, Zhang, Saizheng, Zhang, Ying, Bengio, Yoshua, and Salakhutdinov, Ruslan R. On multiplicative integration with recurrent neural networks. In *NIPS*, 2016.

Yang, Zhilin, Dai, Zihang, Salakhutdinov, Ruslan, and Cohen, William W. Breaking the softmax bottleneck: A high-rank RNN language model. *ICLR*, 2018.

Yin, Wenpeng, Kann, Katharina, Yu, Mo, and Schütze, Hinrich. Comparative study of CNN and RNN for natural language processing. *arXiv:1702.01923*, 2017.

Yu, Fisher and Koltun, Vladlen. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

Zhang, Saizheng, Wu, Yuhuai, Che, Tong, Lin, Zhouhan, Memisevic, Roland, Salakhutdinov, Ruslan R, and Bengio, Yoshua. Architectural complexity measures of recurrent neural networks. In *NIPS*, 2016.

Zhang, Xiang, Zhao, Junbo Jake, and LeCun, Yann. Character-level convolutional networks for text classification. In *NIPS*, 2015.

An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling

Supplementary Material

A. Hyperparameters Settings

A.1. Hyperparameters for TCN

Table 2 lists the hyperparameters we used when applying the generic TCN model on various tasks and datasets. The most important factor for picking parameters is to make sure that the TCN has a sufficiently large receptive field by choosing k and d that can cover the amount of context needed for the task.

As discussed in Section 5, the number of hidden units was chosen so that the model size is approximately at the same level as the recurrent models with which we are comparing. In Table 2, a gradient clip of N/A means no gradient clipping was applied. In larger tasks (e.g., language modeling), we empirically found that gradient clipping (we randomly picked a threshold from $[0.3, 1]$) helps with regularizing TCN and accelerating convergence.

All weights were initialized from a Gaussian disitribution $\mathcal{N}(0, 0.01)$. In general, we found TCN to be relatively insensitive to hyperparameter changes, as long as the effective history (i.e., receptive field) size is sufficient.

A.2. Hyperparameters for LSTM/GRU

Table 3 reports hyperparameter settings that were used for the LSTM. These values are picked from hyperparameter search for LSTMs that have up to 3 layers, and the optimizers are chosen from {SGD, Adam, RMSprop, Adagrad}. For certain larger datasets, we adopted the settings used in prior work (e.g., Grave et al. (2017) on Wikitext-103). GRU hyperparameters were chosen in a similar fashion, but typically with more hidden units than in LSTM to keep the total network size approximately the same (since a GRU cell is more compact).

B. State-of-the-Art Results

As previously noted, the generic TCN and LSTM/GRU models we used can be outperformed by more specialized architectures on some tasks. State-of-the-art results are summarized in Table 4. The same TCN architecture is used across all tasks. Note that the size of the state-of-the-art model may be different from the size of the TCN.

C. Effect of Filter Size and Residual Block

In this section we briefly study the effects of different components of a TCN layer. Overall, we believe dilation is required for modeling long-term dependencies, and so we mainly focus on two other factors here: the filter size k used by each layer, and the effect of residual blocks.

We perform a series of controlled experiments, with the results of the ablative analysis shown in Figure 6. As before, we kept the model size and depth exactly the same for different models, so that the dilation factor is strictly controlled. The experiments were conducted on three different tasks: copy memory, permuted MNIST (P-MNIST), and Penn Treebank word-level language modeling. These experiments confirm that both factors (filter size and residual connections) contribute to sequence modeling performance.

Filter size k. In both the copy memory and the P-MNIST tasks, we observed faster convergence and better accuracy for larger filter sizes. In particular, looking at Figure 6a, a TCN with filter size ≤ 3 only converges to the same level as random guessing. In contrast, on word-level language modeling, a smaller kernel with filter size of $k = 3$ works best. We believe this is because a smaller kernel (along with fixed dilation) tends to focus more on the local context, which is especially important for PTB language modeling (in fact, the very success of n-gram models suggests that only a relatively short memory is needed for modeling language).

Residual block. In all three scenarios that we compared here, we observed that the residual function stabilized training and brought faster convergence with better final results. Especially in language modeling, we found that residual connections contribute substantially to performance (See Figure 6f).

D. Gating Mechanisms

One component that had been used in prior work on convolutional architectures for language modeling is the gated activation (van den Oord et al., 2016; Dauphin et al., 2017). We have chosen not to use gating in the generic TCN model. We now examine this choice more closely.

Table 2. TCN parameter settings for experiments in Section 5.

TCN SETTINGS							
Dataset/Task	**Subtask**	k	n	**Hidden**	**Dropout**	**Grad Clip**	**Note**
The Adding Problem	$T = 200$	6	7	27	0.0	N/A	
	$T = 400$	7	7	27			
	$T = 600$	8	8	24			
Seq. MNIST	-	7	8	25	0.0	N/A	
		6	8	20			
Permuted MNIST	-	7	8	25	0.0	N/A	
		6	8	20			
Copy Memory Task	$T = 500$	6	9	10	0.05	1.0	RMSprop 5e-4
	$T = 1000$	8	8	10			
	$T = 2000$	8	9	10			
Music JSB Chorales	-	3	2	150	0.5	0.4	
Music Nottingham	-	6	4	150	0.2	0.4	
Word-level LM	PTB	3	4	600	0.5	0.4	Embed. size 600
	Wiki-103	3	5	1000	0.4		Embed. size 400
	LAMBADA	4	5	500			Embed. size 500
Char-level LM	PTB	3	3	450	0.1	0.15	Embed. size 100
	text8	2	5	520			

Dauphin et al. (2017) compared the effects of gated linear units (GLU) and gated tanh units (GTU), and adopted GLU in their non-dilated gated ConvNet. Following the same choice, we now compare TCNs using ReLU and TCNs with gating (GLU), represented by an elementwise product between two convolutional layers, with one of them also passing through a sigmoid function $\sigma(x)$. Note that the gates architecture uses approximately twice as many convolutional layers as the ReLU-TCN.

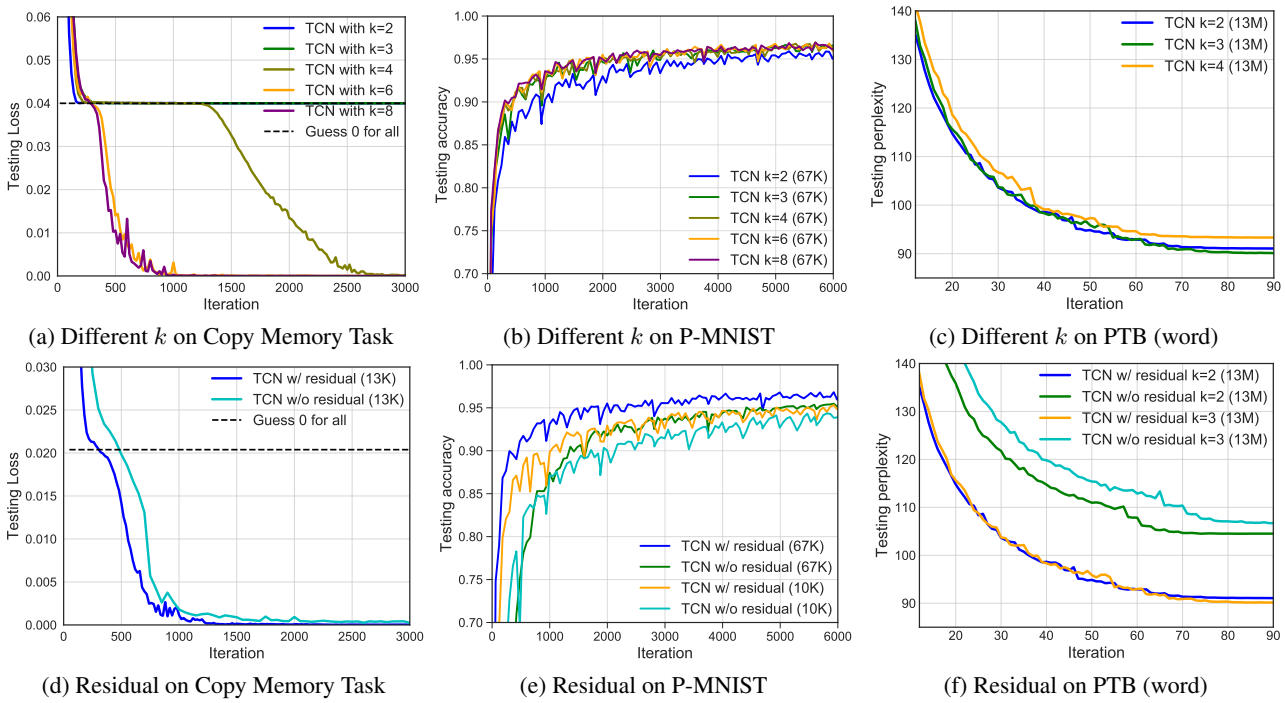
The results are shown in Table 5, where we kept the number of model parameters at about the same size. The GLU does further improve TCN accuracy on certain language modeling datasets like PTB, which agrees with prior work. However, we do not observe comparable benefits on other tasks, such as polyphonic music modeling or synthetic stress tests that require longer information retention. On the copy memory task with $T = 1000$, we found that TCN with gating converged to a worse result than TCN with ReLU (though still better than recurrent models).

Table 3. LSTM parameter settings for experiments in Section 5.

LSTM SETTINGS (KEY PARAMETERS)							
Dataset/Task	**Subtask**	n	**Hidden**	**Dropout**	**Grad Clip**	**Bias**	**Note**
The Adding Problem	$T = 200$	2	77	0.0	50	5.0	SGD 1e-3
	$T = 400$	2	77		50	10.0	Adam 2e-3
	$T = 600$	1	130		5	1.0	-
Seq. MNIST	-	1	130	0.0	1	1.0	RMSprop 1e-3
Permuted MNIST	-	1	130	0.0	1	10.0	RMSprop 1e-3
Copy Memory Task	$T = 500$	1	50	0.05	0.25	-	RMSprop/Adam
	$T = 1000$	1	50		1		
	$T = 2000$	3	28		1		
Music JSB Chorales	-	2	200	0.2	1	10.0	SGD/Adam
Music Nottingham	-	3	280	0.1	0.5	-	Adam 4e-3
		1	500		1	-	
Word-level LM	PTB	3	700	0.4	0.3	1.0	SGD 30, Emb. 700, etc.
	Wiki-103	-	-	-	-	-	Grave et al. (2017)
	LAMBADA	-	-	-	-	-	Grave et al. (2017)
Char-level LM	PTB	2	600	0.1	0.5	-	Emb. size 120
	text8	1	1024	0.15	0.5	-	Adam 1e-2

Table 4. State-of-the-art (SoTA) results for tasks in Section 5.

TCN VS. SOTA RESULTS					
Task	**TCN Result**	**Size**	**SoTA**	**Size**	**Model**
Seq. MNIST (acc.)	99.0	21K	99.0	21K	Dilated GRU (Chang et al., 2017)
P-MNIST (acc.)	97.2	42K	95.9	42K	Zoneout (Krueger et al., 2017)
Adding Prob. 600 (loss)	5.8e-5	70K	5.3e-5	70K	Regularized GRU
Copy Memory 1000 (loss)	3.5e-5	70K	0.011	70K	EURNN (Jing et al., 2017)
JSB Chorales (loss)	8.10	300K	3.47	-	DBN+LSTM (Vohra et al., 2015)
Nottingham (loss)	3.07	1M	1.32	-	DBN+LSTM (Vohra et al., 2015)
Word PTB (ppl)	88.68	13M	47.7	22M	AWD-LSTM-MoS + Dynamic Eval. (Yang et al., 2018)
Word Wiki-103 (ppl)	45.19	148M	40.4	>300M	Neural Cache Model (Large) (Grave et al., 2017)
Word LAMBADA (ppl)	1279	56M	138	>100M	Neural Cache Model (Large) (Grave et al., 2017)
Char PTB (bpc)	1.31	3M	1.22	14M	2-LayerNorm HyperLSTM (Ha et al., 2017)
Char text8 (bpc)	1.45	4.6M	1.29	>12M	HM-LSTM (Chung et al., 2016)



(a) Different k on Copy Memory Task (b) Different k on P-MNIST (c) Different k on PTB (word)

(d) Residual on Copy Memory Task (e) Residual on P-MNIST (f) Residual on PTB (word)

Figure 6. Controlled experiments that study the effect of different components of the TCN model.

Table 5. An evaluation of gating in TCN. A plain TCN is compared to a TCN that uses gated activations.

Task	**TCN**	**TCN + Gating**
Sequential MNIST (acc.)	**99.0**	**99.0**
Permuted MNIST (acc.)	**97.2**	96.9
Adding Problem $T = 600$ (loss)	**5.8e-5**	**5.6e-5**
Copy Memory $T = 1000$ (loss)	**3.5e-5**	0.00508
JSB Chorales (loss)	**8.10**	8.13
Nottingham (loss)	**3.07**	3.12
Word-level PTB (ppl)	88.68	**87.94**
Char-level PTB (bpc)	1.31	**1.306**
Char text8 (bpc)	**1.45**	1.485

Phrase-Based & Neural Unsupervised Machine Translation

Guillaume Lample[†]
Facebook AI Research
Sorbonne Universités
glample@fb.com

Myle Ott
Facebook AI Research
myleott@fb.com

Alexis Conneau
Facebook AI Research
Université Le Mans
aconneau@fb.com

Ludovic Denoyer[†]
Sorbonne Universités
ludovic.denoyer@lip6.fr

Marc'Aurelio Ranzato
Facebook AI Research
ranzato@fb.com

Abstract

Machine translation systems achieve near human-level performance on some languages, yet their effectiveness strongly relies on the availability of large amounts of parallel sentences, which hinders their applicability to the majority of language pairs. This work investigates how to learn to translate when having access to only large monolingual corpora in each language. We propose two model variants, a neural and a phrase-based model. Both versions leverage a careful initialization of the parameters, the denoising effect of language models and automatic generation of parallel data by iterative back-translation. These models are significantly better than methods from the literature, while being simpler and having fewer hyper-parameters. On the widely used WMT'14 English-French and WMT'16 German-English benchmarks, our models respectively obtain 28.1 and 25.2 BLEU points without using a single parallel sentence, outperforming the state of the art by more than 11 BLEU points. On low-resource languages like English-Urdu and English-Romanian, our methods achieve even better results than semi-supervised and supervised approaches leveraging the paucity of available bitexts. Our code for NMT and PBSMT is publicly available.[1]

1 Introduction

Machine Translation (MT) is a flagship of the recent successes and advances in the field of natural language processing. Its practical applications and use as a testbed for sequence transduction algorithms have spurred renewed interest in this topic.

While recent advances have reported near human-level performance on several language pairs using neural approaches (Wu et al., 2016; Hassan et al., 2018), other studies have highlighted several open challenges (Koehn and Knowles, 2017; Isabelle et al., 2017; Sennrich, 2017). A major challenge is the reliance of current learning algorithms on large parallel corpora. Unfortunately, the vast majority of language pairs have very little, if any, parallel data: learning algorithms need to better leverage monolingual data in order to make MT more widely applicable.

While a large body of literature has studied the use of monolingual data to boost translation performance when limited supervision is available, two recent approaches have explored the fully unsupervised setting (Lample et al., 2018; Artetxe et al., 2018), relying only on monolingual corpora in each language, as in the pioneering work by Ravi and Knight (2011). While there are subtle technical differences between these two recent works, we identify several common principles underlying their success.

First, they carefully initialize the MT system with an inferred bilingual dictionary. Second, they leverage strong language models, via training the sequence-to-sequence system (Sutskever et al., 2014; Bahdanau et al., 2015) as a denoising autoencoder (Vincent et al., 2008). Third, they turn the unsupervised problem into a supervised one by automatic generation of sentence pairs via *back-translation* (Sennrich et al., 2015a), i.e., the source-to-target model is applied to source sentences to generate inputs for training the target-to-source model, and vice versa. Finally, they constrain the latent representations produced by the encoder to be shared across the two languages. Empirically, these methods achieve remarkable results considering the fully unsupervised setting; for instance, about 15 BLEU points on the WMT'14 English-French benchmark.

The first contribution of this paper is a model

[†]Sorbonne Universités, UPMC Univ Paris 06, CNRS, UMR 7606, LIP6, F-75005, Paris, France.

[1]https://github.com/facebookresearch/UnsupervisedMT

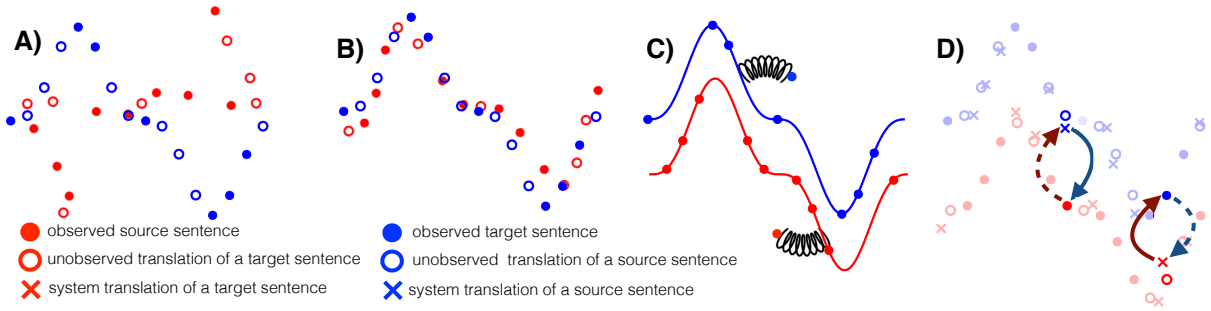


Figure 1: Toy illustration of the three principles of unsupervised MT. **A)** There are two monolingual datasets. Markers correspond to sentences (see legend for details). **B)** First principle: **Initialization**. The two distributions are roughly aligned, e.g. by performing word-by-word translation with an inferred bilingual dictionary. **C)** Second principle: **Language modeling**. A language model is learned independently in each domain to infer the structure in the data (underlying continuous curve); it acts as a data-driven prior to denoise/correct sentences (illustrated by the spring pulling a sentence outside the manifold back in). **D)** Third principle: **Back-translation**. Starting from an observed source sentence (filled red circle) we use the current source $\rightarrow$ target model to translate (dashed arrow), yielding a potentially incorrect translation (blue cross near the empty circle). Starting from this (back) translation, we use the target $\rightarrow$ source model (continuous arrow) to reconstruct the sentence in the original language. The discrepancy between the reconstruction and the initial sentence provides error signal to train the target $\rightarrow$ source model parameters. The same procedure is applied in the opposite direction to train the source $\rightarrow$ target model.

that combines these two previous neural approaches, simplifying the architecture and loss function while still following the above mentioned principles. The resulting model outperforms previous approaches and is both easier to train and tune. Then, we apply the same ideas and methodology to a traditional phrase-based statistical machine translation (PBSMT) system (Koehn et al., 2003). PBSMT models are well-known to outperform neural models when labeled data is scarce because they merely count occurrences, whereas neural models typically fit hundred of millions of parameters to learn distributed representations, which may generalize better when data is abundant but is prone to overfit when data is scarce. Our PBSMT model is simple, easy to interpret, fast to train and often achieves similar or better results than its NMT counterpart. We report gains of up to +10 BLEU points on widely used benchmarks when using our NMT model, and up to +12 points with our PBSMT model. Furthermore, we apply these methods to distant and low-resource languages, like English-Russian, English-Romanian and English-Urdu, and report competitive performance against both semi-supervised and supervised baselines.

2 Principles of Unsupervised MT

Learning to translate with only monolingual data is an ill-posed task, since there are potentially many ways to associate target with source sentences. Nevertheless, there has been exciting progress towards this goal in recent years, as discussed in the related work of Section 5. In this section, we abstract away from the specific assumptions made by each prior work and instead focus on identifying the common principles underlying unsupervised MT.

We claim that unsupervised MT can be accomplished by leveraging the three components illustrated in Figure 1: (i) suitable initialization of the translation models, (ii) language modeling and (iii) iterative back-translation. In the following, we describe each of these components and later discuss how they can be better instantiated in both a neural model and phrase-based model.

Initialization: Given the ill-posed nature of the task, model initialization expresses a natural prior over the space of solutions we expect to reach, jump-starting the process by leveraging approximate translations of words, short phrases or even sub-word units (Sennrich et al., 2015b). For instance, Klementiev et al. (2012) used a provided bilingual dictionary, while Lample et al. (2018) and Artetxe et al. (2018) used dictionaries inferred in an unsupervised way (Conneau et al., 2018; Artetxe et al., 2017). The motivating intuition is that while such initial "word-by-word" translation may be poor if languages or corpora are not closely related, it still preserves some of the original semantics.

Language Modeling: Given large amounts of monolingual data, we can train language models on both source and target languages. These models express a data-driven prior about how sentences should read in each language, and they improve the quality of the translation models by per-

Algorithm 1: Unsupervised MT

Language models: Learn language models P_s and P_t over source and target languages;
Initial translation models: Leveraging P_s and P_t, learn two initial translation models, one in each direction: $P^{(0)}_{s\to t}$ and $P^{(0)}_{t\to s}$;
for k=1 **to** N **do**
  Back-translation: Generate source and target sentences using the current translation models, $P^{(k-1)}_{t\to s}$ and $P^{(k-1)}_{s\to t}$, factoring in language models, P_s and P_t;
  Train new translation models $P^{(k)}_{s\to t}$ and $P^{(k)}_{t\to s}$ using the generated sentences and leveraging P_s and P_t;
end

forming local substitutions and word reorderings.

Iterative Back-translation: The third principle is back-translation (Sennrich et al., 2015a), which is perhaps the most effective way to leverage monolingual data in a semi-supervised setting. Its application in the unsupervised setting is to couple the source-to-target translation system with a backward model translating from the target to source language. The goal of this model is to generate a source sentence for each target sentence in the monolingual corpus. This turns the daunting unsupervised problem into a supervised learning task, albeit with noisy source sentences. As the original model gets better at translating, we use the current model to improve the back-translation model, resulting in a coupled system trained with an iterative algorithm (He et al., 2016).

3 Unsupervised MT systems

Equipped with the three principles detailed in Section 2, we now discuss how to effectively combine them in the context of a NMT model (Section 3.1) and PBSMT model (Section 3.2).

In the reminder of the paper, we denote the space of source and target sentences by $\mathcal{S}$ and $\mathcal{T}$, respectively, and the language models trained on source and target monolingual datasets by P_s and P_t, respectively. Finally, we denote by $P_{s\to t}$ and $P_{t\to s}$ the translation models from source to target and vice versa. An overview of our approach is given in Algorithm 1.

3.1 Unsupervised NMT

We now introduce a new unsupervised NMT method, which is derived from earlier work by Artetxe et al. (2018) and Lample et al. (2018). We first discuss how the previously mentioned three key principles are instantiated in our work, and then introduce an additional property of the system, the sharing of internal representations across languages, which is specific and critical to NMT. From now on, we assume that a NMT model consists of an encoder and a decoder. Section 4 gives the specific details of this architecture.

Initialization: While prior work relied on bilingual dictionaries, here we propose a more effective and simpler approach which is particularly suitable for related languages.[2] First, instead of considering words, we consider byte-pair encodings (BPE) (Sennrich et al., 2015b), which have two major advantages: they reduce the vocabulary size and they eliminate the presence of unknown words in the output translation. Second, instead of learning an explicit mapping between BPEs in the source and target languages, we define BPE tokens by *jointly* processing both monolingual corpora. If languages are related, they will naturally share a good fraction of BPE tokens, which eliminates the need to infer a bilingual dictionary. In practice, we i) join the monolingual corpora, ii) apply BPE tokenization on the resulting corpus, and iii) learn token embeddings (Mikolov et al., 2013) on the same corpus, which are then used to initialize the lookup tables in the encoder and decoder.

Language Modeling: In NMT, language modeling is accomplished via denoising autoencoding, by minimizing:

$$\mathcal{L}^{lm} = \mathbb{E}_{x\sim\mathcal{S}}[-\log P_{s\to s}(x|C(x))] + \mathbb{E}_{y\sim\mathcal{T}}[-\log P_{t\to t}(y|C(y))] \quad (1)$$

where C is a noise model with some words dropped and swapped as in Lample et al. (2018). $P_{s\to s}$ and $P_{t\to t}$ are the composition of encoder and decoder both operating on the source and target sides, respectively.

Back-translation: Let us denote by $u^*(y)$ the sentence in the source language inferred from $y \in \mathcal{T}$ such that $u^*(y) = \arg\max P_{t\to s}(u|y)$. Similarly, let us denote by $v^*(x)$ the sentence in the target language inferred from $x \in \mathcal{S}$ such that $v^*(x) = \arg\max P_{s\to t}(v|x)$. The pairs $(u^*(y), y)$ and $(x, v^*(x)))$ constitute automatically-generated parallel sentences which, following the back-translation principle, can be

[2] For unrelated languages, we need to infer a dictionary to properly initialize the embeddings (Conneau et al., 2018).

used to train the two MT models by minimizing the following loss:

$$\mathcal{L}^{back} = \mathbb{E}_{y\sim\mathcal{T}}[-\log P_{s\to t}(y|u^*(y))] + \mathbb{E}_{x\sim\mathcal{S}}[-\log P_{t\to s}(x|v^*(x))]. \quad (2)$$

Note that when minimizing this objective function we do not back-prop through the reverse model which generated the data, both for the sake of simplicity and because we did not observe improvements when doing so. The objective function minimized at every iteration of stochastic gradient descent, is simply the sum of $\mathcal{L}^{lm}$ in Eq. 1 and $\mathcal{L}^{back}$ in Eq. 2. To prevent the model from cheating by using different subspaces for the language modeling and translation tasks, we add an additional constraint which we discuss next.

Sharing Latent Representations: A shared encoder representation acts like an interlingua, which is translated in the decoder target language regardless of the input source language. This ensures that the benefits of language modeling, implemented via the denoising autoencoder objective, nicely transfer to translation from noisy sources and eventually help the NMT model to translate more fluently. In order to share the encoder representations, we share all encoder parameters (including the embedding matrices since we perform joint tokenization) across the two languages to ensure that the latent representation of the source sentence is robust to the source language. Similarly, we share the decoder parameters across the two languages. While sharing the encoder is critical to get the model to work, sharing the decoder simply induces useful regularization. Unlike prior work (Johnson et al., 2016), the first token of the decoder specifies the language the module is operating with, while the encoder does not have any language identifier.

3.2 Unsupervised PBSMT

In this section, we discuss how to perform unsupervised machine translation using a Phrase-Based Statistical Machine Translation (PBSMT) system (Koehn et al., 2003) as the underlying backbone model. Note that PBSMT models are known to perform well on low-resource language pairs, and are therefore a potentially good alternative to neural models in the unsupervised setting.

When translating from x to y, a PBSMT system scores y according to: $\arg\max_y P(y|x) = \arg\max_y P(x|y)P(y)$, where $P(x|y)$ is derived from so called "phrase tables", and $P(y)$ is the score assigned by a language model.

Given a dataset of bitexts, PBSMT first infers an alignment between source and target phrases. It then populates phrase tables, whose entries store the probability that a certain n-gram in the source/target language is mapped to another n-gram in the target/source language.

In the unsupervised setting, we can easily train a language model on monolingual data, but it is less clear how to populate the phrase tables, which are a necessary component for good translation. Fortunately, similar to the neural case, the principles of Section 2 are effective to solve this problem.

Initialization: We populate the initial phrase tables (from source to target and from target to source) using an inferred bilingual dictionary built from monolingual corpora using the method proposed by Conneau et al. (2018). In the following, we will refer to phrases as single words, but the very same arguments trivially apply to longer n-grams. Phrase tables are populated with the scores of the translation of a source word to:

$$p(t_j|s_i) = \frac{e^{\frac{1}{T}\cos(e(t_j),We(s_i))}}{\sum_k e^{\frac{1}{T}\cos(e(t_k),We(s_i))}}, \quad (3)$$

where t_j is the j-th word in the target vocabulary and s_i is the i-th word in the source vocabulary, T is a hyper-parameter used to tune the peakiness of the distribution[3], W is the rotation matrix mapping the source embeddings into the target embeddings (Conneau et al., 2018), and $e(x)$ is the embedding of x.

Language Modeling: Both in the source and target domains we learn smoothed n-gram language models using KenLM (Heafield, 2011), although neural models could also be considered. These remain fixed throughout training iterations.

Iterative Back-Translation: To jump-start the iterative process, we use the unsupervised phrase tables and the language model on the target side to construct a seed PBSMT. We then use this model to translate the source monolingual corpus into the target language (back-translation step). Once the data has been generated, we train a PBSMT in supervised mode to map the generated data back to the original source sentences. Next, we perform

[3]We set $T = 30$ in all our experiments, following the setting of Smith et al. (2017).

both generation and training process but in the reverse direction. We repeat these steps as many times as desired (see Algorithm 2 in Section A).

Intuitively, many entries in the phrase tables are not correct because the input to the PBSMT at any given point during training is noisy. Despite that, the language model may be able to fix some of these mistakes at generation time. As long as that happens, the translation improves, and with that also the phrase tables at the next round. There will be more entries that correspond to correct phrases, which makes the PBSMT model stronger because it has bigger tables and it enables phrase swaps over longer spans.

4 Experiments

We first describe the datasets and experimental protocol we used. Then, we compare the two proposed unsupervised approaches to earlier attempts, to semi-supervised methods and to the very same models but trained with varying amounts of labeled data. We conclude with an ablation study to understand the relative importance of the three principles introduced in Section 2.

4.1 Datasets and Methodology

We consider five language pairs: English-French, English-German, English-Romanian, English-Russian and English-Urdu. The first two pairs are used to compare to recent work on unsupervised MT (Artetxe et al., 2018; Lample et al., 2018). The last three pairs are instead used to test our PBSMT unsupervised method on truly low-resource pairs (Gu et al., 2018) or unrelated languages that do not even share the same alphabet.

For English, French, German and Russian, we use all available sentences from the WMT monolingual News Crawl datasets from years 2007 through 2017. For Romanian, the News Crawl dataset is only composed of 2.2 million sentences, so we augment it with the monolingual data from WMT'16, resulting in 2.9 million sentences. In Urdu, we use the dataset of Jawaid et al. (2014), composed of about 5.5 million monolingual sentences. We report results on *newstest* 2014 for $en-fr$, and *newstest* 2016 for $en-de$, $en-ro$ and $en-ru$. For Urdu, we use the LDC2010T21 and LDC2010T23 corpora each with about 1800 sentences as validation and test sets, respectively.

We use Moses scripts (Koehn et al., 2007) for tokenization. NMT is trained with 60,000 BPE codes. PBSMT is trained with true-casing, and by removing diacritics from Romanian on the source side to deal with their inconsistent use across the monolingual dataset (Sennrich et al., 2016).

Source	Target	$P(s\|t)$	$P(t\|s)$
heureux	happy	0.931	0.986
	delighted	0.458	0.003
	grateful	0.128	0.003
	thrilled	0.392	0.002
	glad	0.054	0.001
Royaume-Uni	Britain	0.242	0.720
	UK	0.816	0.257
	U.K.	0.697	0.011
	United Kingdom	0.770	0.010
	British	0.000	0.002
Union européenne	European Union	0.869	0.772
	EU	0.335	0.213
	E.U.	0.539	0.006
	member states	0.007	0.006
	27-nation bloc	0.410	0.002

Table 1: **Unsupervised phrase table.** Example of candidate French to English phrase translations, along with their corresponding conditional likelihoods.

4.2 Initialization

Both the NMT and PBSMT approaches require either cross-lingual BPE embeddings (to initialize the shared lookup tables) or n-gram embeddings (to initialize the phrase table). We generate embeddings using fastText (Bojanowski et al., 2017) with an embedding dimension of 512, a context window of size 5 and 10 negative samples. For NMT, fastText is applied on the concatenation of source and target corpora, which results in cross-lingual BPE embeddings.

For PBSMT, we generate n-gram embeddings on the source and target corpora independently, and align them using the MUSE library (Conneau et al., 2018). Since learning unique embeddings of every possible phrase would be intractable, we consider the most frequent 300,000 source phrases, and align each of them to its 200 nearest neighbors in the target space, resulting in a phrase table of 60 million phrase pairs which we score using the formula in Eq. 3.

In practice, we observe a small but significant difference of about 1 BLEU point using a phrase table of bigrams compared to a phrase table of unigrams, but did not observe any improvement using longer phrases. Table 1 shows an extract of a French-English unsupervised phrase table, where we can see that unigrams are correctly aligned to bigrams, and vice versa.

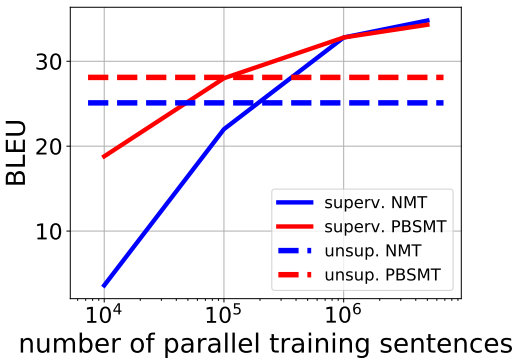


Figure 2: Comparison between supervised and unsupervised approaches on WMT'14 En-Fr, as we vary the number of parallel sentences for the supervised methods.

4.3 Training

The next subsections provide details about the architecture and training procedure of our models.

4.3.1 NMT

In this study, we use NMT models built upon LSTM (Hochreiter and Schmidhuber, 1997) and Transformer (Vaswani et al., 2017) cells. For the LSTM model we use the same architecture as in Lample et al. (2018). For the Transformer, we use 4 layers both in the encoder and in the decoder. Following Press and Wolf (2016), we share all lookup tables between the encoder and the decoder, and between the source and the target languages. The dimensionality of the embeddings and of the hidden layers is set to 512. We used the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 10^{-4}, $\beta_1 = 0.5$, and a batch size of 32. At decoding time, we generate greedily.

4.3.2 PBSMT

The PBSMT uses Moses' default smoothed n-gram language model with phrase reordering disabled during the very first generation. PBSMT is trained in a iterative manner using Algorithm 2. At each iteration, we translate 5 million sentences randomly sampled from the monolingual dataset in the source language. Except for initialization, we use phrase tables with phrases up to length 4.

4.4 Model selection

Moses' implementation of PBSMT has 15 hyperparameters, such as relative weighting of each scoring function, word penalty, etc. In this work, we consider two methods to set these hyperparameters. We either set them to their default values in the toolbox, or we set them using a small validation set of parallel sentences. It turns out that with only 100 labeled sentences in the validation set, PBSMT would overfit to the validation set. For instance, on $en \rightarrow fr$, PBSMT tuned on 100 parallel sentences obtains a BLEU score of 26.42 on *newstest* 2014, compared to 27.09 with default hyper-parameters, and 28.02 when tuned on the 3000 parallel sentences of *newstest* 2013. Therefore, unless otherwise specified, all PBSMT models considered in the paper use default hyperparameter values, and do not use any parallel resource whatsoever.

For the NMT, we also consider two model selection procedures: an *unsupervised criterion* based on the BLEU score of a "round-trip" translation (source $\rightarrow$ target $\rightarrow$ source and target $\rightarrow$ source $\rightarrow$ target) as in Lample et al. (2018), and cross-validation using a small validation set with 100 parallel sentences. In our experiments, we found the unsupervised criterion to be highly correlated with the test metric when using the Transformer model, but not always for the LSTM. Therefore, unless otherwise specified, we select the best LSTM models using a small validation set of 100 parallel sentences, and the best Transformer models with the unsupervised criterion.

Model	en-fr	fr-en	en-de	de-en
(Artetxe et al., 2018)	15.1	15.6	-	-
(Lample et al., 2018)	15.0	14.3	9.6	13.3
(Yang et al., 2018)	17.0	15.6	10.9	14.6
NMT (LSTM)	24.5	23.7	14.7	19.6
NMT (Transformer)	25.1	24.2	17.2	21.0
PBSMT (Iter. 0)	16.2	17.5	11.0	15.6
PBSMT (Iter. n)	**28.1**	27.2	17.9	22.9
NMT + PBSMT	27.1	26.3	17.5	22.1
PBSMT + NMT	27.6	**27.7**	**20.2**	**25.2**

Table 2: **Comparison with previous approaches.** BLEU score for different models on the $en - fr$ and $en - de$ language pairs. Just using the unsupervised phrase table, and without back-translation (PBSMT (Iter. 0)), the PBSMT outperforms previous approaches. Combining PBSMT with NMT gives the best results.

4.5 Results

The results reported in Table 2 show that our unsupervised NMT and PBSMT systems largely outperform previous unsupervised baselines. We report large gains on all language pairs and directions. For instance, on the $en \rightarrow fr$ task, our unsupervised PBSMT obtains a BLEU score of 28.1, outperforming the previous best result by more than 11 BLEU points. Even on a more complex task like $en \rightarrow de$, both PBSMT and NMT surpass the baseline score by more than 10 BLEU

	en → fr	fr→ en	en→ de	de→ en	en→ ro	ro→ en	en→ ru	ru→ en
Unsupervised PBSMT								
Unsupervised phrase table	-	17.50	-	15.63	-	14.10	-	8.08
Back-translation - Iter. 1	24.79	26.16	15.92	22.43	18.21	21.49	11.04	15.16
Back-translation - Iter. 2	27.32	26.80	17.65	22.85	20.61	22.52	12.87	16.42
Back-translation - Iter. 3	27.77	26.93	17.94	22.87	21.18	22.99	13.13	16.52
Back-translation - Iter. 4	27.84	27.20	17.77	22.68	21.33	23.01	13.37	**16.62**
Back-translation - Iter. 5	**28.11**	27.16	-	-	-	-	-	-
Unsupervised NMT								
LSTM	24.48	23.74	14.71	19.60	-	-	-	-
Transformer	25.14	24.18	17.16	21.00	21.18	19.44	7.98	9.09
Phrase-based + Neural network								
NMT + PBSMT	27.12	26.29	17.52	22.06	21.95	23.73	10.14	12.62
PBSMT + NMT	27.60	**27.68**	**20.23**	**25.19**	**25.13**	**23.90**	**13.76**	**16.62**

Table 3: **Fully unsupervised results.** We report the BLEU score for PBSMT, NMT, and their combinations on 8 directed language pairs. Results are obtained on *newstest* 2014 for $en - fr$ and *newstest* 2016 for every other pair.

points. Even before iterative back-translation, the PBSMT model significantly outperforms previous approaches, and can be trained in a few minutes.

Table 3 illustrates the quality of the PBSMT model during the iterative training process. For instance, the $fr \rightarrow en$ model obtains a BLEU score of 17.5 at iteration 0 – i.e. after the unsupervised phrase table construction – while it achieves a score of 27.2 at iteration 4. This highlights the importance of multiple back-translation iterations. The last rows of Table 3 also show that we get additional gains by further tuning the NMT model on the data generated by PBSMT (PBSMT + NMT). We simply add the data generated by the unsupervised PBSMT system to the back-translated data produced by the NMT model. By combining PBSMT and NMT, we achieve BLEU scores of 20.2 and 25.2 on the challenging $en \rightarrow de$ and $de \rightarrow en$ translation tasks. While we also tried bootstraping the PBSMT model with back-translated data generated by a NMT model (NMT + PBSMT), this did not improve over PBSMT alone.

Next, we compare to fully supervised models. Figure 2 shows the performance of the same architectures trained in a fully supervised way using parallel training sets of varying size. The unsupervised PBSMT model achieves the same performance as its supervised counterpart trained on more than 100,000 parallel sentences.

This is confirmed on low-resource languages. In particular, on $ro \rightarrow en$, our unsupervised PBSMT model obtains a BLEU score of 23.9, outperforming Gu et al. (2018)'s method by 1 point, despite its use of 6,000 parallel sentences, a seed dictionary, and a multi-NMT system combining parallel resources from 5 different languages.

On Russian, our unsupervised PBSMT model obtains a BLEU score of 16.6 on $ru \rightarrow en$, showing that this approach works reasonably well on distant languages. Finally we train on $ur \rightarrow en$, which is both low resource and distant. In a supervised mode, PBSMT using the noisy and out-of-domain 800,000 parallel sentences from Tiedemann (2012) achieves a BLEU score of 9.8. Instead, our unsupervised PBSMT system achieves 12.3 BLEU using only a validation set of 1800 sentences to tune Moses hyper-parameters.

4.6 Ablation Study

In Figure 3 we report results from an ablation study, to better understand the importance of the three principles when training PBSMT. This study shows that more iterations only partially compensate for lower quality phrase table initialization (Left), language models trained over less data (Middle) or less monolingual data (Right). Moreover, the influence of the quality of the language model becomes more prominent as we iterate. These findings suggests that better initialization methods and more powerful language models may further improve our results.

We perform a similar ablation study for the NMT system (see Appendix). We find that back-translation and auto-encoding are critical components, without which the system fails to learn. We also find that initialization of embeddings is very important, and we gain 7 BLEU points compared to prior work (Artetxe et al., 2018; Lample et al., 2018) by learning BPE embeddings over the concatenated monolingual corpora.

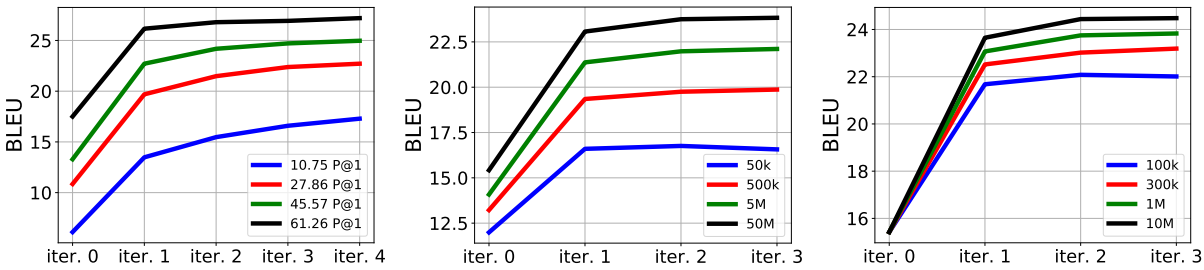


Figure 3: Results with PBSMT on the $fr \rightarrow en$ pair at different iterations. We vary: Left) the quality of the initial alignment between the source and target embeddings (measured in P@1 on the word translation task), Middle) the number of sentences used to train the language models, Right) the number of sentences used for back-translation.

5 Related Work

A large body of literature has studied using monolingual data to boost translation performance when limited supervision is available. This limited supervision is typically provided as a small set of parallel sentences (Sennrich et al., 2015a; Gulcehre et al., 2015; He et al., 2016; Gu et al., 2018; Wang et al., 2018); large sets of parallel sentences in related languages (Firat et al., 2016; Johnson et al., 2016; Chen et al., 2017; Zheng et al., 2017); cross-lingual dictionaries (Klementiev et al., 2012; Irvine and Callison-Burch, 2014, 2016); or comparable corpora (Munteanu et al., 2004; Irvine and Callison-Burch, 2013).

Learning to translate *without* any form of supervision has also attracted interest, but is challenging. In their seminal work, Ravi and Knight (2011) leverage linguistic prior knowledge to reframe the unsupervised MT task as deciphering and demonstrate the feasibility on short sentences with limited vocabulary. Earlier work by Carbonell et al. (2006) also aimed at unsupervised MT, but leveraged a bilingual dictionary to seed the translation. Both works rely on a language model on the target side to correct for translation fluency.

Subsequent work (Klementiev et al., 2012; Irvine and Callison-Burch, 2014, 2016) relied on bilingual dictionaries, small parallel corpora of several thousand sentences, and linguistically motivated features to prune the search space. Irvine and Callison-Burch (2014) use monolingual data to expand phrase tables learned in a supervised setting. In our work we also expand phrase tables, but we initialize them with an inferred bilingual n-gram dictionary, following work from the connectionist community aimed at improving PBSMT with neural models (Schwenk, 2012; Kalchbrenner and Blunsom, 2013; Cho et al., 2014).

In recent years back-translation has become a popular method of augmenting training sets with monolingual data on the target side (Sennrich et al., 2015a), and has been integrated in the "dual learning" framework of He et al. (2016) and subsequent extensions (Wang et al., 2018). Our approach is similar to the dual learning framework, except that in their model gradients are backpropagated through the reverse model and they pretrain using a relatively large amount of labeled data, whereas our approach is fully unsupervised.

Finally, our work can be seen as an extension of recent studies (Lample et al., 2018; Artetxe et al., 2018; Yang et al., 2018) on *fully unsupervised* MT with two major contributions. First, we propose a much simpler and more effective initialization method for related languages. Second, we abstract away three principles of unsupervised MT and apply them to a PBSMT, which even outperforms the original NMT. Moreover, our results show that the combination of PBSMT and NMT achieves even better performance.

6 Conclusions and Future Work

In this work, we identify three principles underlying recent successes in fully unsupervised MT and show how to apply these principles to PBSMT and NMT systems. We find that PBSMT systems often outperform NMT systems in the fully unsupervised setting, and that by combining these systems we can greatly outperform previous approaches from the literature. We apply our approach to several popular benchmark language pairs, obtaining state of the art results, and to several low-resource and under-explored language pairs.

It's an open question whether there are more effective instantiations of these principles or other principles altogether, and under what conditions our iterative process is guaranteed to converge. Future work may also extend to the semi-supervised setting.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, volume 1, pages 451–462.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In International Conference on Learning Representations (ICLR).

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In International Conference on Learning Representations (ICLR).

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5:135–146.

Jaime Carbonell, Steve Klein, David Miller, Michael Steinbaum, Tomer Grassiany, and Jochen Frey. 2006. Context-based machine translation. In The Association for Machine Translation in the Americas.

Y. Chen, Y. Liu, Y. Cheng, and V.O.K. Li. 2017. A teacher-student framework for zero-resource neural machine translation. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder—decoder for statistical machine translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724——1734.

A. Conneau, G. Lample, M. Ranzato, L. Denoyer, and H. Jégou. 2018. Word translation without parallel data. In International Conference on Learning Representations (ICLR).

O. Firat, B. Sankaran, Y. Al-Onaizan, F.T.Y. Vural, and K. Cho. 2016. Zero-resource translation with multilingual neural machine translation. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O.K. Li. 2018. Universal neural machine translation for extremely low resource languages. In Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. arXiv preprint arXiv:1503.03535.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic chinese to english news translation. In arXiv:1803.05567.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In Advances in Neural Information Processing Systems, pages 820–828.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In Proceedings of the Sixth Workshop on Statistical Machine Translation, pages 187–197. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation, 9(8):1735–1780.

Ann Irvine and Chris Callison-Burch. 2013. Combining bilingual and comparable corpora for low resource machine translation. In Proceedings of the eighth workshop on statistical machine translation, pages 262–270.

Ann Irvine and Chris Callison-Burch. 2014. Hallucinating phrase translations for low resource mt. In Proceedings of the Eighteenth Conference on Computational Natural Language Learning, pages 160–170.

Ann Irvine and Chris Callison-Burch. 2016. End-to-end statistical machine translation with zero or small parallel texts. In Journal of Natural Language Engineering, volume 22, pages 517–548.

Pierre Isabelle, Colin Cherry, and George Foster. 2017. A challenge set approach to evaluating machine translation. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2486–2496.

Bushra Jawaid, Amir Kamran, and Ondrej Bojar. 2014. A tagged corpus and a tagger for urdu. In LREC.

M. Johnson, M. Schuster, Q.V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, M. Hughes, and J. Dean. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. In Transactions of the Association for Computational Linguistics.

Nal Kalchbrenner and Phil Blunsom. 2013. Two recurrent continuous translation models. In Conference on Empirical Methods in Natural Language Processing.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Alex Klementiev, Ann Irvine, Chris Callison-Burch, and David Yarowsky. 2012. Toward statistical machine translation without parallel corpora. In Proceedings of the 13th Conference of the European Chapter of the Association for computational Linguistics, Avignon, France. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Ondrej Bojar Chris Dyer, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In Annual Meeting of the Association for Computational Linguistics (ACL), demo session.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In Proceedings of the First Workshop on Neural Machine Translation, pages 28–39.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL), volume 1, pages 48—54.

G. Lample, A. Conneau, L. Denoyer, and M. Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In International Conference on Learning Representations (ICLR).

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, pages 3111–3119.

D.S. Munteanu, A. Fraser, and D. Marcu. 2004. Improved machine translation performance via parallel sentence extraction from comparable corpora. In Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (NAACL).

Ofir Press and Lior Wolf. 2016. Using the output embedding to improve language models. arXiv preprint arXiv:1608.05859.

S. Ravi and K. Knight. 2011. Deciphering foreign language. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 12–21.

Holger Schwenk. 2012. Continuous space translation models for phrase-based statistical machine translation. In International Conference on Computational Linguistics, pages 1071—1080.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? assessing mt quality with contrastive translation pairs. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, pages 376–382.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015a. Improving neural machine translation models with monolingual data. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 86–96.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015b. Neural machine translation of rare words with subword units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1715–1725.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for wmt 16. In Proceedings of the First Conference on Machine Translation, pages 371–376.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In Internaltional Conference on Learning Representations.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems, pages 3104–3112.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In Proceedings of the 8th International Conference on Language Resources and Evaluation.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. arXiv preprint arXiv:1706.03762.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th international conference on Machine learning, pages 1096–1103.

Yijun Wang, Yingce Xia, Li Zhao, Jiang Bian, Tao Qin, Guiquan Liu, and Tie-Yan Liu. 2018. Dual transfer learning for neural machine translation with marginal distribution regularization. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. Transactions of the Association for Computational Linguistics, pages 339–351.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Unsupervised neural machine translation with weight sharing. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics.

H. Zheng, Y. Cheng, and Y. Liu. 2017. Maximum expected likelihood estimation for zero-resource neural machine translation. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI), pages 4251–4257.

A Supplemental Material

In this Appendix, we report the detailed algorithm for unsupervised PBSMT, a detailed ablation study using NMT and conclude with some example translations.

Algorithm 2: Unsupervised PBSMT

Learn bilingual dictionary using Conneau et al. (2018);
Populate phrase tables using Eq. 3 and learn a language model to build $P^{(0)}_{s\to t}$;
Use $P^{(0)}_{s\to t}$ to translate the source monolingual dataset, yielding $\mathcal{D}^{(0)}_{\text{t}}$;
for i=1 **to** N **do**
    Train model $P^{(i)}_{t\to s}$ using $\mathcal{D}^{(i-1)}_{\text{t}}$;
    Use $P^{(i)}_{t\to s}$ to translate the target monolingual dataset, yielding $\mathcal{D}^{(i)}_{\text{s}}$;
    Train model $P^{(i)}_{s\to t}$ using $\mathcal{D}^{(i)}_{\text{s}}$;
    Use $P^{(i)}_{s\to t}$ to translate the source monolingual dataset, yielding $\mathcal{D}^{(i)}_{\text{t}}$;
end

A.1 NMT Ablation study

In Table 4 we report results from an ablation study we performed for NMT using the Transformer architecture. All results are for the $en \to fr$ task.

First, we analyze the effect of different initialization methods for the embedding matrices. If we switch from BPE tokens to words, BLEU drops by 4 points. If we used BPE but train embeddings in each language independently and then map them via MUSE (Conneau et al., 2018), BLEU drops by 3 points. Finally, compared to the word aligned procedure used by Lample et al. (2018), based on words and MUSE, the gain is about 7 points. To stress the importance of initialization, we also report performance using random initialization of BPE embeddings. In this case, convergence is much slower and to a much lower accuracy, achieving a BLEU score of 10.5.

The table also demonstrates the critical importance of the auto-encoding and back-translation terms in the loss, and the robustness of our approach to choice of architectures.

A.2 Qualitative study

Table 5 shows example translations from the French-English *newstest* 2014 dataset at different iterations of the learning algorithm for both NMT and PBSMT models. Prior to the first iteration of back-translation, using only the unsupervised phrase table, the PBSMT translations are similar to word-by-word translations and do not respect the syntax of the target language, yet still contain most of the semantics of the original sentences. As we increase the number of epochs in NMT and as we iterate for PBSMT, we observe a continuous improvement in the quality of the unsupervised translations. Interestingly, in the second example, both the PBSMT and NMT models fail to adapt to the polysemy of the French word "langue", which can be translated as "tongue" or "language" in English. These translations were both present in the unsupervised phrase table, but the conditional probability of "language" to be the correct translation of "langue" was very high compared to the one of "tongue": $P(\text{language}|\text{langue}) = 0.92$, while $P(\text{tongue}|\text{langue}) = 0.0005$. As a comparison, the phrase table of a Moses model trained in a supervised way contains $P(\text{language}|\text{langue}) = 0.633, P(\text{tongue}|\text{langue}) = 0.0076$, giving a higher probability for "langue" to be properly translated. This underlines the importance of the initial unsupervised phrase alignment procedure, as it was shown in Figure 1.

Finally, in Table 6 we report a random subset of test sentences translated from Russian to English, showing that the model mostly retains the semantics, while making some mistakes in the grammar as well as in the choice of words and entities. In Table 7, we show examples of translations from German to English with PBSMT, NMT, and PBSMT+NMT to show how the combination of these models performs better than them individually.

	en → fr	fr → en
Embedding Initialization		
Concat + fastText (BPE) [default]	25.1	24.2
Concat + fastText (Words)	21.0	20.9
fastText + Align (BPE)	22.0	21.3
fastText + Align (Words)	18.5	18.4
Random initialization	10.5	10.5
Loss function		
without $\mathcal{L}^{lm}$ of Eq. 1	0.0	0.0
without $\mathcal{L}^{back}$ of Eq. 2	0.0	0.0
Architecture		
without sharing decoder	24.6	23.7
LSTM instead of Transformer	24.5	23.7

Table 4: **Ablation study of unsupervised NMT.** BLEU scores are computed over *newstest* 2014.

Source	**Je rêve constamment d'eux, peut-être pas toutes les nuits mais plusieurs fois par semaine c'est certain.**
NMT Epoch 1	I constantly dream, but not all nights but by several times it is certain.
NMT Epoch 3	I continually dream them, perhaps not all but several times per week is certain.
NMT Epoch 45	I constantly dream of them, perhaps not all nights but several times a week it 's certain.
PBSMT Iter. 0	I dream of, but they constantly have all those nights but several times a week is too much. "
PBSMT Iter. 1	I had dreams constantly of them, probably not all nights but several times a week it is large.
PBSMT Iter. 4	I dream constantly of them, probably not all nights but several times a week it is certain.
Reference	**I constantly dream of them, perhaps not every night, but several times a week for sure.**

Source	**La protéine que nous utilisons dans la glace réagit avec la langue à pH neutre.**
NMT Epoch 1	The protein that we use in the ice with the language to pH.
NMT Epoch 8	The protein we use into the ice responds with language to pH neutral.
NMT Epoch 45	The protein we use in ice responds with the language from pH to neutral.
PBSMT Iter. 0	The protein that used in the ice responds with the language and pH neutral.
PBSMT Iter. 1	The protein that we use in the ice responds with the language to pH neutral.
PBSMT Iter. 4	The protein that we use in the ice reacts with the language to a neutral pH.
Reference	**The protein we are using in the ice cream reacts with your tongue at neutral pH.**

Source	**Selon Google, les déguisements les plus recherchés sont les zombies, Batman, les pirates et les sorcières.**
NMT Epoch 1	According to Google, there are more than zombies, Batman, and the pirates.
NMT Epoch 8	Google's most wanted outfits are the zombies, Batman, the pirates and the evil.
NMT Epoch 45	Google said the most wanted outfits are the zombies, Batman, the pirates and the witch.
PBSMT Iter. 0	According to Google, fancy dress and most wanted fugitives are the bad guys, Wolverine, the pirates and their minions.
PBSMT Iter. 1	According to Google, the outfits are the most wanted fugitives are zombies, Batman, pirates and witches.
PBSMT Iter. 4	According to Google, the outfits, the most wanted list are zombies, Batman, pirates and witches.
Reference	**According to Google, the highest searched costumes are zombies, Batman, pirates and witches.**

Table 5: **Unsupervised translations.** Examples of translations on the French-English pair of *newstest* 2014 at different iterations of training. For PBSMT, we show translations at iterations 0, 1 and 4, where the model obtains BLEU scores of 15.4, 23.7 and 24.7 respectively. For NMT, we show examples of translations after epochs 1, 8 and 42, where the model obtains BLEU scores of 12.3, 17.5 and 24.2 respectively. Iteration 0 refers to the PBSMT model obtained using the unsupervised phrase table, and an epoch corresponds to training the NMT model on 500k monolingual sentences. At the end of training, both models generate very good translations.

Russian → English	
Source	Изменения предусматривают сохранение льготы на проезд в общественном пассажирском транспорте.
Hypothesis	The changes involve keeping the benefits of parking in a public passenger transportation.
Reference	These changes make allowances for the preservation of discounted travel on public transportation.
Source	Шесть из 10 республиканцев говорят, что они согласны с Трампом по поводу иммиграции.
Hypothesis	Six in 10 Republicans say that they agree with Trump regarding immigration.
Reference	Six in 10 Republicans say they agree with Trump on immigration.
Source	Metcash пытается защитить свои магазины IGA от натиска Aldi в Южной Австралии и Западной Австралии .
Hypothesis	Metcash is trying to protect their shops IGA from the onslaught of Aldi in South Australia and Western Australia.
Reference	Metcash is trying to protect its IGA stores from an Aldi onslaught in South Australia and Western Australia.
Source	В них сегодня работают четыре сотни студентов из столичных колледжей и вузов.
Hypothesis	Others today employs four hundreds of students from elite colleges and universities.
Reference	Four hundred students from colleges and universities in the capital are working inside of it today.

Table 6: **Unsupervised translations: Russian-English.** Examples of translations on the Russian-English pair of *newstest* 2016 using the PBSMT model.

German → English	
Source	Flüchtlinge brauchen Unterkünfte : Studie warnt vor Wohnungslücke
PBSMT	Refugees need accommodation : Study warns Wohnungslücke
NMT	Refugees need forestry study to warn housing gap
PBSMT+NMT	Refugees need accommodation : Study warns of housing gap
Reference	Refugees need accommodation : study warns of housing gap
Source	Konflikte : Mehrheit unterstützt die Anti-IS-Ausbildungsmission
PBSMT	Conflict : Majority supports Anti-IS-Ausbildungsmission
NMT	Tensions support majority anti-IS-recruiting mission
PBSMT+NMT	Tensions : Majority supports the anti-IS-recruitment mission
Reference	Conflicts : the majority support anti ISIS training mission
Source	Roboterautos : Regierung will Vorreiterrolle für Deutschland
PBSMT	Roboterautos : Government will step forward for Germany
NMT	Robotic cars will pre-reiterate government for Germany
PBSMT+NMT	Robotic cars : government wants pre-orders for Germany
Reference	Robot cars : Government wants Germany to take a leading role
Source	Pfund steigt durch beschleunigtes Lohnwachstum im Vereinigten Königreich
PBSMT	Pound rises as UK beschleunigtes Lohnwachstum .
NMT	£ rises through rapid wage growth in the U.S.
PBSMT+NMT	Pound rises by accelerating wage growth in the United Kingdom
Reference	Pound rises as UK wage growth accelerates
Source	46 Prozent sagten , dass sie die Tür offen lassen , um zu anderen Kandidaten zu wechseln .
PBSMT	52 per cent said that they left the door open to them to switch to other candidates .
NMT	46 percent said that they would let the door open to switch to other candidates .
PBSMT+NMT	46 percent said that they left the door open to switch to other candidates .
Reference	46 percent said they are leaving the door open to switching candidates .
Source	Selbst wenn die Republikaner sich um einen anderen Kandidaten sammelten , schlägt Trump noch fast jeden .
PBSMT	Even if the Republicans a way to other candidates collected , beats Trump , yet almost everyone .
NMT	Even if Republicans are to donate to a different candidate , Trump takes to almost every place .
PBSMT+NMT	Even if Republicans gather to nominate another candidate , Trump still beats nearly everyone .
Reference	Even if Republicans rallied around another candidate , Trump still beats almost everyone .
Source	Ich glaube sicher , dass es nicht genügend Beweise gibt , um ein Todesurteil zu rechtfertigen .
PBSMT	I think for sure that there was not enough evidence to justify a death sentence .
NMT	I believe it 's not sure there 's enough evidence to justify a executions .
PBSMT+NMT	I sure believe there is not enough evidence to justify a death sentence .
Reference	I certainly believe there was not enough evidence to justify a death sentence .
Source	Auch wenn der Laden gut besucht ist , ist es nicht schwer , einen Parkplatz zu finden .
PBSMT	Even if the store visited it is , it is not hard to find a parking lot .
NMT	To be sure , the shop is well visited , but it 's not a troubled driveway .
PBSMT+NMT	Even if the shop is well visited , it is not hard to find a parking lot .
Reference	While the store can get busy , parking is usually not hard to find .
Source	Die Suite , in dem der kanadische Sänger wohnt , kostet am Tag genau so viel , wie ihre Mama Ewa im halben Jahr verdient .
PBSMT	The suite in which the Canadian singer grew up , costs on the day so much as their mum Vera in half year earned .
NMT	The Suite , in which the Canadian singer lived , costs day care exactly as much as her mom Ewa earned during the decade .
PBSMT+NMT	The suite , in which the Canadian singer lived , costs a day precisely as much as her mom Ewa earned in half .
Reference	The suite where the Canadian singer is staying costs as much for one night as her mother Ewa earns in six months .
Source	Der durchschnittliche BMI unter denen , die sich dieser Operation unterzogen , sank von 31 auf 24,5 bis Ende des fünften Jahres in dieser Studie .
PBSMT	The average BMI among those who undergoing this operation decreased by 22 to 325 by the end of fifth year in this study .
NMT	The average BMI among those who undergo surgery , sank from 31 to 24,500 by the end of the fifth year in that study .
PBSMT+NMT	The average BMI among those who undergo this surgery fell from 31 to 24.5 by the end of the fifth year in this study .
Reference	The average BMI among those who had surgery fell from 31 to 24.5 by the end of their fifth year in the study .
Source	Die 300 Plakate sind von Künstlern , die ihre Arbeit im Museum für grausame Designs in Banksys Dismaland ausgestellt haben .
PBSMT	The 250 posters are by artists and their work in the museum for gruesome designs in Dismaland Banksys have displayed .
NMT	The 300 posters are posters of artists who have their work Museum for real-life Designs in Banksys and Dismalausgestellt .
PBSMT+NMT	The 300 posters are from artists who have displayed their work at the Museum of cruel design in Banksys Dismaland .
Reference	The 300 posters are by artists who exhibited work at the Museum of Cruel Designs in Banksy 's Dismaland .
Source	Bis zum Ende des Tages gab es einen weiteren Tod : Lamm nahm sich das Leben , als die Polizei ihn einkesselte .
PBSMT	At the end of the day it 's a further death : Lamb took out life as police him einkesselte .
NMT	By the end of the day there was another death : Lamm emerged this year as police were trying to looseless him .
PBSMT+NMT	By the end of the day , there 's another death : Lamm took out life as police arrived to him .
Reference	By the end of the day , there would be one more death : Lamb took his own life as police closed in on him .
Source	Chaos folgte an der Grenze , als Hunderte von Migranten sich in einem Niemandsland ansammelten und serbische Beamte mit Empörung reagierten .
PBSMT	Chaos followed at the border , as hundreds of migrants in a frontier ansammelten and Serb officers reacted with outrage .
NMT	Chaos followed an avalanche as hundreds of thousands of immigrants fled to a mandsdom in answerage and Serbian officers responded with outrage .
PBSMT+NMT	Chaos followed at the border as hundreds of immigrants gathered in a bush town and Serb officers reacted with outrage .
Reference	Chaos ensued at the border , as hundreds of migrants piled up in a no man 's land , and Serbian officials reacted with outrage .
Source	" Zu unserer Reise gehörten viele dunkle Bahn- und Busfahrten , ebenso Hunger , Durst , Kälte und Angst " , schrieb sie .
PBSMT	" To our trip included many of the dark rail and bus rides , such as hunger , thirst , cold and fear , " she wrote .
NMT	" During our trip , many included dark bus and bus trips , especially hunger , Durst , and cold fear , " she wrote .
PBSMT+NMT	" Our trip included many dark rail and bus journeys , as well as hunger , thirst , cold and fear , " she wrote .
Reference	" Our journey involved many dark train and bus rides , as well as hunger , thirst , cold and fear , " she wrote .

Table 7: **Unsupervised translations: German-English.** Examples of translations on the German-English pair of *newstest* 2016 using the PBSMT, NMT, and PBSMT+NMT.

Linguistically-Informed Self-Attention for Semantic Role Labeling

Emma Strubell[1], **Patrick Verga**[1], **Daniel Andor**[2], **David Weiss**[2] **and Andrew McCallum**[1]
[1]College of Information and Computer Sciences
University of Massachusetts Amherst
{strubell, pat, mccallum}@cs.umass.edu
[2]Google AI Language
New York, NY
{andor, djweiss}@google.com

Abstract

Current state-of-the-art semantic role labeling (SRL) uses a deep neural network with no explicit linguistic features. However, prior work has shown that gold syntax trees can dramatically improve SRL decoding, suggesting the possibility of increased accuracy from explicit modeling of syntax. In this work, we present linguistically-informed self-attention (LISA): a neural network model that combines multi-head self-attention with multi-task learning across dependency parsing, part-of-speech tagging, predicate detection and SRL. Unlike previous models which require significant pre-processing to prepare linguistic features, LISA can incorporate syntax using merely raw tokens as input, encoding the sequence only once to simultaneously perform parsing, predicate detection and role labeling for all predicates. Syntax is incorporated by training one attention head to attend to syntactic parents for each token. Moreover, if a high-quality syntactic parse is already available, it can be beneficially injected at test time without re-training our SRL model. In experiments on CoNLL-2005 SRL, LISA achieves new state-of-the-art performance for a model using predicted predicates and standard word embeddings, attaining 2.5 F1 absolute higher than the previous state-of-the-art on newswire and more than 3.5 F1 on out-of-domain data, nearly 10% reduction in error. On ConLL-2012 English SRL we also show an improvement of more than 2.5 F1. LISA also out-performs the state-of-the-art with contextually-encoded (ELMo) word representations, by nearly 1.0 F1 on news and more than 2.0 F1 on out-of-domain text.

1 Introduction

Semantic role labeling (SRL) extracts a high-level representation of meaning from a sentence, labeling e.g. *who* did *what* to *whom*. Explicit representations of such semantic information have been shown to improve results in challenging downstream tasks such as dialog systems (Tur et al., 2005; Chen et al., 2013), machine reading (Berant et al., 2014; Wang et al., 2015) and translation (Liu and Gildea, 2010; Bazrafshan and Gildea, 2013).

Though syntax was long considered an obvious prerequisite for SRL systems (Levin, 1993; Punyakanok et al., 2008), recently deep neural network architectures have surpassed syntactically-informed models (Zhou and Xu, 2015; Marcheggiani et al., 2017; He et al., 2017; Tan et al., 2018; He et al., 2018), achieving state-of-the art SRL performance with no explicit modeling of syntax. An additional benefit of these end-to-end models is that they require just raw tokens and (usually) detected predicates as input, whereas richer linguistic features typically require extraction by an auxiliary pipeline of models.

Still, recent work (Roth and Lapata, 2016; He et al., 2017; Marcheggiani and Titov, 2017) indicates that neural network models could see even higher accuracy gains by leveraging syntactic information rather than ignoring it. He et al. (2017) indicate that many of the errors made by a syntax-free neural network on SRL are tied to certain syntactic confusions such as prepositional phrase attachment, and show that while constrained inference using a relatively low-accuracy predicted parse can provide small improvements in SRL accuracy, providing a gold-quality parse leads to substantial gains. Marcheggiani and Titov (2017) incorporate syntax from a high-quality parser (Kiperwasser and Goldberg, 2016) using graph convolutional neural networks (Kipf and Welling, 2017), but like He et al. (2017) they attain only small increases over a model with no syntactic parse, and even perform worse than a syntax-free model on out-of-domain data. These works suggest that though syntax has the potential to improve neural network SRL models, we have not

yet designed an architecture which maximizes the benefits of auxiliary syntactic information.

In response, we propose *linguistically-informed self-attention* (LISA): a model that combines multi-task learning (Caruana, 1993) with stacked layers of multi-head self-attention (Vaswani et al., 2017); the model is trained to: (1) jointly predict parts of speech and predicates; (2) perform parsing; and (3) attend to syntactic parse parents, while (4) assigning semantic role labels. Whereas prior work typically requires separate models to provide linguistic analysis, including most syntax-free neural models which still rely on external predicate detection, our model is truly end-to-end: earlier layers are trained to predict prerequisite parts-of-speech and predicates, the latter of which are supplied to later layers for scoring. Though prior work re-encodes each sentence to predict each desired task and again with respect to each predicate to perform SRL, we more efficiently encode each sentence only once, predict its predicates, part-of-speech tags and labeled syntactic parse, then predict the semantic roles for all predicates in the sentence in parallel. The model is trained such that, as syntactic parsing models improve, providing high-quality parses at test time will improve its performance, allowing the model to leverage updated parsing models without requiring re-training.

In experiments on the CoNLL-2005 and CoNLL-2012 datasets we show that our linguistically-informed models out-perform the syntax-free state-of-the-art. On CoNLL-2005 with predicted predicates and standard word embeddings, our single model out-performs the previous state-of-the-art model on the WSJ test set by 2.5 F1 points absolute. On the challenging out-of-domain Brown test set, our model improves substantially over the previous state-of-the-art by more than 3.5 F1, a nearly 10% reduction in error. On CoNLL-2012, our model gains more than 2.5 F1 absolute over the previous state-of-the-art. Our models also show improvements when using contextually-encoded word representations (Peters et al., 2018), obtaining nearly 1.0 F1 higher than the state-of-the-art on CoNLL-2005 news and more than 2.0 F1 improvement on out-of-domain text.[1]

[1]Our implementation in TensorFlow (Abadi et al., 2015) is available at : http://github.com/strubell/LISA

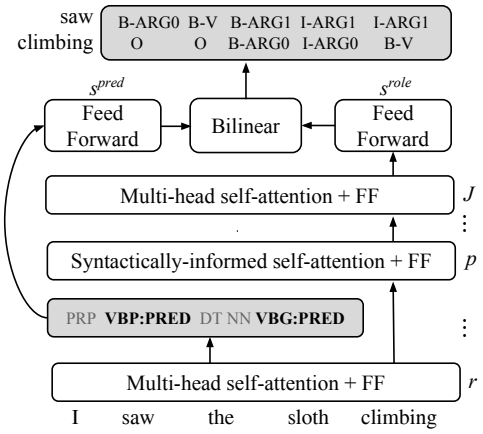


Figure 1: Word embeddings are input to J layers of multi-head self-attention. In layer p one attention head is trained to attend to parse parents (Figure 2). Layer r is input for a joint predicate/POS classifier. Representations from layer r corresponding to predicted predicates are passed to a bilinear operation scoring distinct predicate and role representations to produce per-token SRL predictions with respect to each predicted predicate.

2 Model

Our goal is to design an efficient neural network model which makes use of linguistic information as effectively as possible in order to perform end-to-end SRL. LISA achieves this by combining: (1) A new technique of supervising neural attention to predict syntactic dependencies with (2) multi-task learning across four related tasks.

Figure 1 depicts the overall architecture of our model. The basis for our model is the Transformer encoder introduced by Vaswani et al. (2017): we transform word embeddings into contextually-encoded token representations using stacked multi-head self-attention and feed-forward layers (§2.1).

To incorporate syntax, one self-attention head is trained to attend to each token's syntactic parent, allowing the model to use this attention head as an oracle for syntactic dependencies. We introduce this *syntactically-informed self-attention* (Figure 2) in more detail in §2.2.

Our model is designed for the more realistic setting in which gold predicates are not provided at test-time. Our model predicts predicates and integrates part-of-speech (POS) information into earlier layers by re-purposing representations closer to the input to predict predicate and POS tags us-

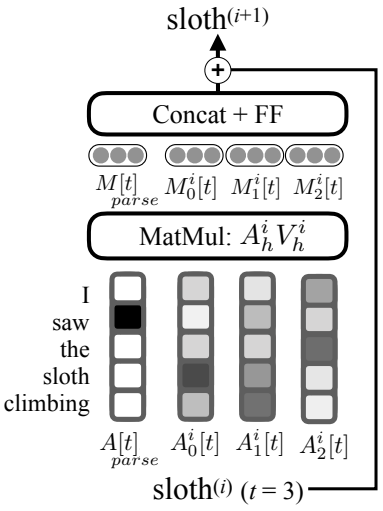


Figure 2: Syntactically-informed self-attention for the query word *sloth*. Attention weights A_{parse} heavily weight the token's syntactic governor, *saw*, in a weighted average over the token values V_{parse}. The other attention heads act as usual, and the attended representations from all heads are concatenated and projected through a feed-forward layer to produce the syntactically-informed representation for *sloth*.

ing hard parameter sharing (§2.3). We simplify optimization and benefit from shared statistical strength derived from highly correlated POS and predicates by treating tagging and predicate detection as a single task, performing multi-class classification into the joint Cartesian product space of POS and predicate labels.

Though typical models, which re-encode the sentence for each predicate, can simplify SRL to token-wise tagging, our joint model requires a different approach to classify roles with respect to each predicate. Contextually encoded tokens are projected to distinct *predicate* and *role* embeddings (§2.4), and each predicted predicate is scored with the sequence's role representations using a bilinear model (Eqn. 6), producing per-label scores for BIO-encoded semantic role labels for each token and each semantic frame.

The model is trained end-to-end by maximum likelihood using stochastic gradient descent (§2.5).

2.1 Self-attention token encoder

The basis for our model is a multi-head self-attention token encoder, recently shown to achieve state-of-the-art performance on SRL (Tan et al., 2018), and which provides a natural mechanism for incorporating syntax, as described in §2.2. Our implementation replicates Vaswani et al. (2017).

The input to the network is a sequence $\mathcal{X}$ of T token representations x_t. In the standard setting these token representations are initialized to pre-trained word embeddings, but we also experiment with supplying pre-trained ELMo representations combined with task-specific learned parameters, which have been shown to substantially improve performance of other SRL models (Peters et al., 2018). For experiments with gold predicates, we concatenate a predicate indicator embedding p_t following previous work (He et al., 2017).

We project[2] these input embeddings to a representation that is the same size as the output of the self-attention layers. We then add a positional encoding vector computed as a deterministic sinusoidal function of t, since the self-attention has no innate notion of token position.

We feed this token representation as input to a series of J residual multi-head self-attention layers with feed-forward connections. Denoting the jth self-attention layer as $T^{(j)}(\cdot)$, the output of that layer $s_t^{(j)}$, and $LN(\cdot)$ layer normalization, the following recurrence applied to initial input $c_t^{(p)}$:

$$s_t^{(j)} = LN(s_t^{(j-1)} + T^{(j)}(s_t^{(j-1)})) \qquad (1)$$

gives our final token representations $s_t^{(j)}$. Each $T^{(j)}(\cdot)$ consists of: (a) multi-head self-attention and (b) a feed-forward projection.

The multi-head self attention consists of H attention heads, each of which learns a distinct attention function to attend to all of the tokens in the sequence. This self-attention is performed for each token for each head, and the results of the H self-attentions are concatenated to form the final self-attended representation for each token.

Specifically, consider the matrix $S^{(j-1)}$ of T token representations at layer $j-1$. For each attention head h, we project this matrix into distinct key, value and query representations $K_h^{(j)}$, $V_h^{(j)}$ and $Q_h^{(j)}$ of dimensions $T \times d_k$, $T \times d_q$, and $T \times d_v$, respectively. We can then multiply $Q_h^{(j)}$ by $K_h^{(j)}$ to obtain a $T \times T$ matrix of attention weights $A_h^{(j)}$ between each pair of tokens in the sentence. Following Vaswani et al. (2017) we perform scaled dot-product attention: We scale the weights by the inverse square root of their embedding dimension

[2] All linear projections include bias terms, which we omit in this exposition for the sake of clarity.

and normalize with the softmax function to produce a distinct distribution for each token over all the tokens in the sentence:

$$A_h^{(j)} = \text{softmax}(d_k^{-0.5} Q_h^{(j)} {K_h^{(j)}}^T) \quad (2)$$

These attention weights are then multiplied by $V_h^{(j)}$ for each token to obtain the self-attended token representations $M_h^{(j)}$:

$$M_h^{(j)} = A_h^{(j)} V_h^{(j)} \quad (3)$$

Row t of $M_h^{(j)}$, the self-attended representation for token t at layer j, is thus the weighted sum with respect to t (with weights given by $A_h^{(j)}$) over the token representations in $V_h^{(j)}$.

The outputs of all attention heads for each token are concatenated, and this representation is passed to the feed-forward layer, which consists of two linear projections each followed by leaky ReLU activations (Maas et al., 2013). We add the output of the feed-forward to the initial representation and apply layer normalization to give the final output of self-attention layer j, as in Eqn. 1.

2.2 Syntactically-informed self-attention

Typically, neural attention mechanisms are left on their own to learn to attend to relevant inputs. Instead, we propose training the self-attention to attend to specific tokens corresponding to the syntactic structure of the sentence as a mechanism for passing linguistic knowledge to later layers.

Specifically, we replace one attention head with the deep bi-affine model of Dozat and Manning (2017), trained to predict syntactic dependencies. Let A_{parse} be the parse attention weights, at layer i. Its input is the matrix of token representations $S^{(i-1)}$. As with the other attention heads, we project $S^{(i-1)}$ into key, value and query representations, denoted K_{parse}, Q_{parse}, V_{parse}. Here the key and query projections correspond to *parent* and *dependent* representations of the tokens, and we allow their dimensions to differ from the rest of the attention heads to more closely follow the implementation of Dozat and Manning (2017). Unlike the other attention heads which use a dot product to score key-query pairs, we score the compatibility between K_{parse} and Q_{parse} using a bi-affine operator U_{heads} to obtain attention weights:

$$A_{parse} = \text{softmax}(Q_{parse} U_{heads} K_{parse}^T) \quad (4)$$

These attention weights are used to compose a weighted average of the value representations V_{parse} as in the other attention heads.

We apply auxiliary supervision at this attention head to encourage it to attend to each token's parent in a syntactic dependency tree, and to encode information about the token's dependency label. Denoting the attention weight from token t to a candidate head q as $A_{parse}[t, q]$, we model the probability of token t having parent q as:

$$P(q = \text{head}(t) \mid \mathcal{X}) = A_{parse}[t, q] \quad (5)$$

using the attention weights $A_{parse}[t]$ as the distribution over possible heads for token t. We define the root token as having a self-loop. This attention head thus emits a directed graph[3] where each token's parent is the token to which the attention A_{parse} assigns the highest weight.

We also predict dependency labels using per-class bi-affine operations between parent and dependent representations Q_{parse} and K_{parse} to produce per-label scores, with locally normalized probabilities over dependency labels y_t^{dep} given by the softmax function. We refer the reader to Dozat and Manning (2017) for more details.

This attention head now becomes an oracle for syntax, denoted $\mathcal{P}$, providing a dependency parse to downstream layers. This model not only predicts its own dependency arcs, but allows for the injection of auxiliary parse information at test time by simply setting A_{parse} to the parse parents produced by e.g. a state-of-the-art parser. In this way, our model can benefit from improved, external parsing models without re-training. Unlike typical multi-task models, ours maintains the ability to leverage external syntactic information.

2.3 Multi-task learning

We also share the parameters of lower layers in our model to predict POS tags and predicates. Following He et al. (2017), we focus on the end-to-end setting, where predicates must be predicted on-the-fly. Since we also train our model to predict syntactic dependencies, it is beneficial to give the model knowledge of POS information. While much previous work employs a pipelined approach to both POS tagging for dependency parsing and predicate detection for SRL, we take a multi-task learning (MTL) approach (Caruana,

[3] Usually the head emits a tree, but we do not enforce it here.

1993), sharing the parameters of earlier layers in our SRL model with a joint POS and predicate detection objective. Since POS is a strong predictor of predicates[4] and the complexity of training a multi-task model increases with the number of tasks, we combine POS tagging and predicate detection into a joint label space: For each POS tag TAG which is observed co-occurring with a predicate, we add a label of the form TAG:PREDICATE.

Specifically, we feed the representation $s_t^{(r)}$ from a layer r preceding the syntactically-informed layer p to a linear classifier to produce per-class scores r_t for token t. We compute locally-normalized probabilities using the softmax function: $P(y_t^{prp} \mid \mathcal{X}) \propto \exp(r_t)$, where y_t^{prp} is a label in the joint space.

2.4 Predicting semantic roles

Our final goal is to predict semantic roles for each predicate in the sequence. We score each predicate against each token in the sequence using a bilinear operation, producing per-label scores for each token for each predicate, with predicates and syntax determined by oracles $\mathcal{V}$ and $\mathcal{P}$.

First, we project each token representation $s_t^{(J)}$ to a predicate-specific representation s_t^{pred} and a role-specific representation s_t^{role}. We then provide these representations to a bilinear transformation U for scoring. So, the role label scores s_{ft} for the token at index t with respect to the predicate at index f (i.e. token t and frame f) are given by:

$$s_{ft} = (s_f^{pred})^T U s_t^{role} \tag{6}$$

which can be computed in parallel across all semantic frames in an entire minibatch. We calculate a locally normalized distribution over role labels for token t in frame f using the softmax function: $P(y_{ft}^{role} \mid \mathcal{P}, \mathcal{V}, \mathcal{X}) \propto \exp(s_{ft})$.

At test time, we perform constrained decoding using the Viterbi algorithm to emit valid sequences of BIO tags, using unary scores s_{ft} and the transition probabilities given by the training data.

2.5 Training

We maximize the sum of the likelihoods of the individual tasks. In order to maximize our model's ability to leverage syntax, during training we clamp $\mathcal{P}$ to the gold parse ($\mathcal{P}_G$) and $\mathcal{V}$ to gold predicates $\mathcal{V}_G$ when passing parse and predicate representations to later layers, whereas syntactic head prediction and joint predicate/POS prediction are conditioned only on the input sequence $\mathcal{X}$. The overall objective is thus:

$$\frac{1}{T}\sum_{t=1}^{T}\Big[\sum_{f=1}^{F}\log P(y_{ft}^{role} \mid \mathcal{P}_G, \mathcal{V}_G, \mathcal{X}) + \log P(y_t^{prp} \mid \mathcal{X}) + \lambda_1 \log P(\text{head}(t) \mid \mathcal{X}) + \lambda_2 \log P(y_t^{dep} \mid \mathcal{P}_G, \mathcal{X})\Big] \tag{7}$$

where λ_1 and λ_2 are penalties on the syntactic attention loss.

We train the model using Nadam (Dozat, 2016) SGD combined with the learning rate schedule in Vaswani et al. (2017). In addition to MTL, we regularize our model using dropout (Srivastava et al., 2014). We use gradient clipping to avoid exploding gradients (Bengio et al., 1994; Pascanu et al., 2013). Additional details on optimization and hyperparameters are included in Appendix A.

3 Related work

Early approaches to SRL (Pradhan et al., 2005; Surdeanu et al., 2007; Johansson and Nugues, 2008; Toutanova et al., 2008) focused on developing rich sets of linguistic features as input to a linear model, often combined with complex constrained inference e.g. with an ILP (Punyakanok et al., 2008). Täckström et al. (2015) showed that constraints could be enforced more efficiently using a clever dynamic program for exact inference. Sutton and McCallum (2005) modeled syntactic parsing and SRL jointly, and Lewis et al. (2015) jointly modeled SRL and CCG parsing.

Collobert et al. (2011) were among the first to use a neural network model for SRL, a CNN over word embeddings which failed to out-perform non-neural models. FitzGerald et al. (2015) successfully employed neural networks by embedding lexicalized features and providing them as factors in the model of Täckström et al. (2015).

More recent neural models are syntax-free. Zhou and Xu (2015), Marcheggiani et al. (2017) and He et al. (2017) all use variants of deep LSTMs with constrained decoding, while Tan et al. (2018) apply self-attention to obtain state-of-the-art SRL with gold predicates. Like this work, He et al. (2017) present end-to-end experiments, predicting predicates using an LSTM, and He et al.

[4] All predicates in CoNLL-2005 are verbs; CoNLL-2012 includes some nominal predicates.

(2018) jointly predict SRL spans and predicates in a model based on that of Lee et al. (2017), obtaining state-of-the-art predicted predicate SRL. Concurrent to this work, Peters et al. (2018) and He et al. (2018) report significant gains on PropBank SRL by training a wide LSTM language model and using a task-specific transformation of its hidden representations (ELMo) as a deep, and computationally expensive, alternative to typical word embeddings. We find that LISA obtains further accuracy increases when provided with ELMo word representations, especially on out-of-domain data.

Some work has incorporated syntax into neural models for SRL. Roth and Lapata (2016) incorporate syntax by embedding dependency paths, and similarly Marcheggiani and Titov (2017) encode syntax using a graph CNN over a predicted syntax tree, out-performing models without syntax on CoNLL-2009. These works are limited to incorporating partial dependency paths between tokens whereas our technique incorporates the entire parse. Additionally, Marcheggiani and Titov (2017) report that their model does not out-perform syntax-free models on out-of-domain data, a setting in which our technique excels.

MTL (Caruana, 1993) is popular in NLP, and others have proposed MTL models which incorporate subsets of the tasks we do (Collobert et al., 2011; Zhang and Weiss, 2016; Hashimoto et al., 2017; Peng et al., 2017; Swayamdipta et al., 2017), and we build off work that investigates where and when to combine different tasks to achieve the best results (Søgaard and Goldberg, 2016; Bingel and Søgaard, 2017; Alonso and Plank, 2017). Our specific method of incorporating supervision into self-attention is most similar to the concurrent work of Liu and Lapata (2018), who use edge marginals produced by the matrix-tree algorithm as attention weights for document classification and natural language inference.

The question of training on gold versus predicted labels is closely related to learning to search (Daumé III et al., 2009; Ross et al., 2011; Chang et al., 2015) and scheduled sampling (Bengio et al., 2015), with applications in NLP to sequence labeling and transition-based parsing (Choi and Palmer, 2011; Goldberg and Nivre, 2012; Ballesteros et al., 2016). Our approach may be interpreted as an extension of teacher forcing (Williams and Zipser, 1989) to MTL. We leave exploration of more advanced scheduled sampling techniques to future work.

4 Experimental results

We present results on the CoNLL-2005 shared task (Carreras and Màrquez, 2005) and the CoNLL-2012 English subset of OntoNotes 5.0 (Pradhan et al., 2013), achieving state-of-the-art results for a single model with predicted predicates on both corpora. We experiment with both standard pre-trained GloVe word embeddings (Pennington et al., 2014) and pre-trained ELMo representations with fine-tuned task-specific parameters (Peters et al., 2018) in order to best compare to prior work. Hyperparameters that resulted in the best performance on the validation set were selected via a small grid search, and models were trained for a maximum of 4 days on one TitanX GPU using early stopping on the validation set. We convert constituencies to dependencies using the Stanford head rules v3.5 (de Marneffe and Manning, 2008). A detailed description of hyperparameter settings and data pre-processing can be found in Appendix A.

We compare our **LISA** models to four strong baselines: For experiments using predicted predicates, we compare to He et al. (2018) and the ensemble model (**PoE**) from He et al. (2017), as well as a version of our own self-attention model which does not incorporate syntactic information (**SA**). To compare to more prior work, we present additional results on CoNLL-2005 with models given gold predicates at test time. In these experiments we also compare to Tan et al. (2018), the previous state-of-the art SRL model using gold predicates and standard embeddings.

We demonstrate that our models benefit from injecting state-of-the-art predicted parses at test time (**+D&M**) by fixing the attention to parses predicted by Dozat and Manning (2017), the winner of the 2017 CoNLL shared task (Zeman et al., 2017) which we re-train using ELMo embeddings. In all cases, using these parses at test time improves performance.

We also evaluate our model using the gold syntactic parse at test time (**+Gold**), to provide an upper bound for the benefit that syntax could have for SRL using LISA. These experiments show that despite LISA's strong performance, there remains substantial room for improvement. In §4.3 we perform further analysis comparing SRL models using gold and predicted parses.

	Dev			WSJ Test			Brown Test		
GloVe	P	R	F1	P	R	F1	P	R	F1
He et al. (2017) PoE	81.8	81.2	81.5	82.0	83.4	82.7	69.7	70.5	70.1
He et al. (2018)	81.3	81.9	81.6	81.2	83.9	82.5	69.7	71.9	70.8
SA	83.52	81.28	82.39	84.17	83.28	83.72	72.98	70.1	71.51
LISA	83.1	81.39	82.24	84.07	83.16	83.61	73.32	70.56	71.91
+D&M	**84.59**	**82.59**	**83.58**	**85.53**	**84.45**	**84.99**	**75.8**	**73.54**	**74.66**
+*Gold*	*87.91*	*85.73*	*86.81*	—	—	—	—	—	—
ELMo									
He et al. (2018)	84.9	**85.7**	85.3	84.8	**87.2**	86.0	73.9	**78.4**	76.1
SA	85.78	84.74	85.26	86.21	85.98	86.09	77.1	75.61	76.35
LISA	**86.07**	84.64	**85.35**	86.69	86.42	86.55	78.95	77.17	78.05
+D&M	85.83	84.51	85.17	**87.13**	86.67	**86.90**	**79.02**	77.49	**78.25**
+*Gold*	*88.51*	*86.77*	*87.63*	—	—	—	—	—	—

Table 1: Precision, recall and F1 on the CoNLL-2005 development and test sets.

WSJ Test	P	R	F1
He et al. (2018)	84.2	83.7	83.9
Tan et al. (2018)	84.5	85.2	84.8
SA	84.7	84.24	84.47
LISA	84.72	84.57	84.64
+D&M	**86.02**	**86.05**	**86.04**

Brown Test	P	R	F1
He et al. (2018)	74.2	73.1	73.7
Tan et al. (2018)	73.5	74.6	74.1
SA	73.89	72.39	73.13
LISA	74.77	74.32	74.55
+D&M	**76.65**	**76.44**	**76.54**

Table 2: Precision, recall and F1 on CoNLL-2005 with gold predicates.

4.1 Semantic role labeling

Table 1 lists precision, recall and F1 on the CoNLL-2005 development and test sets using predicted predicates. For models using GloVe embeddings, our syntax-free SA model already achieves a new state-of-the-art by jointly predicting predicates, POS and SRL. LISA with its own parses performs comparably to SA, but when supplied with D&M parses LISA out-performs the previous state-of-the-art by 2.5 F1 points. On the out-of-domain Brown test set, LISA also performs comparably to its syntax-free counterpart with its own parses, but with D&M parses LISA performs exceptionally well, more than 3.5 F1 points higher than He et al. (2018). Incorporating ELMo embeddings improves all scores. The gap in SRL F1 between models using LISA and D&M parses is smaller due to LISA's improved parsing accuracy (see §4.2), but LISA with D&M parses still achieves the highest F1: nearly 1.0 absolute F1 higher than the previous state-of-the art on WSJ, and more than 2.0 F1 higher on Brown. In both settings LISA leverages domain-agnostic syntactic information rather than over-fitting to the newswire training data which leads to high performance even on out-of-domain text.

To compare to more prior work we also evaluate our models in the artificial setting where gold predicates are provided at test time. For fair comparison we use GloVe embeddings, provide predicate indicator embeddings on the input and re-encode the sequence relative to each gold predicate. Here LISA still excels: with D&M parses, LISA out-performs the previous state-of-the-art by more than 2 F1 on both WSJ and Brown.

Table 3 reports precision, recall and F1 on the CoNLL-2012 test set. We observe performance similar to that observed on ConLL-2005: Using GloVe embeddings our SA baseline already out-performs He et al. (2018) by nearly 1.5 F1. With its own parses, LISA slightly under-performs our syntax-free model, but when provided with stronger D&M parses LISA outperforms the state-of-the-art by more than 2.5 F1. Like CoNLL-2005, ELMo representations improve all models and close the F1 gap between models supplied with LISA and D&M parses. On this dataset ELMo also substantially narrows the

Dev	P	R	F1
	GloVe		
He et al. (2018)	79.2	79.7	79.4
SA	82.32	79.76	81.02
LISA	81.77	79.65	80.70
+D&M	**82.97**	**81.14**	**82.05**
+*Gold*	*87.57*	*85.32*	*86.43*
	ELMo		
He et al. (2018)	82.1	**84.0**	83.0
SA	84.35	82.14	83.23
LISA	**84.19**	82.56	**83.37**
+D&M	84.09	82.65	83.36
+*Gold*	*88.22*	*86.53*	*87.36*

Test	P	R	F1
	GloVe		
He et al. (2018)	79.4	80.1	79.8
SA	82.55	80.02	81.26
LISA	81.86	79.56	80.70
+D&M	**83.3**	**81.38**	**82.33**
	ELMo		
He et al. (2018)	81.9	**84.0**	82.9
SA	**84.39**	82.21	83.28
LISA	83.97	82.29	83.12
+D&M	84.14	82.64	**83.38**

Table 3: Precision, recall and F1 on the CoNLL-2012 development and test sets. Italics indicate a synthetic upper bound obtained by providing a gold parse at test time.

difference between models with- and without syntactic information. This suggests that for this challenging dataset, ELMo already encodes much of the information available in the D&M parses. Yet, higher accuracy parses could still yield improvements since providing gold parses increases F1 by 4 points even with ELMo embeddings.

4.2 Parsing, POS and predicate detection

We first report the labeled and unlabeled attachment scores (LAS, UAS) of our parsing models on the CoNLL-2005 and 2012 test sets (Table 4) with GloVe (G) and ELMo (E) embeddings. D&M achieves the best scores. Still, LISA's GloVe UAS is comparable to popular off-the-shelf dependency parsers such as spaCy,[5] and with ELMo embeddings comparable to the standalone D&M parser. The difference in parse accuracy between LISA$_G$ and D&M likely explains the large increase in SRL performance we see from decoding with D&M parses in that setting.

[5]spaCy reports 94.48 UAS on WSJ using Stanford dependencies v3.3: https://spacy.io/usage/facts-figures

Data	Model	POS	UAS	LAS
WSJ	D&M$_E$	—	96.48	94.40
	LISA$_G$	96.92	94.92	91.87
	LISA$_E$	97.80	96.28	93.65
Brown	D&M$_E$	—	92.56	88.52
	LISA$_G$	94.26	90.31	85.82
	LISA$_E$	95.77	93.36	88.75
CoNLL-12	D&M$_E$	—	94.99	92.59
	LISA$_G$	96.81	93.35	90.42
	LISA$_E$	98.11	94.84	92.23

Table 4: Parsing (labeled and unlabeled attachment) and POS accuracies attained by the models used in SRL experiments on test datasets. Subscript G denotes GloVe and E ELMo embeddings.

	Model	P	R	F1
WSJ	He et al. (2017)	94.5	98.5	96.4
	LISA	98.9	97.9	98.4
Brown	He et al. (2017)	89.3	95.7	92.4
	LISA	95.5	91.9	93.7
CoNLL-12	LISA	99.8	94.7	97.2

Table 5: Predicate detection precision, recall and F1 on CoNLL-2005 and CoNLL-2012 test sets.

In Table 5 we present predicate detection precision, recall and F1 on the CoNLL-2005 and 2012 test sets. SA and LISA with and without ELMo attain comparable scores so we report only LISA+GloVe. We compare to He et al. (2017) on CoNLL-2005, the only cited work reporting comparable predicate detection F1. LISA attains high predicate detection scores, above 97 F1, on both in-domain datasets, and out-performs He et al. (2017) by 1.5-2 F1 points even on the out-of-domain Brown test set, suggesting that multi-task learning works well for SRL predicate detection.

4.3 Analysis

First we assess SRL F1 on sentences divided by parse accuracy. Table 6 lists average SRL F1 (across sentences) for the four conditions of LISA and D&M parses being correct or not (**L**±, **D**±). Both parsers are correct on 26% of sentences.

	L+/D+	L–/D+	L+/D–	L–/D–
Proportion	26%	12%	4%	56%
SA	79.29	75.14	75.97	75.08
LISA	79.51	74.33	79.69	75.00
+D&M	79.03	76.96	77.73	76.52
+Gold	*79.61*	*78.38*	*81.41*	*80.47*

Table 6: Average SRL F1 on CoNLL-2005 for sentences where LISA (L) and D&M (D) parses were completely correct (+) or incorrect (–).

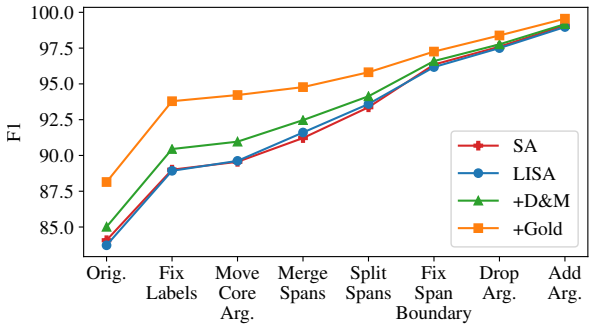


Figure 3: Performance of CoNLL-2005 models after performing corrections from He et al. (2017).

Here there is little difference between any of the models, with LISA models tending to perform slightly better than SA. Both parsers make mistakes on the majority of sentences (57%), difficult sentences where SA also performs the worst. These examples are likely where gold and D&M parses improve the most over other models in overall F1: Though both parsers fail to correctly parse the entire sentence, the D&M parser is less wrong (87.5 vs. 85.7 average LAS), leading to higher SRL F1 by about 1.5 average F1.

Following He et al. (2017), we next apply a series of corrections to model predictions in order to understand which error types the gold parse resolves: e.g. *Fix Labels* fixes labels on spans matching gold boundaries, and *Merge Spans* merges adjacent predicted spans into a gold span.[6]

In Figure 3 we see that much of the performance gap between the gold and predicted parses is due to span boundary errors (*Merge Spans*, *Split Spans* and *Fix Span Boundary*), which supports the hypothesis proposed by He et al. (2017) that incorporating syntax could be particularly helpful for resolving these errors. He et al. (2017) also point out

[6]Refer to He et al. (2017) for a detailed explanation of the different error types.

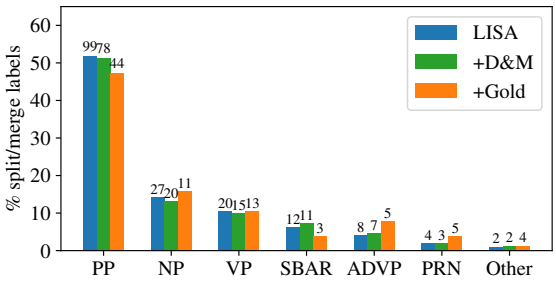


Figure 4: Percent and count of split/merge corrections performed in Figure 3, by phrase type.

that these errors are due mainly to prepositional phrase (PP) attachment mistakes. We also find this to be the case: Figure 4 shows a breakdown of split/merge corrections by phrase type. Though the number of corrections decreases substantially across phrase types, the proportion of corrections attributed to PPs remains the same (approx. 50%) even after providing the correct PP attachment to the model, indicating that PP span boundary mistakes are a fundamental difficulty for SRL.

5 Conclusion

We present linguistically-informed self-attention: a multi-task neural network model that effectively incorporates rich linguistic information for semantic role labeling. LISA out-performs the state-of-the-art on two benchmark SRL datasets, including out-of-domain. Future work will explore improving LISA's parsing accuracy, developing better training techniques and adapting to more tasks.

Acknowledgments

We are grateful to Luheng He for helpful discussions and code, Timothy Dozat for sharing his code, and to the NLP reading groups at Google and UMass and the anonymous reviewers for feedback on drafts of this work. This work was supported in part by an IBM PhD Fellowship Award to E.S., in part by the Center for Intelligent Information Retrieval, and in part by the National Science Foundation under Grant Nos. DMR-1534431 and IIS-1514053. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2015. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow.org*.

Héctor Martínez Alonso and Barbara Plank. 2017. When is multitask learning effective? semantic sequence prediction under varying data conditions. In *EACL*.

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A. Smith. 2016. Training with exploration improves a greedy stack lstm parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2005–2010.

Marzieh Bazrafshan and Daniel Gildea. 2013. Semantic roles for string to tree machine translation. In *ACL*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Brad Huang, Christopher D. Manning, Abby Vander Linden, Brittany Harding, and Peter Clark. 2014. Modeling biological processes for reading comprehension. In *EMNLP*.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *EACL*.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *CoNLL*.

Rich Caruana. 1993. Multitask learning: a knowledge-based source of inductive bias. In *ICML*.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. 2015. Learning to search better than your teacher. In *ICML*.

Yun-Nung Chen, William Yang Wang, and Alexander I Rudnicky. 2013. Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing. In *Proc. of ASRU-IEEE*.

Jinho D. Choi and Martha Palmer. 2011. Getting the most out of transition-based dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: short papers*, pages 687–692.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75(3):297–325.

Timothy Dozat. 2016. Incorporating nesterov momentum into adam. In *ICLR Workshop track*.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *ICLR*.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 960–970.

W. N. Francis and H. Kučera. 1964. Manual of information to accompany a standard corpus of present-day edited american english, for use with digital computers. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of COLING 2012: Technical Papers*, pages 959–976.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Conference on Empirical Methods in Natural Language Processing*.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *ACL*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Richard Johansson and Pierre Nugues. 2008. Dependency-based semantic role labeling of propbank. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 69–78.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations (ICLR)*, San Diego, California, USA.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *EMNLP*.

Beth Levin. 1993. *English verb classes and alternations: A preliminary investigation*. University of Chicago press.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint A* CCG Parsing and Semantic Role Labeling. In *EMNLP*.

Ding Liu and Daniel Gildea. 2010. Semantic role features for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*.

Yang Liu and Mirella Lapata. 2018. Learning structured text representations. *Transactions of the Association for Computational Linguistics*, 6:63–75.

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, volume 30.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *CoNLL*.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn TreeBank. *Computational Linguistics – Special issue on using large corpora: II*, 19(2):313–330.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The stanford typed dependencies representation. In *COLING 2008 Workshop on Cross-framework and Cross-domain Parser Evaluation*.

Yurii Nesterov. 1983. A method of solving a convex programming problem with convergence rate $o(1/k^2)$. volume 27, pages 372–376.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1).

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30 th International Conference on Machine Learning*.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of the Association for Computational Linguistics 43rd annual meeting (ACL)*.

Vasin Punyakanok, Dan Roth, and Wen-Tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1192–1202.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 231–235.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere R. Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*, 29:105–151.

Charles Sutton and Andrew McCallum. 2005. Joint parsing and semantic role labeling. In *CoNLL*.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A. Smith. 2017. Frame-semantic parsing with softmax-margin segmental rnns and a syntactic scaffold. In *arXiv:1706.09528*.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *TACL*, 3:29–41.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *AAAI*.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Gokhan Tur, Dilek Hakkani-Tür, and Ananlada Chotimongkol. 2005. Semi-supervised learning for spoken language understanding using semantic role labeling. In *ASRU*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *31st Conference on Neural Information Processing Systems (NIPS)*.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *ACL*.

R. J. Williams and D. Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, et al. 2017. Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1557–1566. Association for Computational Linguistics.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

CoNLL-2005	Greedy F1	Viterbi F1	Δ F1
LISA	81.99	82.24	+0.25
+D&M	83.37	83.58	+0.21
+Gold	*86.57*	*86.81*	*+0.24*

CoNLL-2012	Greedy F1	Viterbi F1	Δ F1
LISA	80.11	80.70	+0.59
+D&M	81.55	82.05	+0.50
+Gold	*85.94*	*86.43*	*+0.49*

Table 7: Comparison of development F1 scores with and without Viterbi decoding at test time.

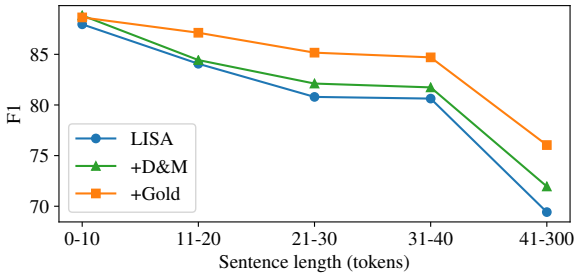


Figure 5: F1 score as a function of sentence length.

A Supplemental Material

A.1 Supplemental analysis

Here we continue the analysis from §4.3. All experiments in this section are performed on CoNLL-2005 development data unless stated otherwise.

First, we compare the impact of Viterbi decoding with LISA, D&M, and gold syntax trees (Table 7), finding the same trends across both datasets. We find that Viterbi has nearly the same impact for LISA, D&M and gold parses: Gold parses provide little improvement over predicted parses in terms of BIO label consistency.

We also assess SRL F1 as a function of sentence length and distance from span to predicate. In Figure 5 we see that providing LISA with gold parses is particularly helpful for sentences longer than 10 tokens. This likely directly follows from the tendency of syntactic parsers to perform worse on longer sentences. With respect to distance between arguments and predicates, (Figure 6), we do not observe this same trend, with all distances performing better with better parses, and especially gold.

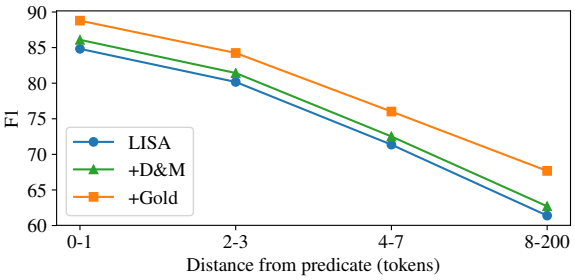


Figure 6: CoNLL-2005 F1 score as a function of the distance of the predicate from the argument span.

	L+/D+	L-/D+	L+/D-	L-/D-
Proportion	37%	10%	4%	49%
SA	76.12	75.97	82.25	65.78
LISA	76.37	72.38	85.50	65.10
+D&M	76.33	79.65	75.62	66.55
+Gold	*76.71*	*80.67*	*86.03*	*72.22*

Table 8: Average SRL F1 on CoNLL-2012 for sentences where LISA (L) and D&M (D) parses were correct (+) or incorrect (-).

A.2 Supplemental results

Due to space constraints in the main paper we list additional experimental results here. Table 9 lists development scores on the CoNLL-2005 dataset with predicted predicates, which follow the same trends as the test data.

A.3 Data and pre-processing details

We initialize word embeddings with 100d pre-trained GloVe embeddings trained on 6 billion tokens of Wikipedia and Gigaword (Pennington et al., 2014). We evaluate the SRL performance of our models using the `srl-eval.pl` script

WSJ Dev	P	R	F1
He et al. (2018)	84.2	83.7	83.9
Tan et al. (2018)	82.6	83.6	83.1
SA	83.12	82.81	82.97
LISA	83.6	83.74	83.67
+D&M	**85.04**	**85.51**	**85.27**
+Gold	*89.11*	*89.38*	*89.25*

Table 9: Precision, recall and F1 on the CoNLL-2005 development set with gold predicates.

provided by the CoNLL-2005 shared task,[7] which computes segment-level precision, recall and F1 score. We also report the predicate detection scores output by this script. We evaluate parsing using the `eval.pl` CoNLL script, which excludes punctuation.

We train distinct D&M parsers for CoNLL-2005 and CoNLL-2012. Our D&M parsers are trained and validated using the same SRL data splits, except that for CoNLL-2005 section 22 is used for development (rather than 24), as this section is typically used for validation in PTB parsing. We use Stanford dependencies v3.5 (de Marneffe and Manning, 2008) and POS tags from the Stanford CoreNLP `left3words` model (Toutanova et al., 2003). We use the pre-trained ELMo models[8] and learn task-specific combinations of the ELMo representations which are provided as input instead of GloVe embeddings to the D&M parser with otherwise default settings.

A.3.1 CoNLL-2012

We follow the CoNLL-2012 split used by He et al. (2018) to evaluate our models, which uses the annotations from here[9] but the subset of those documents from the CoNLL-2012 co-reference split described here[10] (Pradhan et al., 2013). This dataset is drawn from seven domains: newswire, web, broadcast news and conversation, magazines, telephone conversations, and text from the bible. The text is annotated with gold part-of-speech, syntactic constituencies, named entities, word sense, speaker, co-reference and semantic role labels based on the PropBank guidelines (Palmer et al., 2005). Propositions may be verbal or nominal, and there are 41 distinct semantic role labels, excluding continuation roles and including the predicate. We convert the semantic proposition and role segmentations to BIO boundary-encoded tags, resulting in 129 distinct BIO-encoded tags (including continuation roles).

A.3.2 CoNLL-2005

The CoNLL-2005 data (Carreras and Màrquez, 2005) is based on the original PropBank corpus (Palmer et al., 2005), which labels the Wall Street Journal portion of the Penn TreeBank corpus (PTB) (Marcus et al., 1993) with predicate-argument structures, plus a challenging out-of-domain test set derived from the Brown corpus (Francis and Kučera, 1964). This dataset contains only verbal predicates, though some are multi-word verbs, and 28 distinct role label types. We obtain 105 SRL labels including continuations after encoding predicate argument segment boundaries with BIO tags.

A.4 Optimization and hyperparameters

We train the model using the Nadam (Dozat, 2016) algorithm for adaptive stochastic gradient descent (SGD), which combines Adam (Kingma and Ba, 2015) SGD with Nesterov momentum (Nesterov, 1983). We additionally vary the learning rate lr as a function of an initial learning rate lr_0 and the current training step $step$ as described in Vaswani et al. (2017) using the following function:

$$lr = lr_0 \cdot \min(step^{-0.5}, step \cdot warm^{-1.5}) \quad (8)$$

which increases the learning rate linearly for the first $warm$ training steps, then decays it proportionally to the inverse square root of the step number. We found this learning rate schedule essential for training the self-attention model. We only update optimization moving-average accumulators for parameters which receive gradient updates at a given step.[11]

In all of our experiments we used initial learning rate 0.04, $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 1 \times 10^{-12}$ and dropout rates of 0.1 everywhere. We use 10 or 12 self-attention layers made up of 8 attention heads each with embedding dimension 25, with 800d feed-forward projections. In the syntactically-informed attention head, Q_{parse} has dimension 500 and K_{parse} has dimension 100. The size of $predicate$ and $role$ representations and the representation used for joint part-of-speech/predicate classification is 200. We train with $warm = 8000$ warmup steps and clip gradient norms to 1. We use batches of approximately 5000 tokens.

[7] http://www.lsi.upc.es/~srlconll/srl-eval.pl

[8] https://github.com/allenai/bilm-tf

[9] http://cemantix.org/data/ontonotes.html

[10] http://conll.cemantix.org/2012/data.html

[11] Also known as *lazy* or *sparse* optimizer updates.

What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties

Alexis Conneau
Facebook AI Research
Université Le Mans
aconneau@fb.com

German Kruszewski
Facebook AI Research
germank@fb.com

Guillaume Lample
Facebook AI Research
Sorbonne Universités
glample@fb.com

Loïc Barrault
Université Le Mans
loic.barrault@univ-lemans.fr

Marco Baroni
Facebook AI Research
mbaroni@fb.com

Abstract

Although much effort has recently been devoted to training high-quality sentence embeddings, we still have a poor understanding of what they are capturing. "Downstream" tasks, often based on sentence classification, are commonly used to evaluate the quality of sentence representations. The complexity of the tasks makes it however difficult to infer what kind of information is present in the representations. We introduce here 10 probing tasks designed to capture simple linguistic features of sentences, and we use them to study embeddings generated by three different encoders trained in eight distinct ways, uncovering intriguing properties of both encoders and training methods.

1 Introduction

Despite Ray Mooney's quip that you cannot cram the meaning of a whole %&!$# sentence into a single $&!#* vector, sentence embedding methods have achieved impressive results in tasks ranging from machine translation (Sutskever et al., 2014; Cho et al., 2014) to entailment detection (Williams et al., 2018), spurring the quest for "universal embeddings" trained once and used in a variety of applications (e.g., Kiros et al., 2015; Conneau et al., 2017; Subramanian et al., 2018). Positive results on concrete problems suggest that embeddings capture important linguistic properties of sentences. However, real-life "downstream" tasks require complex forms of inference, making it difficult to pinpoint the information a model is relying upon. Impressive as it might be that a system can tell that the sentence "A movie that doesn't aim too high, but it doesn't need to" (Pang and Lee, 2004) expresses a subjective viewpoint, it is hard to tell *how* the system (or even a human) comes to this conclusion. Complex tasks can also carry hidden biases that models might lock onto (Jabri et al., 2016). For example, Lai and Hockenmaier (2014) show that the simple heuristic of checking for explicit negation words leads to good accuracy in the SICK sentence entailment task.

Model introspection techniques have been applied to sentence encoders in order to gain a better understanding of which properties of the input sentences their embeddings retain (see Section 5). However, these techniques often depend on the specifics of an encoder architecture, and consequently cannot be used to compare different methods. Shi et al. (2016) and Adi et al. (2017) introduced a more general approach, relying on the notion of what we will call *probing tasks*. A probing task is a classification problem that focuses on simple linguistic properties of sentences. For example, one such task might require to categorize sentences by the tense of their main verb. Given an encoder (e.g., an LSTM) pre-trained on a certain task (e.g., machine translation), we use the sentence embeddings it produces to train the tense classifier (without further embedding tuning). If the classifier succeeds, it means that the pre-trained encoder is storing readable tense information into the embeddings it creates. Note that: (i) The probing task asks a simple question, minimizing interpretability problems. (ii) Because of their simplicity, it is easier to control for biases in probing tasks than in downstream tasks. (iii) The probing task methodology is agnostic with respect to the encoder architecture, as long as it produces a vector representation of sentences.

We greatly extend earlier work on probing tasks as follows. First, we introduce a larger set of probing tasks (10 in total), organized by the type of linguistic properties they probe. Second, we systematize the probing task methodology, controlling for

a number of possible nuisance factors, and framing all tasks so that they only require single sentence representations as input, for maximum generality and to ease result interpretation. Third, we use our probing tasks to explore a wide range of state-of-the-art encoding architectures and training methods, and further relate probing and downstream task performance. Finally, we are publicly releasing our probing data sets and tools, hoping they will become a standard way to study the linguistic properties of sentence embeddings.[1]

2 Probing tasks

In constructing our probing benchmarks, we adopted the following criteria. First, for generality and interpretability, the task classification problem should only require single sentence embeddings as input (as opposed to, e.g., sentence and word embeddings, or multiple sentence representations). Second, it should be possible to construct large training sets in order to train parameter-rich multi-layer classifiers, in case the relevant properties are non-linearly encoded in the sentence vectors. Third, nuisance variables such as lexical cues or sentence length should be controlled for. Finally, and most importantly, we want tasks that address an interesting set of linguistic properties. We thus strove to come up with a set of tasks that, while respecting the previous constraints, probe a wide range of phenomena, from superficial properties of sentences such as which words they contain to their hierarchical structure to subtle facets of semantic acceptability. We think the current task set is reasonably representative of different linguistic domains, but we are not claiming that it is exhaustive. We expect future work to extend it.

The sentences for all our tasks are extracted from the Toronto Book Corpus (Zhu et al., 2015), more specifically from the random pre-processed portion made available by Paperno et al. (2016). We only sample sentences in the 5-to-28 word range. We parse them with the Stanford Parser (2017-06-09 version), using the pre-trained PCFG model (Klein and Manning, 2003), and we rely on the part-of-speech, constituency and dependency parsing information provided by this tool where needed. For each task, we construct training sets containing 100k sentences, and 10k-sentence validation and test sets. All sets are balanced, having an equal number of instances of each target class.

Surface information These tasks test the extent to which sentence embeddings are preserving surface properties of the sentences they encode. One can solve the surface tasks by simply looking at tokens in the input sentences: no linguistic knowledge is called for. The first task is to predict the *length* of sentences in terms of number of words (**SentLen**). Following Adi et al. (2017), we group sentences into 6 equal-width bins by length, and treat SentLen as a 6-way classification task. The *word content* (**WC**) task tests whether it is possible to recover information about the original words in the sentence from its embedding. We picked 1000 mid-frequency words from the source corpus vocabulary (the words with ranks between 2k and 3k when sorted by frequency), and sampled equal numbers of sentences that contain one and only one of these words. The task is to tell which of the 1k words a sentence contains (1k-way classification). This setup allows us to probe a sentence embedding for word content without requiring an auxiliary word embedding (as in the setup of Adi and colleagues).

Syntactic information The next batch of tasks test whether sentence embeddings are sensitive to syntactic properties of the sentences they encode. The *bigram shift* (**BShift**) task tests whether an encoder is sensitive to legal word orders. In this binary classification problem, models must distinguish intact sentences sampled from the corpus from sentences where we inverted two random adjacent words ("What *you are* doing out there?").

The *tree depth* (**TreeDepth**) task checks whether an encoder infers the hierarchical structure of sentences, and in particular whether it can group sentences by the depth of the longest path from root to any leaf. Since tree depth is naturally correlated with sentence length, we de-correlate these variables through a structured sampling procedure. In the resulting data set, tree depth values range from 5 to 12, and the task is to categorize sentences into the class corresponding to their depth (8 classes). As an example, the following is a long (22 tokens) but shallow (max depth: 5) sentence: "[$_1$ [$_2$ But right now, for the time being, my past, my fears, and my thoughts [$_3$ were [$_4$ my [$_5$business]]].]]" (the outermost brackets correspond to the ROOT and S nodes in the parse).

[1] https://github.com/facebookresearch/SentEval/tree/master/data/probing

In the top constituent task (**TopConst**), sentences must be classified in terms of the sequence of top constituents immediately below the sentence (S) node. An encoder that successfully addresses this challenge is not only capturing latent syntactic structures, but clustering them by constituent types. TopConst was introduced by Shi et al. (2016). Following them, we frame it as a 20-way classification problem: 19 classes for the most frequent top constructions, and one for all other constructions. As an example, "[Then] [very dark gray letters on a black screen] [appeared] [.]" has top constituent sequence: "ADVP NP VP .".

Note that, while we would not expect an untrained human subject to be explicitly aware of tree depth or top constituency, similar information must be implicitly computed to correctly parse sentences, and there is suggestive evidence that the brain tracks something akin to tree depth during sentence processing (Nelson et al., 2017).

Semantic information These tasks also rely on syntactic structure, but they further require some understanding of what a sentence denotes. The **Tense** task asks for the tense of the main-clause verb (VBP/VBZ forms are labeled as present, VBD as past). No target form occurs across the train/dev/test split, so that classifiers cannot rely on specific words (it is not clear that Shi and colleagues, who introduced this task, controlled for this factor). The *subject number* (**SubjNum**) task focuses on the number of the subject of the main clause (number in English is more often explicitly marked on nouns than verbs). Again, there is no target overlap across partitions. Similarly, *object number* (**ObjNum**) tests for the number of the direct object of the main clause (again, avoiding lexical overlap). To solve the previous tasks correctly, an encoder must not only capture tense and number, but also extract structural information (about the main clause and its arguments). We grouped Tense, SubjNum and ObjNum with the semantic tasks, since, at least for models that treat words as unanalyzed input units (without access to morphology), they must rely on what a sentence denotes (e.g., whether the described event took place in the past), rather than on structural/syntactic information. We recognize, however, that the boundary between syntactic and semantic tasks is somewhat arbitrary.

In the *semantic odd man out* (**SOMO**) task, we modified sentences by replacing a random noun or verb o with another noun or verb r. To make the task more challenging, the bigrams formed by the replacement with the previous and following words in the sentence have frequencies that are comparable (on a log-scale) with those of the original bigrams. That is, if the original sentence contains bigrams $w_{n-1}o$ and ow_{n+1}, the corresponding bigrams $w_{n-1}r$ and rw_{n+1} in the modified sentence will have comparable corpus frequencies. No sentence is included in both original and modified format, and no replacement is repeated across train/dev/test sets. The task of the classifier is to tell whether a sentence has been modified or not. An example modified sentence is: " No one could see this Hayes and I wanted to know if it was real or a *spoonful* (orig.: *ploy*)." Note that judging plausibility of a syntactically well-formed sentence of this sort will often require grasping rather subtle semantic factors, ranging from selectional preference to topical coherence.

The coordination inversion (**CoordInv**) benchmark contains sentences made of two coordinate clauses. In half of the sentences, we inverted the order of the clauses. The task is to tell whether a sentence is intact or modified. Sentences are balanced in terms of clause length, and no sentence appears in both original and inverted versions. As an example, original "They might be only memories, but I can still feel each one" becomes: "I can still feel each one, but they might be only memories." Often, addressing CoordInv requires an understanding of broad discourse and pragmatic factors.

Row **Hum. Eval.** of Table 2 reports human-validated "reasonable" upper bounds for all the tasks, estimated in different ways, depending on the tasks. For the surface ones, there is always a straightforward correct answer that a human annotator with enough time and patience could find. The upper bound is thus estimated at 100%. The TreeDepth, TopConst, Tense, SubjNum and ObjNum tasks depend on automated PoS and parsing annotation. In these cases, the upper bound is given by the proportion of sentences correctly annotated by the automated procedure. To estimate this quantity, one linguistically-trained author checked the annotation of 200 randomly sampled test sentences from each task. Finally, the BShift, SOMO and CoordInv manipulations can accidentally generate acceptable sentences. For

example, one modified SOMO sentence is: "He pulled out the large round *onion* (orig.: *cork*) and saw the amber balm inside.", that is arguably not more anomalous than the original. For these tasks, we ran Amazon Mechanical Turk experiments in which subjects were asked to judge whether 1k randomly sampled test sentences were acceptable or not. Reported human accuracies are based on majority voting. See Appendix for details.

3 Sentence embedding models

In this section, we present the three sentence encoders that we consider and the seven tasks on which we train them.

3.1 Sentence encoder architectures

A wide variety of neural networks encoding sentences into fixed-size representations exist. We focus here on three that have been shown to perform well on standard NLP tasks.

BiLSTM-last/max For a sequence of T words $\{w_t\}_{t=1,\dots,T}$, a bidirectional LSTM computes a set of T vectors $\{h_t\}_t$. For $t \in [1,\dots,T]$, h_t is the concatenation of a forward LSTM and a backward LSTM that read the sentences in two opposite directions. We experiment with two ways of combining the varying number of $(h_1,\dots,h_T)$ to form a fixed-size vector, either by selecting the last hidden state of h_T or by selecting the maximum value over each dimension of the hidden units. The choice of these models are motivated by their demonstrated efficiency in seq2seq (Sutskever et al., 2014) and universal sentence representation learning (Conneau et al., 2017), respectively.[2]

Gated ConvNet We also consider the non-recurrent convolutional equivalent of LSTMs, based on stacked gated temporal convolutions. Gated convolutional networks were shown to perform well as neural machine translation encoders (Gehring et al., 2017) and language modeling decoders (Dauphin et al., 2017). The encoder is composed of an input word embedding table that is augmented with positional encodings (Sukhbaatar et al., 2015), followed by a stack of temporal convolutions with small kernel size. The output of each convolutional layer is filtered by a gating mechanism, similar to the one of LSTMs. Finally, max-pooling along the temporal dimension is performed on the output feature maps of the last convolution (Collobert and Weston, 2008).

[2]We also experimented with a unidirectional LSTM, with consistently poorer results.

3.2 Training tasks

Seq2seq systems have shown strong results in machine translation (Zhou et al., 2016). They consist of an *encoder* that encodes a source sentence into a fixed-size representation, and a *decoder* which acts as a conditional language model and that generates the target sentence. We train **Neural Machine Translation** systems on three language pairs using about 2M sentences from the Europarl corpora (Koehn, 2005). We pick **English-French**, which involves two similar languages, **English-German**, involving larger syntactic differences, and **English-Finnish**, a distant pair. We also train with an **AutoEncoder** objective (Socher et al., 2011) on Europarl source English sentences. Following Vinyals et al. (2015), we train a seq2seq architecture to generate linearized grammatical parse trees (see Table 1) from source sentences (**Seq2Tree**). We use the Stanford parser to generate trees for Europarl source English sentences. We train **SkipThought** vectors (Kiros et al., 2015) by predicting the next sentence given the current one (Tang et al., 2017), on 30M sentences from the Toronto Book Corpus, excluding those in the probing sets. Finally, following Conneau et al. (2017), we train sentence encoders on **Natural Language Inference** using the concatenation of the SNLI (Bowman et al., 2015) and MultiNLI (Bowman et al., 2015) data sets (about 1M sentence pairs). In this task, a sentence encoder is trained to encode two sentences, which are fed to a classifier and whose role is to distinguish whether the sentences are contradictory, neutral or entailed. Finally, as in Conneau et al. (2017), we also include **Untrained** encoders with random weights, which act as random projections of pre-trained word embeddings.

3.3 Training details

BiLSTM encoders use 2 layers of 512 hidden units ($\sim$4M parameters), Gated ConvNet has 8 convolutional layers of 512 hidden units, kernel size 3 ($\sim$12M parameters). We use pre-trained fastText word embeddings of size 300 (Mikolov et al., 2018) without fine-tuning, to isolate the impact of encoder architectures and to handle words outside the training sets. Training task performance and further details are in Appendix.

task	source	target
AutoEncoder	I myself was out on an island in the Swedish archipelago , at Sandhamn .	I myself was out on an island in the Swedish archipelago , at Sand@ ham@ n .
NMT En-Fr	I myself was out on an island in the Swedish archipelago , at Sandhamn .	Je me trouvais ce jour là sur une île de l' archipel suédois , à Sand@ ham@ n .
NMT En-De	We really need to up our particular contribution in that regard .	Wir müssen wirklich unsere spezielle Hilfs@ leistung in dieser Hinsicht aufstocken .
NMT En-Fi	It is too early to see one system as a universal panacea and dismiss another .	Nyt on liian aikaista nostaa yksi järjestelmä jal@ usta@ lle ja antaa jollekin toiselle huono arvo@ sana .
SkipThought	the old sami was gone , and he was a different person now .	the new sami didn 't mind standing barefoot in dirty white , sans ra@ y-@ bans and without beautiful women following his every move .
Seq2Tree	Dikoya is a village in Sri Lanka .	(ROOT (S (NP NNP)NP (VP VBZ (NP (NP DT NN)NP (PP IN (NP NNP NNP)NP)PP)NP)VP .)S)ROOT

Table 1: **Source and target examples for seq2seq training tasks.**

4 Probing task experiments

Baselines Baseline and human-bound performance are reported in the top block of Table 2. **Length** is a linear classifier with sentence length as sole feature. **NB-uni-tfidf** is a Naive Bayes classifier using words' tfidf scores as features, **NB-bi-tfidf** its extension to bigrams. Finally, **BoV-fastText** derives sentence representations by averaging the fastText embeddings of the words they contain (same embeddings used as input to the encoders).[3]

Except, trivially, for Length on SentLen and the NB baselines on WC, there is a healthy gap between top baseline performance and human upper bounds. NB-uni-tfidf evaluates to what extent our tasks can be addressed solely based on knowledge about the distribution of words in the training sentences. Words are of course to some extent informative for most tasks, leading to relatively high performance in Tense, SubjNum and ObjNum. Recall that the words containing the probed features are disjoint between train and test partitions, so we are not observing a confound here, but rather the effect of the redundancies one expects in natural language data. For example, for Tense, since sentences often contain more than one verb in the same tense, NB-uni-tfidf can exploit non-target verbs as cues: the NB features most associated to the past class are verbs in the past tense (e.g "sensed", "lied", "announced"), and similarly for present (e.g "uses", "chuckles", "frowns"). Using bigram features (NB-bi-tfidf) brings in general little or no improvement with respect to the unigram baseline, except, trivially, for the BShift task, where NB-bi-tfidf can easily detect unlikely bigrams. NB-bi-tfidf has below-random performance on SOMO, confirming that the semantic intruder is not given away by superficial bigram cues.

Our first striking result is the good overall performance of Bag-of-Vectors, confirming early insights that aggregated word embeddings capture surprising amounts of sentence information (Pham et al., 2015; Arora et al., 2017; Adi et al., 2017). BoV's good WC and SentLen performance was already established by Adi et al. (2017). Not surprisingly, word-order-unaware BoV performs randomly in BShift and in the more sophisticated semantic tasks SOMO and CoordInv. More interestingly, BoV is very good at the Tense, SubjNum, ObjNum, and TopConst tasks (much better than the word-based baselines), and well above chance in TreeDepth. The good performance on Tense, SubjNum and ObjNum has a straightforward explanation we have already hinted at above. Many sentences are naturally "redundant", in the sense that most tensed verbs in a sentence are in the same tense, and similarly for number in nouns. In 95.2% Tense, 75.9% SubjNum and 78.7% ObjNum test sentences, the target tense/number feature is also the majority one for the whole sentence. Word embeddings capture features such as number and tense (Mikolov et al., 2013), so aggregated word embeddings will naturally track these features' majority values in a sentence. BoV's TopConst and TreeDepth performance is more surprising. Accuracy is well above NB, showing that BoV is exploiting cues beyond specific words strongly associated to the target classes. We conjecture that more abstract word features captured

[3]Similar results are obtained summing embeddings, and using GloVe embeddings (Pennington et al., 2014).

Task	SentLen	WC	TreeDepth	TopConst	BShift	Tense	SubjNum	ObjNum	SOMO	CoordInv
Baseline representations										
Majority vote	20.0	0.5	17.9	5.0	50.0	50.0	50.0	50.0	50.0	50.0
Hum. Eval.	100	100	84.0	84.0	98.0	85.0	88.0	86.5	81.2	85.0
Length	**100**	0.2	18.1	9.3	50.6	56.5	50.3	50.1	50.2	50.0
NB-uni-tfidf	22.7	**97.8**	24.1	41.9	49.5	77.7	68.9	64.0	38.0	50.5
NB-bi-tfidf	23.0	95.0	24.6	53.0	**63.8**	75.9	69.1	65.4	39.9	**55.7**
BoV-fastText	66.6	91.6	**37.1**	**68.1**	50.8	**89.1**	**82.1**	**79.8**	**54.2**	54.8
BiLSTM-last encoder										
Untrained	36.7	43.8	28.5	76.3	49.8	84.9	84.7	74.7	51.1	64.3
AutoEncoder	**99.3**	23.3	35.6	78.2	62.0	84.3	84.7	82.1	49.9	65.1
NMT En-Fr	83.5	**55.6**	42.4	81.6	62.3	88.1	89.7	89.5	52.0	71.2
NMT En-De	83.8	53.1	42.1	81.8	60.6	88.6	89.3	87.3	51.5	**71.3**
NMT En-Fi	82.4	52.6	40.8	81.3	58.8	88.4	86.8	85.3	52.1	71.0
Seq2Tree	94.0	14.0	**59.6**	**89.4**	**78.6**	**89.9**	**94.4**	**94.7**	49.6	67.8
SkipThought	68.1	35.9	33.5	75.4	60.1	89.1	80.5	77.1	**55.6**	67.7
NLI	75.9	47.3	32.7	70.5	54.5	79.7	79.3	71.3	53.3	66.5
BiLSTM-max encoder										
Untrained	73.3	**88.8**	46.2	71.8	70.6	89.2	85.8	81.9	73.3	68.3
AutoEncoder	**99.1**	17.5	45.5	74.9	71.9	86.4	87.0	83.5	73.4	71.7
NMT En-Fr	80.1	58.3	51.7	81.9	73.7	89.5	90.3	89.1	73.2	75.4
NMT En-De	79.9	56.0	52.3	82.2	72.1	90.5	90.9	89.5	73.4	**76.2**
NMT En-Fi	78.5	58.3	50.9	82.5	71.7	90.0	90.3	88.0	73.2	75.4
Seq2Tree	93.3	10.3	**63.8**	**89.6**	**82.1**	**90.9**	**95.1**	**95.1**	73.2	71.9
SkipThought	66.0	35.7	44.6	72.5	73.8	90.3	85.0	80.6	**73.6**	71.0
NLI	71.7	87.3	41.6	70.5	65.1	86.7	80.7	80.3	62.1	66.8
GatedConvNet encoder										
Untrained	90.3	17.1	30.3	47.5	62.0	78.2	72.2	70.9	61.4	59.6
AutoEncoder	**99.4**	16.8	46.3	75.2	71.9	87.7	88.5	86.5	**73.5**	72.4
NMT En-Fr	84.8	41.3	44.6	77.6	67.9	87.9	88.8	86.6	66.1	72.0
NMT En-De	89.6	49.0	50.5	81.7	72.3	90.4	91.4	89.7	72.8	**75.1**
NMT En-Fi	89.3	**51.5**	49.6	81.8	70.9	90.4	90.9	89.4	72.4	**75.1**
Seq2Tree	96.5	8.7	**62.0**	**88.9**	**83.6**	**91.5**	**94.5**	**94.3**	**73.5**	73.8
SkipThought	79.1	48.4	45.7	79.2	73.4	90.7	86.6	81.7	72.4	72.3
NLI	73.8	29.2	43.2	63.9	70.7	81.3	77.5	74.4	73.3	71.0

Table 2: **Probing task accuracies.** Classification performed by a MLP with sigmoid nonlinearity, taking pre-learned sentence embeddings as input (see Appendix for details and logistic regression results).

by the embeddings (such as the part of speech of a word) might signal different syntactic structures. For example, sentences in the "WHNP SQ ." top constituent class (e.g., "*How* long before you leave us again?") must contain a wh word, and will often feature an auxiliary or modal verb. BoV can rely on this information to noisily predict the correct class.

Encoding architectures Comfortingly, proper encoding architectures clearly outperform BoV. An interesting observation in Table 2 is that different encoder architectures trained with the same objective, and achieving similar performance on the training task,[4] can lead to linguistically different embeddings, as indicated by the probing tasks. Coherently with the findings of Conneau et al. (2017) for the downstream tasks, this suggests that the prior imposed by the encoder architecture strongly preconditions the nature of the embeddings. Complementing recent evidence that convolutional architectures are on a par with recurrent ones in seq2seq tasks (Gehring et al., 2017), we find that Gated ConvNet's overall probing task performance is comparable to that of the best LSTM architecture (although, as shown in Appendix, the LSTM has a slight edge on downstream tasks). We also replicate the finding of Conneau et al. (2017) that BiLSTM-max outperforms BiLSTM-last both in the downstream tasks (see Appendix) and in the probing tasks (Table 2). Interestingly, the latter only outperforms the former in SentLen, a task that captures a superficial aspect of sentences (how many words they contain), that could get in the way of inducing more useful linguistic knowledge.

[4]See Appendix for details on training task performance.

Training tasks We focus next on how different training tasks affect BiLSTM-max, but the patterns are generally representative across architectures. NMT training leads to encoders that are more linguistically aware than those trained on the NLI data set, despite the fact that we confirm the finding of Conneau and colleagues that NLI is best for downstream tasks (Appendix). Perhaps, NMT captures richer linguistic features useful for the probing tasks, whereas shallower or more *ad-hoc* features might help more in our current downstream tasks. Suggestively, the one task where NLI clearly outperforms NMT is WC. Thus, NLI training is better at preserving shallower word features that might be more useful in downstream tasks (cf. Figure 2 and discussion there).

Unsupervised training (SkipThought and AutoEncoder) is not on a par with supervised tasks, but still effective. AutoEncoder training leads, unsurprisingly, to a model excelling at SentLen, but it attains low performance in the WC prediction task. This curious result might indicate that the latter information is stored in the embeddings in a complex way, not easily readable by our MLP. At the other end, Seq2Tree is trained to predict annotation from the same parser we used to create some of the probing tasks. Thus, its high performance on TopConst, Tense, SubjNum, ObjNum and TreeDepth is probably an artifact. Indeed, for most of these tasks, Seq2Tree performance is above the human bound, that is, Seq2Tree learned to mimic the parser errors in our benchmarks. For the more challenging SOMO and CoordInv tasks, that only indirectly rely on tagging/parsing information, Seq2Tree is comparable to NMT, that does not use explicit syntactic information.

Perhaps most interestingly, BiLSTM-max already achieves very good performance without any training (Untrained row in Table 2). Untrained BiLSTM-max also performs quite well in the downstream tasks (Appendix). This architecture must encode priors that are intrinsically good for sentence representations. Untrained BiLSTM-max exploits the input fastText embeddings, and multiplying the latter by a random recurrent matrix provides a form of positional encoding. However, good performance in a task such as SOMO, where BoV fails and positional information alone should not help (the intruder is randomly distributed across the sentence), suggests that other architectural biases are at work. Intriguingly, a preliminary comparison of untrained BiLSTM-max and human subjects on the SOMO sentences evaluated by both reveals that, whereas humans have a bias towards finding sentences acceptable (62% sentences are rated as untampered with, vs. 48% ground-truth proportion), the model has a strong bias in the opposite direction (it rates 83% of the sentences as modified). A cursory look at contrasting errors confirms, unsurprisingly, that those made by humans are perfectly justified, while model errors are opaque. For example, the sentence "I didn't come here to *reunite* (orig. *undermine*) you" seems perfectly acceptable in its modified form, and indeed subjects judged it as such, whereas untrained BiLSTM-max "correctly" rated it as a modified item. Conversely, it is difficult to see any clear reason for the latter tendency to rate perfectly acceptable originals as modified. We leave a more thorough investigation to further work. See similar observations on the effectiveness of untrained ConvNets in vision by Ulyanov et al. (2017).

Probing task comparison A good encoder, such as NMT-trained BiLSTM-max, shows generally good performance across probing tasks. At one extreme, performance is not particularly high on the surface tasks, which might be an indirect sign of the encoder extracting "deeper" linguistic properties. At the other end, performance is still far from the human bounds on TreeDepth, BShift, SOMO and CoordInv. The last 3 tasks ask if a sentence is syntactically or semantically anomalous. This is a daunting job for an encoder that has not been explicitly trained on acceptability, and it is interesting that the best models are, at least to a certain extent, able to produce reasonable anomaly judgments. The asymmetry between the difficult TreeDepth and easier TopConst is also interesting. Intuitively, TreeDepth requires more nuanced syntactic information (down to the deepest leaf of the tree) than TopConst, that only requires identifying broad chunks.

Figure 1 reports how probing task accuracy changes in function of encoder training epochs. The figure shows that NMT probing performance is largely independent of target language, with strikingly similar development patterns across French, German and Finnish. Note in particular the similar probing accuracy curves in French and Finnish, while the corresponding BLEU scores (in lavender) are consistently higher in the former lan-

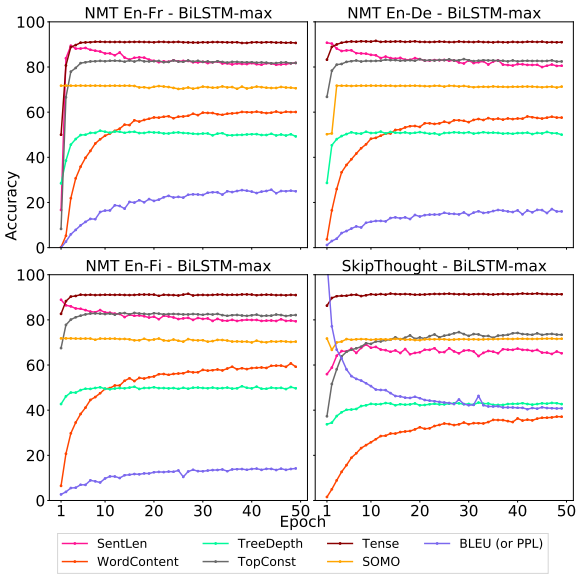


Figure 1: **Probing task scores after each training epoch, for NMT and SkipThought.** We also report training score evolution: BLEU for NMT; perplexity (PPL) for SkipThought.

guage. For both NMT and SkipThought, WC performance keeps increasing with epochs. For the other tasks, we observe instead an early flattening of the NMT probing curves, while BLEU performance keeps increasing. Most strikingly, SentLen performance is actually *decreasing*, suggesting again that, as a model captures deeper linguistic properties, it will tend to forget about this superficial feature. Finally, for the challenging SOMO task, the curves are mostly flat, suggesting that what BiLSTM-max is able to capture about this task is already encoded in its architecture, and further training doesn't help much.

Probing vs. downstream tasks Figure 2 reports correlation between performance on our probing tasks and the downstream tasks available in the SentEval[5] suite (Conneau and Kiela, 2018), which consists of classification (MR, CR, SUBJ, MPQA, SST2, SST5, TREC), natural language inference (SICK-E), semantic relatedness (SICK-R, STSB), paraphrase detection (MRPC) and semantic textual similarity (STS 2012 to 2017) tasks. Strikingly, WC is significantly positively correlated with all downstream tasks. This suggests that, at least for current models, the latter do not require extracting particularly abstract knowledge from the data. Just relying on the *words* contained in the input sentences can get you a long way. Conversely, there is a significant negative correlation between SentLen and most downstream tasks. The number of words in a sentence is not informative about its linguistic contents. The more models abstract away from such information, the more likely it is they will use their capacity to capture more interesting features, as the decrease of the SentLen curve along training (see Figure 1) also suggests. CoordInv and, especially, SOMO, the tasks requiring the most sophisticated semantic knowledge, are those that positively correlate with the largest number of downstream tasks after WC. We observe intriguing asymmetries: SOMO correlates with the SICK-E sentence entailment test, but not with SICK-R, which is about modeling sentence relatedness intuitions. Indeed, logical entailment requires deeper semantic analysis than modeling similarity judgments. TopConst and the number tasks negatively correlate with various similarity and sentiment data sets (SST, STS, SICK-R). This might expose biases in these tasks: SICK-R, for example, deliberately contains sentence pairs with opposite voice, that will have different constituent structure but equal meaning (Marelli et al., 2014). It might also mirrors genuine factors affecting similarity judgments (e.g., two sentences differing only in object number are very similar). Remarkably, TREC question type classification is the downstream task correlating with most probing tasks. Question classification is certainly an outlier among our downstream tasks, but we must leave a full understanding of this behaviour to future work (this is exactly the sort of analysis our probing tasks should stimulate).

5 Related work

Adi et al. (2017) introduced SentLen, WC and a word order test, focusing on a bag-of-vectors baseline, an autoencoder and skip-thought (all trained on the same data used for the probing tasks). We recast their tasks so that they only require a sentence embedding as input (two of their tasks also require word embeddings, polluting sentence-level evaluation), we extend the evaluation to more tasks, encoders and training objectives, and we relate performance on the probing tasks with that on downstream tasks. Shi et al. (2016) also use 3 probing tasks, including Tense and TopConst. It is not clear that they controlled for the same factors we considered (in particular, lexical overlap and

[5] https://github.com/facebookresearch/SentEval

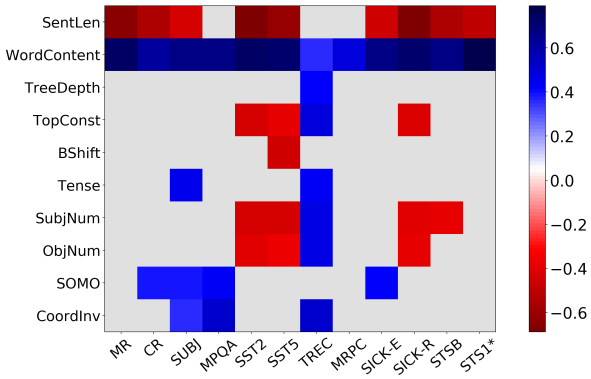


Figure 2: **Spearman correlation matrix between probing and downstream tasks.** Correlations based on all sentence embeddings we investigated (more than 40). Cells in gray denote task pairs that are not significantly correlated (after correcting for multiple comparisons).

sentence length), and they use much smaller training sets, limiting classifier-based evaluation to logistic regression. Moreover, they test a smaller set of models, focusing on machine translation.

Belinkov et al. (2017a), Belinkov et al. (2017b) and Dalvi et al. (2017) are also interested in understanding the type of linguistic knowledge encoded in sentence and word embeddings, but their focus is on word-level morphosyntax and lexical semantics, and specifically on NMT encoders and decoders. Sennrich (2017) also focuses on NMT systems, and proposes a contrastive test to assess how they handle various linguistic phenomena. Other work explores the linguistic behaviour of recurrent networks and related models by using visualization, input/hidden representation deletion techniques or by looking at the word-by-word behaviour of the network (e.g., Nagamine et al., 2015; Hupkes et al., 2017; Li et al., 2016; Linzen et al., 2016; Kàdàr et al., 2017; Li et al., 2017). These methods, complementary to ours, are not agnostic to encoder architecture, and cannot be used for general-purpose cross-model evaluation.

Finally, Conneau et al. (2017) propose a large-scale, multi-task evaluation of sentence embeddings, focusing entirely on downstream tasks.

6 Conclusion

We introduced a set of tasks probing the linguistic knowledge of sentence embedding methods. Their purpose is not to encourage the development of *ad-hoc* models that attain top performance on them, but to help exploring what information is captured by different pre-trained encoders.

We performed an extensive linguistic evaluation of modern sentence encoders. Our results suggest that the encoders are capturing a wide range of properties, well above those captured by a set of strong baselines. We further uncovered interesting patterns of correlation between the probing tasks and more complex "downstream" tasks, and presented a set of intriguing findings about the linguistic properties of various embedding methods. For example, we found that Bag-of-Vectors is surprisingly good at capturing sentence-level properties, thanks to redundancies in natural linguistic input. We showed that different encoder architectures trained with the same objective with similar performance can result in different embeddings, pointing out the importance of the architecture prior for sentence embeddings. In particular, we found that BiLSTM-max embeddings are already capturing interesting linguistic knowledge before training, and that, after training, they detect semantic acceptability without having been exposed to anomalous sentences before. We hope that our publicly available probing task set will become a standard benchmarking tool of the linguistic properties of new encoders, and that it will stir research towards a better understanding of what they learn.

In future work, we would like to extend the probing tasks to other languages (which should be relatively easy, given that they are automatically generated), investigate how multi-task training affects probing task performance and leverage our probing tasks to find more linguistically-aware universal encoders.

Acknowledgments

We thank David Lopez-Paz, Holger Schwenk, Hervé Jégou, Marc'Aurelio Ranzato and Douwe Kiela for useful comments and discussions.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *Proceedings of ICLR Conference Track*. Toulon, France. Published online: https://openreview.net/group?id=ICLR.cc/2017/conference.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proceedings of ICLR Conference Track*. Toulon, France. Published

online: https://openreview.net/group?id=ICLR.cc/2017/conference.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *Advances in neural information processing systems (NIPS)* .

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017a. What do neural machine translation models learn about morphology? In *Proceedings of ACL*. Vancouver, Canada, pages 861–872.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017b. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of IJCNLP*. Taipei, Taiwan, pages 1–10.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Proceedings of EMNLP* .

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine learning*. ACM, pages 160–167.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. *Proceedings of LREC* .

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of EMNLP*. Copenhagen, Denmark, pages 670–680.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, and Stephan Vogel. 2017. Understanding and improving morphological learning in the neural machine translation decoder. In *Proceedings of IJCNLP*. Taipei, Taiwan, pages 142–151.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. *Proceedings of the 34th International Conference on Machine Learning* .

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of ICML*. Sydney, Australia, pages 1243–1252.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2017. Visualisation and diagnostic classifiers reveal how recurrent and recursive neural networks process hierarchical structure. http://arxiv.org/abs/1711.10203.

Allan Jabri, Armand Joulin, and Laurens van der Maaten. 2016. Revisiting visual question answering baselines. In *Proceedings of ECCV*. Amsterdam, the Netherlands, pages 727–739.

Àkos Kàdàr, Grzegorz Chrupała, and Afra Alishahi. 2017. Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics* 43(4):761–780.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*. pages 3294–3302.

Dan Klein and Christopher Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL*. Sapporo, Japan, pages 423–430.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*. volume 5, pages 79–86.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*. Association for Computational Linguistics, pages 177–180.

Alice Lai and Julia Hockenmaier. 2014. Illinois-LH: A denotational and distributional approach to semantics. In *Proceedings of SemEval*. Dublin, Ireland, pages 329–334.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *Proceedings of NAACL*. San Diego, CA, pages 681–691.

Jiwei Li, Monroe Will, and Dan Jurafsky. 2017. Efficient estimation of word representations in vector space. https://arxiv.org/abs/1612.08220.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics* 4:521–535.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of LREC*. Rekjavik, Iceland, pages 216–223.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of LREC*. Miyazaki, Japan.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL*. Atlanta, Georgia, pages 746–751.

Tasha Nagamine, Michael L. Seltzer, and Nima Mesgarani. 2015. Exploring how deep neural networks form phonemic categories. In *Proceedings of INTERSPEECH*. Dresden, Germany, pages 1912–1916.

Matthew Nelson, Imen El Karoui, Kristof Giber, Xiaofang Yang, Laurent Cohen, Hilda Koopman, Sydney Cash, Lionel Naccache, John Hale, Christophe Pallier, and Stanislas Dehaene. 2017. Neurophysiological dynamics of phrase-structure building during sentence processing. *Proceedings of the National Academy of Sciences* 114(18):E3669–E3678.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL*. Barcelona, Spain, pages 271–278.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernandez. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of ACL*. Berlin, Germany, pages 1525–1534.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*. Doha, Qatar, pages 1532–1543.

Nghia The Pham, Germán Kruszewski, Angeliki Lazaridou, and Marco Baroni. 2015. Jointly optimizing word representations for lexical and sentential tasks with the C-PHRASE model. In *Proceedings of ACL*. Beijing, China, pages 971–981.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? assessing MT quality with contrastive translation pairs. In *Proceedings of EACL (Short Papers)*. Valencia, Spain, pages 376–382.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of EMNLP*. Austin, Texas, pages 1526–1534.

Richard Socher, Eric Huang, Jeffrey Pennin, Andrew Ng, and Christopher Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of NIPS*. Granada, Spain, pages 801–809.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. In *International Conference on Learning Representations*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*. pages 2440–2448.

Ilya Sutskever, Oriol Vinyals, and Quoc Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*. Montreal, Canada, pages 3104–3112.

Shuai Tang, Hailin Jin, Chen Fang, Zhaowen Wang, and Virginia R de Sa. 2017. Trimming and improving skip-thought vectors. *Proceedings of the 2nd Workshop on Representation Learning for NLP* .

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2017. Deep image prior. https://arxiv.org/abs/1711.10925.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*. pages 2773–2781.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of NAACL*.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep recurrent models with fast-forward connections for neural machine translation. *arXiv preprint arXiv:1606.04199* .

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of ICCV*. Santiago, Chile, pages 19–27.

7 Appendix

Amazon Mechanical Turk survey

Subjects were recruited through the standard Amazon Mechanical Turk interface.[6] We created independent surveys for the SOMO, CoordInv and BShift tasks. We asked subjects to identify which sentences were acceptable and which were anomalous/inverted. Participants were restricted to those based in an English-speaking country.

To maximize annotation quality, we created a control set. Two authors annotated 200 random sentences from each task in a blind pretest. Those sentences on which they agreed were included in the control set.

We collected at least 10 judgments per sentence, for 1k random sentences from each task. We only retained judgments by subjects that rated at least 10 control sentences with accuracy of at least 90%. After filtering, we were left with averages of 2.5, 2.9 and 12 judgments per sentence for SOMO, CoordInv and BShift, respectively. Responses were aggregated by majority voting, before computing the final accuracies.

We did not record any personal data from subjects, and we only used the judgments in aggregated format to produce the estimated human upper bounds reported in our tables.

Further training details

Encoder training For seq2seq tasks, after hyper-parameter tuning, we chose 2-layer LSTM decoders with 512 hidden units. For NLI, we settled on a multi-layer perceptron with 100 hidden units. As is now common in NMT, we apply Byte Pair Encoding (BPE) (Sennrich, 2017) to target sentences only, with 40k codes (see Table 1 in the main text for examples of transformed target sentences). We tune dropout rate and input embedding size, picking 1024 for BiLSTMs and 512 for Gated ConvNets. We use the Adam optimizer for BiLSTMs and SGD with momentum for Gated ConvNets (after Adam gave very poor results). The encoder representation is fed to the decoder at every time step. For model selection on the validation sets, we use BLEU score[7] for NMT and AutoEncoder, perplexity for SkipThought and accuracy for Seq2Tree and NLI.

Table 3 reports test set performance of the various architectures on the original training tasks. For NMT and Seq2Tree, we left out two random sets of 10k sentences from the training data for dev and test. The NLI dev and test sets are the ones of SNLI. Observe how results are similar for the three encoders, while, as discussed in the main text, they differ in terms of the linguistic properties their sentence embeddings are capturing. The last row of the table reports BLEU scores for our BiLSTM architecture trained with attention, showing that the architecture is on par with current NMT models, when attention is introduced. For comparison, our attention-based model obtains 37 BLEU score on the standard WMT'14 En-Fr benchmark.

Model	En-Fr	En-De	En-Fi	Seq2Tree	NLI
Gated ConvNet	25.9	17.0	14.2	52.3	83.5
BiLSTM-last	27.3	17.9	14.3	55.2	84.0
BiLSTM-max	27.0	18.0	14.7	53.7	85.3
BiLSTM-Att	39.1	27.2	21.9	58.4	-

Table 3: **Test results for training tasks.** Figure of merit is BLEU score for NMT and accuracy for Seq2Tree and NLI.

Probing task training The probing task results reported in the main text are obtained with a MLP that uses the Sigmoid nonlinearity, which we found to perform better than Tanh. We tune the L^2 regularization parameter, the number of hidden states (in [50, 100, 200]) and the dropout rate (in [0, 0.1, 0.2]) on the validation set of each probing task. Only for WC, which has significantly more output classes (1000) than the other tasks, we report Logistic Regression results, since they were consistently better.

Logistic regression results

Logistic regression performance approximates MLP performance (compare Table 4 here to Table 2 in the main text). This suggests that most linguistic properties can be extracted with a linear readout of the embeddings. Interestingly, if we focus on a good model-training combination, such as BiLSTM-max trained on French NMT, the tasks where the improvement from logistic regression to MLP is relatively large (>3%) are those arguably requiring the most nuanced linguistic knowledge (TreeDepth, SOMO, CoordInv).

Downstream task results

We evaluate our architecture+training method combinations on the downstream tasks from the

[6] https://www.mturk.com/

[7] MOSES multi-bleu.perl script (Koehn et al., 2007)

Task	SentLen	WC	TreeDepth	TopConst	BShift	Tense	SubjNum	ObjNum	SOMO	CoordInv
Baseline representations										
Majority vote	20.0	0.5	17.9	5.0	50.0	50.0	50.0	50.0	50.0	50.0
Hum. Eval.	100	100	84.0	84.0	98.0	85.0	88.0	86.5	81.2	85.0
Length	100	0.2	18.1	9.3	50.6	56.5	50.3	50.1	50.2	50.0
NB-uni-tfidf	22.7	97.8	24.1	41.9	49.5	77.7	68.9	64.0	38.0	50.5
NB-bi-tfidf	23.0	95.0	24.6	53.0	63.8	75.9	69.1	65.4	39.9	55.7
BoV fastText	54.8	91.6	32.3	63.1	50.8	87.8	81.9	79.3	50.3	52.7
BiLSTM-last encoder										
Untrained	32.6	43.8	24.6	74.1	52.2	83.7	82.8	71.8	49.9	64.5
AutoEncoder	98.9	23.3	28.2	72.5	60.1	80.0	81.2	76.8	50.7	62.5
NMT En-Fr	82.9	55.6	35.8	79.8	59.6	86.0	87.6	85.5	50.3	66.1
NMT En-De	82.7	53.1	35.2	80.1	58.3	86.6	88.3	84.5	50.5	66.1
NMT En-Fi	81.7	52.6	35.2	79.3	57.5	86.5	84.4	82.6	50.5	65.9
Seq2Tree	93.2	14.0	46.4	88.5	74.9	87.3	90.5	89.7	50.6	63.4
SkipThought	59.5	35.9	30.2	73.1	58.4	88.7	78.4	76.4	53.0	64.6
NLI	71.6	47.3	28.4	67.4	53.3	77.3	76.6	69.6	51.6	64.7
BiLSTM-max encoder										
Untrained	66.2	88.8	43.1	68.8	70.3	88.7	84.6	81.7	73.0	69.1
AutoEncoder	98.5	17.5	42.3	71.0	69.5	85.7	85.0	80.9	73.0	70.9
NMT En-Fr	79.3	58.3	45.7	80.5	71.2	87.8	88.1	86.3	69.9	71.8
NMT En-De	78.2	56.0	46.9	81.0	69.8	89.1	89.7	87.9	71.3	73.5
NMT En-Fi	77.5	58.3	45.8	80.5	69.7	88.2	88.9	86.1	71.9	72.8
Seq2Tree	91.8	10.3	54.6	88.7	80.0	89.5	91.8	90.7	68.6	69.8
SkipThought	59.6	35.7	42.7	70.5	73.4	90.1	83.3	79.0	70.3	70.1
NLI	65.1	87.3	38.5	67.9	63.8	86.0	78.9	78.5	59.5	64.9
GatedConvNet encoder										
Untrained	90.3	17.1	30.3	47.5	62.0	78.2	72.2	70.9	61.4	59.1
AutoEncoder	99.3	16.8	41.9	69.6	68.1	85.4	85.4	82.1	69.8	70.6
NMT En-Fr	84.3	41.3	36.9	73.8	63.7	85.6	85.7	83.8	58.8	68.1
NMT En-De	87.6	49.0	44.7	78.8	68.8	89.5	89.6	86.8	69.5	70.0
NMT En-Fi	89.1	51.5	44.1	78.6	67.2	88.7	88.5	86.3	68.3	71.0
Seq2Tree	94.5	8.7	53.1	87.4	80.9	89.6	91.5	90.8	68.3	71.6
SkipThought	73.2	48.4	40.4	76.2	71.6	89.8	84.0	79.8	68.9	68.0
NLI	70.9	29.2	38.8	59.3	66.8	80.1	77.7	72.8	69.0	69.1

Table 4: **Probing task accuracies with Logistic Regression.** Taking pre-learned sentence embeddings as input.

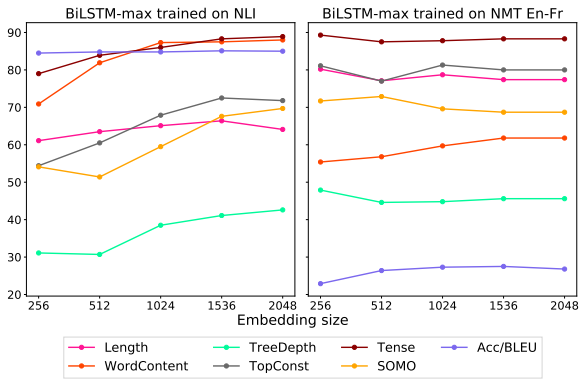


Figure 3: **Evolution of probing tasks results wrt. embedding size**. The sentence representations are generated by a BiLSTM-max encoder trained on either NLI or NMT En-Fr, with increasing sentence embedding size.

SentEval toolkit.[8] See documentation there for the tasks, that range from subjectivity analysis to question-type classification, to paraphrase detection and entailment. Also refer to the SentEval page and to Conneau et al. (2017) for the specifics of training and figures of merit for each task. In all cases, we used as input our pre-trained embeddings without fine-tuning them to the tasks. Results are reported in Table 5.

We replicate the finding of Conneau and colleagues about the effectiveness of the BiLSTM architecture with max pooling, that has also a slight edge over GatedConvNet (an architecture they did not test). As for the probing tasks, we again notice that BiLSTM-max is already effective without training, and more so than the alternative architec-

[8] https://github.com/facebookresearch/SentEval

Model	MR	CR	SUBJ	MPQA	SST-2	SST-5	TREC	MRPC	SICK-E	SICK-R	STSB
Baseline representations											
Chance	50.0	63.8	50.0	68.8	50.0	28.6	21.2	66.5	56.7	0.0	0.0
BoV fastText	78.2	80.2	91.8	88.0	82.3	45.1	83.4	74.4	78.9	82.0	70.2
BiLSTM-last encoder											
Untrained	69.7	70.2	84.8	87.0	77.2	37.6	79.6	68.5	71.6	68.2	54.8
AutoEncoder	66.0	70.7	85.7	81.1	70.0	36.2	84.0	69.9	72.2	67.6	58.3
NMT En-Fr	74.5	78.7	90.3	88.9	79.5	42.0	91.2	73.7	79.7	78.3	69.9
NMT En-De	74.8	78.4	89.8	88.7	78.8	42.3	88.0	74.1	78.8	77.5	69.3
NMT En-Fi	74.2	78.0	89.6	88.9	78.4	39.6	84.6	75.6	79.1	77.1	67.1
Seq2Tree	62.5	69.3	85.7	78.7	64.4	33.0	86.4	73.6	71.9	59.1	44.8
SkipThought	77.1	78.9	92.2	86.7	81.3	43.9	82.4	72.7	77.8	80.0	73.9
NLI	77.3	84.1	88.1	88.6	81.7	43.9	86.0	74.8	83.9	85.6	74.2
BiLSTM-max encoder											
Untrained	75.6	78.2	90.0	88.1	79.9	39.1	80.6	72.2	80.8	83.3	70.2
AutoEncoder	68.3	74.0	87.2	84.6	70.8	34.0	85.0	71.0	75.3	70.4	55.1
NMT En-Fr	76.5	81.1	91.4	89.7	77.7	42.2	89.6	75.1	79.3	78.8	68.8
NMT En-De	77.7	81.2	92.0	89.7	79.3	41.0	88.2	76.2	81.0	80.0	68.7
NMT En-Fi	77.0	81.1	91.5	90.0	80.3	43.4	87.2	75.0	81.7	80.3	69.5
Seq2Tree	65.2	74.4	88.3	80.2	66.5	31.6	85.0	72.0	74.8	65.1	36.1
SkipThought	78.0	82.8	93.0	87.3	81.5	41.9	86.8	73.2	80.0	82.0	71.5
NLI	79.2	86.7	90.0	89.8	83.5	46.4	86.0	74.5	84.5	87.5	76.6
GatedConvNet encoder											
Untrained	65.5	65.3	78.3	82.9	65.8	34.0	67.6	68.1	61.6	56.7	38.9
AutoEncoder	72.1	74.1	86.6	86.0	74.4	36.6	79.6	69.7	72.0	65.8	45.5
NMT En-Fr	74.5	78.3	88.7	88.4	76.8	38.3	86.2	72.5	77.3	73.2	60.4
NMT En-De	77.1	80.4	90.9	89.2	79.2	41.9	90.4	76.8	81.9	78.7	69.4
NMT En-Fi	76.9	82.0	91.2	90.0	78.8	41.9	90.0	76.7	81.1	79.5	70.8
Seq2Tree	65.3	73.1	85.0	79.8	63.7	31.8	81.2	72.9	74.0	58.4	30.8
SkipThought	76.0	81.7	91.5	87.2	77.9	41.5	88.8	72.3	79.5	80.0	67.8
NLI	76.7	84.7	87.4	89.1	79.2	40.9	82.0	70.8	82.0	84.7	64.4
Other sentence embedding methods											
SkipThought	79.4	83.1	93.7	89.3	82.9	-	88.4	72.4	79.5	85.8	72.1
InferSent	81.1	86.3	92.4	90.2	84.6	46.3	88.2	76.2	86.3	88.4	75.8
MultiTask	82.4	88.6	93.8	90.7	85.1	-	94.0	78.3	87.4	88.5	78.7

Table 5: **Downstream tasks results for various sentence encoder architectures pre-trained in different ways.**

tures.

Interestingly, we also confirm Conneau et al.'s finding that NLI is the best source task for pre-training, despite the fact that, as we saw in the main text (Table 2 there), NMT pre-training leads to models that are capturing more linguistic properties. As they observed for downstream tasks, increasing the embedding dimension while adding capacity to the model is beneficial (see Figure 3) also for probing tasks in the case of NLI. However, it does not seem to boost the performance of the NMT En-Fr encoder.

Finally, the table also shows results from the literature recently obtained with various state-of-the-art general-purpose encoders, namely: SkipThought with layer normalization (Ba et al., 2016), InferSent (BiLSTM-max as trained on NLI by Conneau et al.) and MultiTask (Subramanian et al., 2018). A comparison of these results with ours confirms that we are testing models that do not lag much behind the state of the art.

Know What You Don't Know: Unanswerable Questions for SQuAD

Pranav Rajpurkar* **Robin Jia*** **Percy Liang**
Computer Science Department, Stanford University
{pranavsr,robinjia,pliang}@cs.stanford.edu

Abstract

Extractive reading comprehension systems can often locate the correct answer to a question in a context document, but they also tend to make unreliable guesses on questions for which the correct answer is not stated in the context. Existing datasets either focus exclusively on answerable questions, or use automatically generated unanswerable questions that are easy to identify. To address these weaknesses, we present SQuAD 2.0, the latest version of the Stanford Question Answering Dataset (SQuAD). SQuAD 2.0 combines existing SQuAD data with over 50,000 unanswerable questions written adversarially by crowdworkers to look similar to answerable ones. To do well on SQuAD 2.0, systems must not only answer questions when possible, but also determine when no answer is supported by the paragraph and abstain from answering. SQuAD 2.0 is a challenging natural language understanding task for existing models: a strong neural system that gets 86% F1 on SQuAD 1.1 achieves only 66% F1 on SQuAD 2.0.

1 Introduction

Machine reading comprehension has become a central task in natural language understanding, fueled by the creation of many large-scale datasets (Hermann et al., 2015; Hewlett et al., 2016; Rajpurkar et al., 2016; Nguyen et al., 2016; Trischler et al., 2017; Joshi et al., 2017). In turn, these datasets have spurred a diverse array of model architecture improvements (Seo et al., 2016; Hu et al., 2017; Wang et al., 2017; Clark and Gardner, 2017; Huang et al., 2018). Recent work has even produced systems that surpass human-level exact match accuracy on the Stanford Question Answering Dataset (SQuAD), one of the most widely-used reading comprehension benchmarks (Rajpurkar et al., 2016).

Article: Endangered Species Act
Paragraph: "*... Other legislation followed, including the Migratory Bird Conservation Act of 1929, a 1937 treaty prohibiting the hunting of right and gray whales, and the Bald Eagle Protection Act of 1940. These later laws had a low cost to society—the species were relatively rare—and little opposition was raised.*"

Question 1: "*Which laws faced significant opposition?*"
Plausible Answer: *later laws*

Question 2: "*What was the name of the 1937 treaty?*"
Plausible Answer: *Bald Eagle Protection Act*

Figure 1: Two unanswerable questions written by crowdworkers, along with plausible (but incorrect) answers. Relevant keywords are shown in blue.

Nonetheless, these systems are still far from true language understanding. Recent analysis shows that models can do well at SQuAD by learning context and type-matching heuristics (Weissenborn et al., 2017), and that success on SQuAD does not ensure robustness to distracting sentences (Jia and Liang, 2017). One root cause of these problems is SQuAD's focus on questions for which a correct answer is guaranteed to exist in the context document. Therefore, models only need to select the span that seems most related to the question, instead of checking that the answer is actually entailed by the text.

In this work, we construct SQuAD 2.0,[1] a new dataset that combines answerable questions from the previous version of SQuAD (SQuAD 1.1)

* The first two authors contributed equally to this paper.

[1] In the ACL version of this paper, we called our new dataset SQuADRUN; here we use the name SQuAD 2.0, to emphasize that it is in fact the new version of SQuAD.

with 53,775 new, unanswerable questions about the same paragraphs. Crowdworkers crafted these questions so that (1) they are *relevant* to the paragraph, and (2) the paragraph contains a *plausible answer*—something of the same type as what the question asks for. Two such examples are shown in Figure 1.

We confirm that SQuAD 2.0 is both challenging and high-quality. A state-of-the-art model achieves only 66.3% F1 score when trained and tested on SQuAD 2.0, whereas human accuracy is 89.5% F1, a full 23.2 points higher. The same model architecture trained on SQuAD 1.1 gets 85.8% F1, only 5.4 points worse than humans. We also show that our unanswerable questions are more challenging than ones created automatically, either via distant supervision (Clark and Gardner, 2017) or a rule-based method (Jia and Liang, 2017). We release SQuAD 2.0 to the public as new version of SQuAD, and make it the primary benchmark on the official SQuAD leaderboard.[2] We are optimistic that this new dataset will encourage the development of reading comprehension systems that know what they don't know.

2 Desiderata

We first outline our goals for SQuAD 2.0. Besides the generic goals of large size, diversity, and low noise, we posit two desiderata specific to unanswerable questions:

Relevance. The unanswerable questions should appear relevant to the topic of the context paragraph. Otherwise, simple heuristics (e.g., based on word overlap) could distinguish answerable and unanswerable questions (Yih et al., 2013).

Existence of plausible answers. There should be some span in the context whose type matches the type of answer the question asks for. For example, if the question asks, "*What company was founded in 1992?*", then some company should appear in the context. Otherwise, type-matching heuristics could distinguish answerable and unanswerable questions (Weissenborn et al., 2017).

3 Existing datasets

Next, we survey existing reading comprehension datasets with these criteria in mind. We use the term "negative example" to refer to a context passage paired with an unanswerable question.

3.1 Extractive datasets

In extractive reading comprehension datasets, a system must extract the correct answer to a question from a context document or paragraph. The Zero-shot Relation Extraction dataset (Levy et al., 2017) contains negative examples generated with distant supervision. Levy et al. (2017) found that 65% of these negative examples do not have a plausible answer, making them easy to identify.

Other distant supervision strategies can also create negative examples. TriviaQA (Joshi et al., 2017) retrieves context documents from the web or Wikipedia for each question. Some documents do not contain the correct answer, yielding negative examples; however, these are excluded from the final dataset. Clark and Gardner (2017) generate negative examples for SQuAD by pairing existing questions with other paragraphs from the same article based on TF-IDF overlap; we refer to these as TFIDF examples. In general, distant supervision does not ensure the existence of a plausible answer in the retrieved context, and might also add noise, as the context might contain a paraphrase of the correct answer. Moreover, when retrieving from a small set of possible contexts, as in Clark and Gardner (2017), we find that the retrieved paragraphs are often not very relevant to the question, making these negative examples easy to identify.

The NewsQA data collection process also yields unanswerable questions, because crowdworkers write questions given only a summary of an article, not the full text (Trischler et al., 2017). Only 9.5% of their questions are unanswerable, making this strategy hard to scale. Of this fraction, we found that some are misannotated as unanswerable, and others are out-of-scope (e.g., summarization questions). Trischler et al. (2017) also exclude negative examples from their final dataset.

Jia and Liang (2017) propose a rule-based procedure for editing SQuAD questions to make them unanswerable. Their questions are not very diverse: they only replace entities and numbers with similar words, and replace nouns and adjectives with WordNet antonyms. We refer to these unanswerable questions as RULEBASED questions.

3.2 Answer sentence selection datasets

Sentence selection datasets test whether a system can rank sentences that answer a question higher

[2] As with previous versions of SQuAD, we release SQuAD 2.0 under the CC BY-SA 4.0 license.

Reasoning	Description	Example	Percentage
Negation	Negation word inserted or removed.	Sentence: "*Several hospital pharmacies have decided to outsource high risk preparations . . .*" Question: "*What types of pharmacy functions have **never** been outsourced?*"	9%
Antonym	Antonym used.	S: "*the extinction of the dinosaurs. . . allowed the tropical rainforest to spread out across the continent.*" Q: "*The extinction of what led to the **decline** of rainforests?*"	20%
Entity Swap	Entity, number, or date replaced with other entity, number, or date.	S: "*These values are much greater than the 9–88 cm as projected . . . in its Third Assessment Report.*" Q: "*What was the projection of sea level increases in the **fourth assessment report**?*"	21%
Mutual Exclusion	Word or phrase is mutually exclusive with something for which an answer is present.	S: "*BSkyB. . . waiv[ed] the charge for subscribers whose package included two or more premium channels.*" Q: "*What service did BSkyB **give away for free unconditionally**?*"	15%
Impossible Condition	Asks for condition that is not satisfied by anything in the paragraph.	S: "*Union forces left Jacksonville and confronted a Confederate Army at the Battle of Olustee. . . Union forces then retreated to Jacksonville and held the city for the remainder of the war.*" Q: "*After what battle did Union forces leave Jacksonville **for good**?*"	4%
Other Neutral	Other cases where the paragraph does not imply any answer.	S: "*Schuenemann et al. concluded in 2011 that the Black Death. . . was caused by a variant of Y. pestis. . .*" Q: "*Who **discovered** Y. pestis?*"	24%
Answerable	Question is answerable (i.e. dataset noise).		7%

Table 1: Types of negative examples in SQuAD 2.0 exhibiting a wide range of phenomena.

than sentences that do not. Wang et al. (2007) constructed the QASENT dataset from questions in the TREC 8-13 QA tracks. Yih et al. (2013) showed that lexical baselines are highly competitive on this dataset. WikiQA (Yang et al., 2015) pairs questions from Bing query logs with sentences from Wikipedia. Like TFIDF examples, these sentences are not guaranteed to have plausible answers or high relevance to the question. The dataset is also limited in scale (3,047 questions, 1,473 answers).

3.3 Multiple choice datasets

Finally, some datasets, like MCTest (Richardson et al., 2013) and RACE (Lai et al., 2017), pose multiple choice questions, which can have a "none of the above" option. In practice, multiple choice options are often unavailable, making these datasets less suited for training user-facing systems. Multiple choice questions also tend to be quite different from extractive ones, with more emphasis on fill-in-the-blank, interpretation, and summarization (Lai et al., 2017).

4 SQuAD 2.0

We now describe our new dataset, which we constructed to satisfy both the relevance and plausible answer desiderata from Section 2.

4.1 Dataset creation

We employed crowdworkers on the Daemo crowdsourcing platform (Gaikwad et al., 2015) to write unanswerable questions. Each task consisted of an entire article from SQuAD 1.1. For each paragraph in the article, workers were asked to pose up to five questions that were impossible to answer based on the paragraph alone, while referencing entities in the paragraph and ensuring that a plausible answer is present. As inspiration, we also showed questions from SQuAD 1.1 for each paragraph; this further encouraged unanswerable questions to look similar to answerable ones. Workers were asked to spend 7 minutes per paragraph, and were paid $10.50 per hour. Screenshots of our interface are shown in Appendix A.1.

We removed questions from workers who wrote 25 or fewer questions on that article; this filter helped remove noise from workers who had trouble understanding the task, and therefore quit before completing the whole article. We applied this filter to both our new data and the existing answerable questions from SQuAD 1.1. To generate train, development, and test splits, we used the same partition of articles as SQuAD 1.1, and combined the existing data with our new data for each split. For the SQuAD 2.0 development and test sets, we removed articles for which we did not

	SQuAD 1.1	SQuAD 2.0
Train		
Total examples	87,599	130,319
Negative examples	0	43,498
Total articles	442	442
Articles with negatives	0	285
Development		
Total examples	10,570	11,873
Negative examples	0	5,945
Total articles	48	35
Articles with negatives	0	35
Test		
Total examples	9,533	8,862
Negative examples	0	4,332
Total articles	46	28
Articles with negatives	0	28

Table 2: Dataset statistics of SQuAD 2.0, compared to the previous SQuAD 1.1.

collect unanswerable questions. This resulted in a roughly one-to-one ratio of answerable to unanswerable questions in these splits, whereas the train data has roughly twice as many answerable questions as unanswerable ones. Table 2 summarizes overall statistics of SQuAD 2.0.

4.2 Human accuracy

To confirm that our dataset is clean, we hired additional crowdworkers to answer all questions in the SQuAD 2.0 development and test sets. In each task, we showed workers an entire article from the dataset. For each paragraph, we showed all associated questions; unanswerable and answerable questions were shuffled together. For each question, workers were told to either highlight the answer in the paragraph, or mark it as unanswerable. Workers were told to expect every paragraph to have some answerable and some unanswerable questions. They were asked to spend one minute per question, and were paid $10.50 per hour.

To reduce crowdworker noise, we collected multiple human answers for each question and selected the final answer by majority vote, breaking ties in favor of answering questions and preferring shorter answers to longer ones. On average, we collected 4.8 answers per question. We note that for SQuAD 1.1, Rajpurkar et al. (2016) evaluated a single human's performance; therefore, they likely underestimate human accuracy.

4.3 Analysis

We manually inspected 100 randomly chosen negative examples from our development set to understand the challenges these examples present. In Table 1, we define different categories of negative examples, and give examples and their frequency in SQuAD 2.0. We observe a wide range of phenomena, extending beyond expected phenomena like negation, antonymy, and entity changes. In particular, SQuAD 2.0 is much more diverse than RULEBASED, which creates unanswerable questions by applying entity, number, and antonym swaps to existing SQuAD 1.1 questions. We also found that 93% of the sampled negative examples are indeed unanswerable.

5 Experiments

5.1 Models

We evaluated three existing model architectures: the BiDAF-No-Answer (BNA) model proposed by Levy et al. (2017), and two versions of the DocumentQA No-Answer (DocQA) model from Clark and Gardner (2017), namely versions with and without ELMo (Peters et al., 2018). These models all learn to predict the probability that a question is unanswerable, in addition to a distribution over answer choices. At test time, models abstain whenever their predicted probability that a question is unanswerable exceeds some threshold. We tune this threshold separately for each model on the development set. When evaluating on the test set, we use the threshold that maximizes F1 score on the development set. We find this strategy does slightly better than simply taking the argmax prediction, possibly due to the different proportions of negative examples at training and test time.

5.2 Main results

First, we trained and tested all three models on SQuAD 2.0, as shown in Table 3. Following Rajpurkar et al. (2016), we report average exact match and F1 scores.[3] The best model, DocQA + ELMo, achieves only 66.3 F1 on the test set, 23.2 points lower than the human accuracy of 89.5 F1. Note that a baseline that always abstains gets 48.9 test F1; existing models are closer to this baseline than they are to human performance. Therefore, we see significant room for model improvement on this task. We also compare with reported test numbers for analogous model architectures on SQuAD 1.1. There is a much larger gap between humans and machines on SQuAD 2.0 compared to SQuAD 1.1, which confirms that SQuAD 2.0 is a much harder dataset for existing models.

[3] For negative examples, abstaining receives a score of 1, and any other response gets 0, for both exact match and F1.

System	SQuAD 1.1 test		SQuAD 2.0 dev		SQuAD 2.0 test	
	EM	F1	EM	F1	EM	F1
BNA	68.0	77.3	59.8	62.6	59.2	62.1
DocQA	72.1	81.0	61.9	64.8	59.3	62.3
DocQA + ELMo	**78.6**	**85.8**	**65.1**	**67.6**	**63.4**	**66.3**
Human	82.3	91.2	86.3	89.0	86.9	89.5
Human–Machine Gap	3.7	5.4	**21.2**	**21.4**	**23.5**	**23.2**

Table 3: Exact Match (EM) and F1 scores on SQuAD 1.1 and 2.0. The gap between humans and the best tested model is much larger on SQuAD 2.0, suggesting there is a great deal of room for model improvement.

System	SQuAD 1.1 + TFIDF		SQuAD 1.1 + RULEBASED		SQuAD 2.0 dev	
	EM	F1	EM	F1	EM	F1
BNA	72.7	76.6	80.1	84.8	59.8	62.6
DocQA	75.6	79.2	80.8	84.8	61.9	64.8
DocQA + ELMo	**79.4**	**83.0**	**85.7**	**89.6**	**65.1**	**67.6**

Table 4: Exact Match (EM) and F1 scores on the SQuAD 2.0 development set, compared with SQuAD 1.1 with two types of automatically generated negative examples. SQuAD 2.0 is more challenging for current models.

5.3 Automatically generated negatives

Next, we investigated whether automatic ways of generating negative examples can also yield a challenging dataset. We trained and tested all three model architectures on SQuAD 1.1 augmented with either TFIDF or RULEBASED examples. To ensure a fair comparison with SQuAD 2.0, we generated training data by applying TFIDF or RULEBASED only to the 285 articles for which SQuAD 2.0 has unanswerable questions. We tested on the articles and answerable questions in the SQuAD 2.0 development set, adding unanswerable questions in a roughly one-to-one ratio with answerable ones. These results are shown in Table 4. The highest score on SQuAD 2.0 is 15.4 F1 points lower than the highest score on either of the other two datasets, suggesting that automatically generated negative examples are much easier for existing models to detect.

5.4 Plausible answers as distractors

Finally, we measured how often systems were fooled into answering the plausible but incorrect answers provided by crowdworkers for our unanswerable questions. For both computer systems and humans, roughly half of all wrong answers on unanswerable questions exactly matched the plausible answers. This suggests that the plausible answers do indeed serve as effective distractors. Full results are shown in Appendix A.2.

6 Discussion

SQuAD 2.0 forces models to understand whether a paragraph entails that a certain span is the answer to a question. Similarly, recognizing textual entailment (RTE) requires systems to decide whether a hypothesis is entailed by, contradicted by, or neutral with respect to a premise (Marelli et al., 2014; Bowman et al., 2015). Relation extraction systems must understand when a possible relationship between two entities is not entailed by the text (Zhang et al., 2017).

Jia and Liang (2017) created adversarial test examples that fool models trained on SQuAD 1.1. However, models that are trained on similar examples are not easily fooled by their method. In contrast, the adversarial examples in SQuAD 2.0 are difficult even for models trained on examples from the same distribution.

In conclusion, we have presented SQuAD 2.0, a challenging, diverse, and large-scale dataset that forces models to understand when a question cannot be answered given the context. We are optimistic that SQuAD 2.0 will encourage the development of new reading comprehension models that know what they don't know, and therefore understand language at a deeper level.

Reproducibility. All code, data, experiments are available on the CodaLab platform at `https://bit.ly/2rDHBgY`.

Acknowledgments. We would like to thank the anonymous reviewers, Arun Chaganty, Peng Qi, and Sharon Zhou for their constructive feedback. We are grateful to Durim Morina and Michael Bernstein for their help with the Daemo platform. This work was supported by funding from Facebook. R.J. is supported by an NSF Graduate Research Fellowship under Grant No. DGE-114747.

References

S. Bowman, G. Angeli, C. Potts, and C. D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Empirical Methods in Natural Language Processing (EMNLP)*.

C. Clark and M. Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723* .

S. N. Gaikwad, D. Morina, R. Nistala, M. Agarwal, A. Cossette, R. Bhanu, S. Savage, V. Narwal, K. Rajpal, J. Regino, et al. 2015. Daemo: A self-governed crowdsourcing marketplace. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. pages 101–102.

K. M. Hermann, T. Koisk, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

D. Hewlett, A. Lacoste, L. Jones, I. Polosukhin, A. Fandrianto, J. Han, M. Kelcey, and D. Berthelot. 2016. Wikireading: A novel large-scale language understanding task over Wikipedia. In *Association for Computational Linguistics (ACL)*.

M. Hu, Y. Peng, and X. Qiu. 2017. Reinforced mnemonic reader for machine comprehension. *arXiv* .

H. Huang, C. Zhu, Y. Shen, and W. Chen. 2018. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. In *International Conference on Learning Representations (ICLR)*.

R. Jia and P. Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Empirical Methods in Natural Language Processing (EMNLP)*.

M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Association for Computational Linguistics (ACL)*.

G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683* .

O. Levy, M. Seo, E. Choi, and L. Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Computational Natural Language Learning (CoNLL)*.

M. Marelli, S. Menini, M. Baroni, L. Bentivogli, R. bernardi, and R. Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Language Resources and Evaluation Conference (LREC)*.

T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Workshop on Cognitive Computing at NIPS*.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. 2018. Deep contextualized word representations. In *North American Association for Computational Linguistics (NAACL)*.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

M. Richardson, C. J. Burges, and E. Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 193–203.

M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv* .

A. Trischler, T. Wang, X. Yuan, J. Harris, A. Sordoni, P. Bachman, and K. Suleman. 2017. NewsQA: A machine comprehension dataset. In *Workshop on Representation Learning for NLP*.

M. Wang, N. A. Smith, and T. Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for QA. In *Empirical Methods in Natural Language Processing (EMNLP)*.

W. Wang, N. Yang, F. Wei, B. Chang, and M. Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Association for Computational Linguistics (ACL)*.

D. Weissenborn, G. Wiese, and L. Seiffe. 2017. Making neural QA as simple as possible but not simpler. In *Computational Natural Language Learning (CoNLL)*.

Y. Yang, W. Yih, and C. Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 2013–2018.

W. Yih, M. Chang, C. Meek, and A. Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Association for Computational Linguistics (ACL)*.

Y. Zhang, V. Zhong, D. Chen, G. Angeli, and C. D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Empirical Methods in Natural Language Processing (EMNLP)*.

A Supplementary material

A.1 Crowdsourcing details

Figure 2 shows the instructions that crowdworkers were given at the beginning of each question writing task. Figure 3 shows the interface they used to write unanswerable questions for each paragraph. In the interface, workers first write an unanswerable question, then highlight a plausible answer in the paragraph.

A.2 Plausible answers as distractors

As mentioned in Section 5.4, we measured how often systems were fooled into answering the plausible answers provided by crowdworkers for our unanswerable questions. For each system, we first isolated their false positive errors—cases where they predicted an answer to an unanswerable question—on the development set. Within this set of examples, we measured exact match and F1 scores between the system predictions and plausible answers. These numbers are shown in Table 5. Plausible answers account for roughly half of the false positive errors made by each of the computer systems, as well as by human answerers. We conclude that the plausible answers in our dataset do indeed serve their purpose of being distracting spans that could be mistaken for the correct answer.

	EM	F1
BNA	48.6	63.0
DocQA	55.0	69.9
DocQA +ELMo	54.9	69.2
Human	46.4	60.6

Table 5: Exact match (EM) and F1 scores between system predictions and plausible answers, in cases where the system made a false positive error.

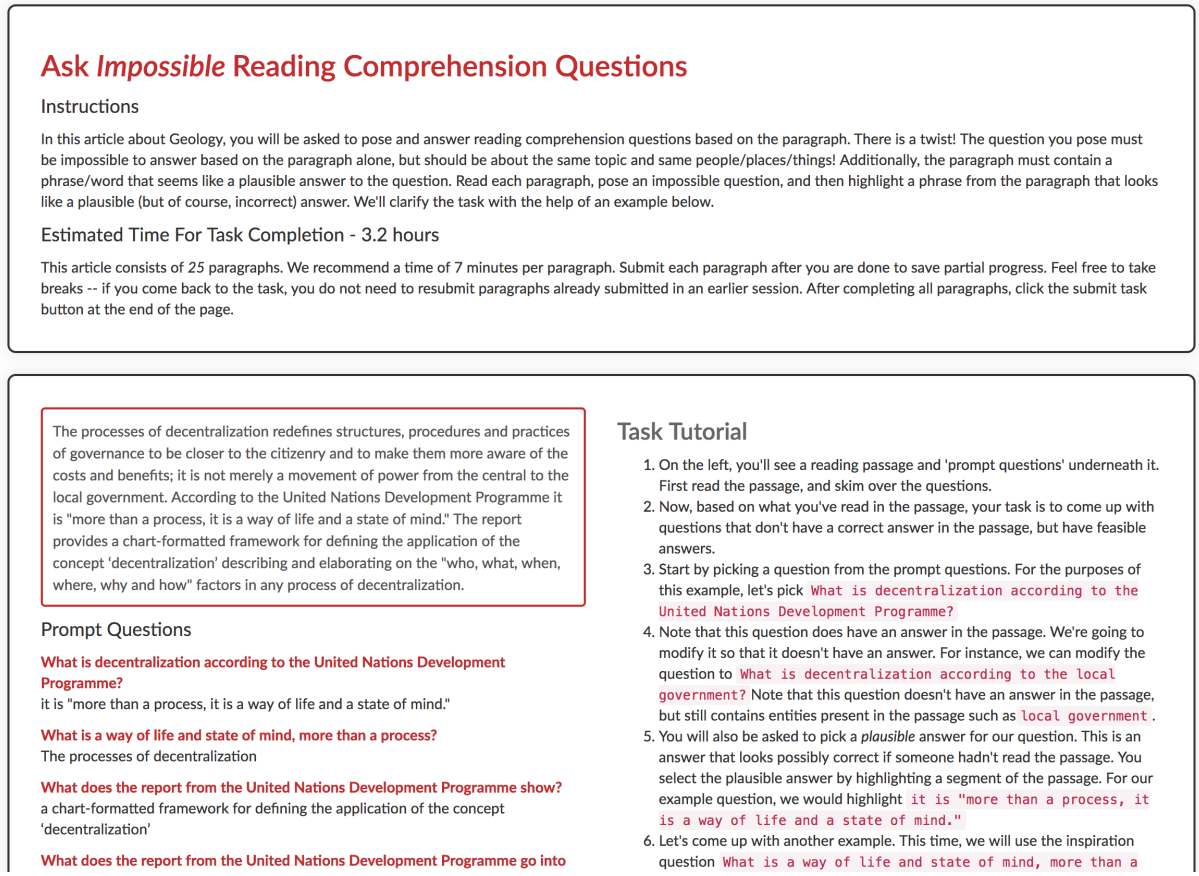


Figure 2: The instructions shown to crowdworkers at the beginning of each question writing task.

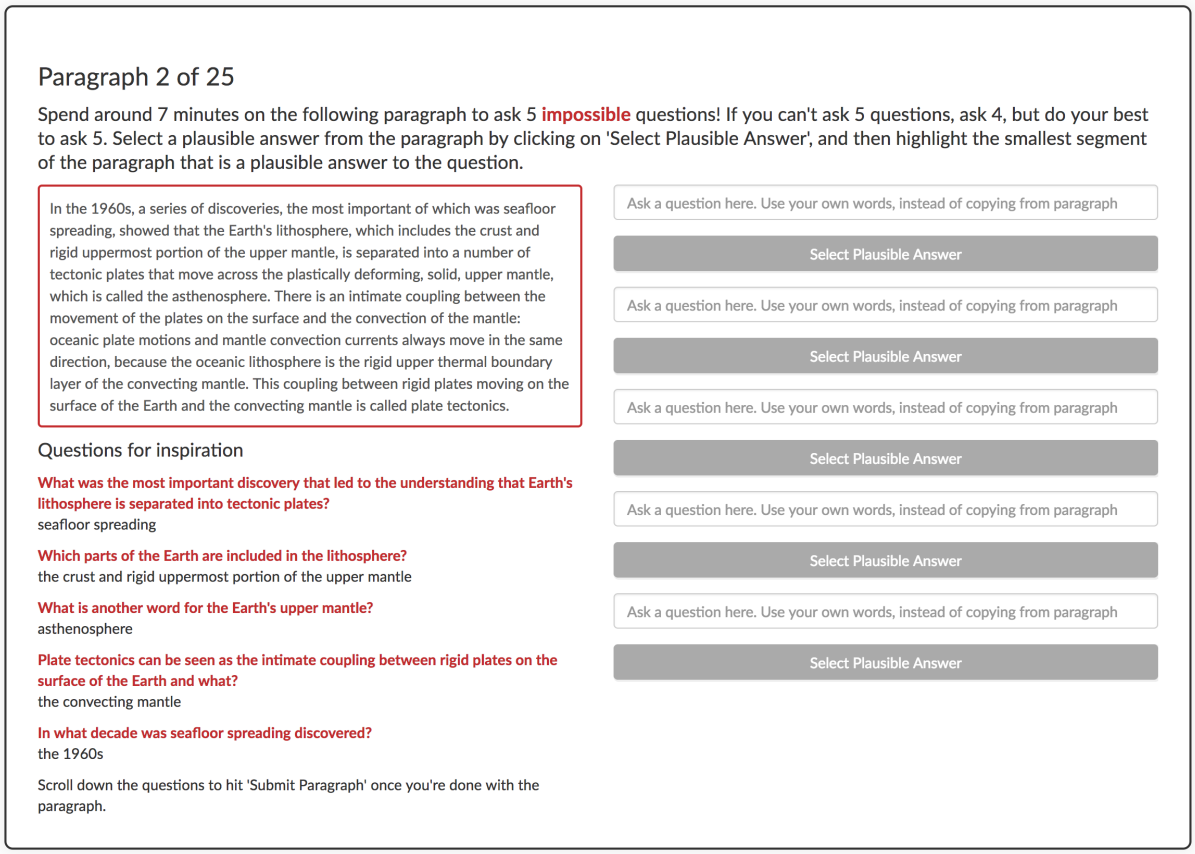


Figure 3: The interface crowdworkers used to write unanswerable questions and annotate plausible answers.

SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference

Rowan Zellers[♠] **Yonatan Bisk**[♠] **Roy Schwartz**[♠♡] **Yejin Choi**[♠♡]
[♠]Paul G. Allen School of Computer Science & Engineering, University of Washington
[♡]Allen Institute for Artificial Intelligence
{rowanz,ybisk,roysch,yejin}@cs.washington.edu
https://rowanzellers.com/swag

Abstract

Given a partial description like "she opened the hood of the car," humans can reason about the situation and anticipate what might come next ("then, she examined the engine"). In this paper, we introduce the task of grounded commonsense inference, unifying natural language inference and commonsense reasoning.

We present SWAG, a new dataset with 113k multiple choice questions about a rich spectrum of grounded situations. To address the recurring challenges of the annotation artifacts and human biases found in many existing datasets, we propose *Adversarial Filtering* (AF), a novel procedure that constructs a de-biased dataset by iteratively training an ensemble of stylistic classifiers, and using them to filter the data. To account for the aggressive adversarial filtering, we use state-of-the-art language models to massively oversample a diverse set of potential counterfactuals. Empirical results demonstrate that while humans can solve the resulting inference problems with high accuracy (88%), various competitive models struggle on our task. We provide comprehensive analysis that indicates significant opportunities for future research.

1 Introduction

When we read a story, we bring to it a large body of implicit knowledge about the physical world. For instance, given the context "on stage, a woman takes a seat at the piano," shown in Table 1, we can easily infer what the situation might *look* like: a woman is giving a piano performance, with a crowd watching her. We can furthermore infer her likely *next* action: she will most likely set her fingers on the piano keys and start playing.

This type of natural language inference requires commonsense reasoning, substantially broadening the scope of prior work that focused primarily on linguistic entailment (Chierchia and McConnell-Ginet, 2000). Whereas the dominant entailment paradigm asks if two natural language sentences (the 'premise' and the 'hypothesis') describe the same set of possible worlds (Dagan et al., 2006; Bowman et al., 2015), here we focus on whether a (multiple-choice) ending describes a possible (*future*) world that can be anticipated from the situation described in the premise, even when it is not strictly entailed. Making such inference necessitates a rich understanding about everyday physical situations, including object affordances (Gibson, 1979) and frame semantics (Baker et al., 1998).

A first step toward grounded commonsense inference with today's deep learning machinery is to create a large-scale dataset. However, recent work has shown that human-written datasets are susceptible to *annotation artifacts*: unintended stylistic patterns that give out clues for the gold labels (Gururangan et al., 2018; Poliak et al., 2018). As a result, models trained on such datasets with hu-

On stage, a woman takes a seat at the piano. She a) sits on a bench as her sister plays with the doll. b) smiles with someone as the music plays. c) is in the crowd, watching the dancers. **d) nervously sets her fingers on the keys.**
A girl is going across a set of monkey bars. She a) jumps up across the monkey bars. b) struggles onto the monkey bars to grab her head. **c) gets to the end and stands on a wooden plank.** d) jumps up and does a back flip.
The woman is now blow drying the dog. The dog **a) is placed in the kennel next to a woman's feet.** b) washes her face with the shampoo. c) walks into frame and walks towards the dog. d) tried to cut her face, so she is trying to do something very close to her face.

Table 1: Examples from SWAG; the correct answer is **bolded**. Adversarial Filtering ensures that stylistic models find all options equally appealing.

man biases run the risk of over-estimating the actual performance on the underlying task, and are vulnerable to adversarial or out-of-domain examples (Wang et al., 2018; Glockner et al., 2018).

In this paper, we introduce Adversarial Filtering (AF), a new method to automatically detect and reduce stylistic artifacts. We use this method to construct SWAG: an adversarial dataset with 113k multiple-choice questions. We start with pairs of temporally adjacent video captions, each with a context and a follow-up event that we *know* is physically possible. We then use a state-of-the-art language model fine-tuned on this data to massively oversample a diverse set of possible negative sentence endings (or *counterfactuals*). Next, we filter these candidate endings aggressively and adversarially using a committee of trained models to obtain a population of de-biased endings with similar stylistic features to the real ones. Finally, these filtered counterfactuals are validated by crowd workers to further ensure data quality.

Extensive empirical results demonstrate unique contributions of our dataset, complementing existing datasets for natural langauge inference (NLI) (Bowman et al., 2015; Williams et al., 2018) and commonsense reasoning (Roemmele et al., 2011; Mostafazadeh et al., 2016; Zhang et al., 2017). **First**, our dataset poses a new challenge of grounded commonsense inference that is easy for humans (88%) while hard for current state-of-the-art NLI models (<60%). **Second**, our proposed adversarial filtering methodology allows for cost-effective construction of a large-scale dataset while substantially reducing known annotation artifacts. The generality of adversarial filtering allows it to be applied to build future datasets, ensuring that they serve as reliable benchmarks.

2 SWAG: Our new dataset

We introduce a new dataset for studying physically grounded commonsense inference, called SWAG.[1] Our task is to predict which event is most likely to occur next in a video. More formally, a model is given a context $\boldsymbol{c} = (\boldsymbol{s}, \boldsymbol{n})$: a complete sentence $\boldsymbol{s}$ and a noun phrase $\boldsymbol{n}$ that begins a second sentence, as well as a list of possible verb phrase sentence endings $\boldsymbol{V} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_4\}$. See Figure 1 for an example triple $(\boldsymbol{s}, \boldsymbol{n}, \boldsymbol{v}_i)$. The model must then select the most appropriate verb phrase $\boldsymbol{v}_{\hat{i}} \in \boldsymbol{V}$.

[1] Short for **S**ituations **W**ith **A**dversarial **G**enerations.

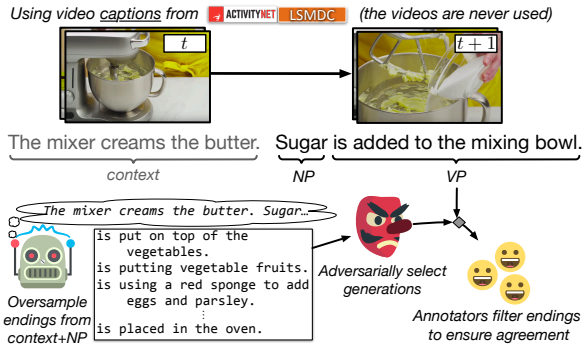


Figure 1: Overview of the data collection process. For a pair of sequential video captions, the second caption is split into noun and verb phrases. A language model generates many negative endings, of which a difficult subset are human-annotated.

Overview Our corpus consists of 113k multiple choice questions (73k training, 20k validation, 20k test) and is derived from pairs of consecutive video captions from ActivityNet Captions (Krishna et al., 2017; Heilbron et al., 2015) and the Large Scale Movie Description Challenge (LSMDC; Rohrbach et al., 2017). The two datasets are slightly different in nature and allow us to achieve broader coverage: ActivityNet contains 20k YouTube clips containing one of 203 activity types (such as doing gymnastics or playing guitar); LSMDC consists of 128k movie captions (audio descriptions and scripts). For each pair of captions, we use a constituency parser (Stern et al., 2017) to split the second sentence into noun and verb phrases (Figure 1).[2] Each question has a human-verified gold ending and 3 distractors.

3 A solution to annotation artifacts

In this section, we outline the construction of SWAG. We seek dataset diversity while minimizing *annotation artifacts*, conditional stylistic patterns such as length and word-preference biases. For many NLI datasets, these biases have been shown to allow shallow models (e.g. bag-of-words) obtain artificially high performance.

To avoid introducing easily "gamed" patterns, we present Adversarial Filtering (AF), a generally-applicable treatment involving the iterative refinement of a set of assignments to increase the entropy under a chosen model family. We then discuss how we generate counterfactual endings, and

[2] We filter out sentences with rare tokens ($\leq$3 occurrences), that are short ($l \leq 5$), or that lack a verb phrase.

Algorithm 1 Adversarial filtering (AF) of negative samples. During our experiments, we set $N^{easy} = 2$ for refining a population of $N^- = 1023$ negative examples to $k = 9$, and used a 80%/20% train/test split.

while convergence not reached **do**
- Split the dataset $\mathcal{D}$ randomly up into training and testing portions $\mathcal{D}^{tr}$ and $\mathcal{D}^{te}$.
- Optimize a model f_θ on $\mathcal{D}^{tr}$.

for index i in $\mathcal{D}^{te}$ **do**
- Identify easy indices: $\mathcal{A}_i^{easy} = \{j \in \mathcal{A}_i : f_\theta(x_i^+) > f_\theta(x_{i,j}^-)\}$
- Replace N^{easy} easy indices $j \in \mathcal{A}_i^{easy}$ with adversarial indices $k \notin \mathcal{A}_i$ satisfying $f_\theta(x_{i,k}^-) > f_\theta(x_{i,j}^-)$.

end for

end while

finally, the models used for filtering.

3.1 Formal definition

In this section, we formalize what it means for a dataset to be *adversarial*. Intuitively, we say that an adversarial dataset for a model f is one on which f will not generalize, even if evaluated on test data from the same distribution. More formally, let our input space be $\mathcal{X}$ and the label space be $\mathcal{Y}$. Our trainable classifier f, taking parameters θ is defined as $f_\theta : \mathcal{X} \to \mathbb{R}^{|\mathcal{Y}|}$. Let our dataset of size N be defined as $\mathcal{D} = \{(x_i, y_i)\}_{1 \le i \le N}$, and let the loss function over the dataset be $L(f_\theta, \mathcal{D})$. We say that a dataset is *adversarial* with respect to f if we expect high empirical error I over all leave-one-out train/test splits (Vapnik, 2000):

$$I(\mathcal{D}, f) = \frac{1}{N} \sum_{i=1}^{N} L(f_{\theta_i^\star}, \{(x_i, y_i)\}), \quad (1)$$

$$\text{where } \theta_i^\star = \operatorname*{argmin}_\theta L(f_\theta, \mathcal{D} \setminus \{(x_i, y_i)\}), \quad (2)$$

with regularization terms omitted for simplicity.

3.2 Adversarial filtering (AF) algorithm

In this section, we outline an approach for generating an adversarial dataset $\mathcal{D}$, effectively maximizing empirical error I with respect to a family of trainable classifiers f. Without loss of generality, we consider the situation where we have N *contexts*, each associated with a single positive example $(x_i^+, 1) \in \mathcal{X} \times \mathcal{Y}$, and a large population of context-specific negative examples $(x_{i,j}^-, 0) \in \mathcal{X} \times \mathcal{Y}$, where $1 \le j \le N^-$ for each i. For instance, the negative examples could be incorrect relations in knowledge-base completion (Socher et al., 2013), or all words in a dictionary for a single-word cloze task (Zweig and Burges, 2011).

Our goal will be to filter the population of negative examples for each instance i to a size of $k \ll N^-$. This will be captured by returning a set of *assignments* $\mathcal{A}$, where for each instance the assignment will be a k-subset $\mathcal{A}_i = [1 \dots N^-]^k$. The filtered dataset will then be:

$$\mathcal{D}^{AF} = \{(x_i, 1), \{(x_{i,j}^-, 0)\}_{j \in \mathcal{A}_i}\}_{1 \le i \le N} \quad (3)$$

Unfortunately, optimizing $I(\mathcal{D}^{AF}, f)$ is difficult as $\mathcal{A}$ is global and non-differentiable. To address this, we present Algorithm 1. On each iteration, we split the data into dummy 'train' and 'test' splits. We train a model f on the training portion and obtain parameters θ, then use the remaining test portion to reassign the indices of $\mathcal{A}$. For each context, we replace some number of 'easy' negatives in $\mathcal{A}$ that f_θ classifies correctly with 'adversarial' negatives outside of $\mathcal{A}$ that f_θ misclassifies.

This process can be thought of as increasing the overall entropy of the dataset: given a strong model f_θ that is compatible with a random subset of the data, we aim to ensure it cannot generalize to the held-out set. We repeat this for several iterations to reduce the generalization ability of the model family f over arbitrary train/test splits.

3.3 Generating candidate endings

To generate counterfactuals for SWAG, we use an LSTM (Hochreiter and Schmidhuber, 1997) language model (LM), conditioned on contexts from video captions. We first pretrain on BookCorpus (Zhu et al., 2015), then finetune on the video caption datasets. The architecture uses standard best practices and was validated on held-out perplexity of the video caption datasets; details are in the appendix. We use the LM to sample N^-=1023 unique endings for a partial caption.[3]

Importantly, we *greedily* sample the endings, since beam search decoding biases the generated endings to be of lower perplexity (and thus easily distinguishable from found endings). We find this process gives good counterfactuals: the generated endings tend to use *topical* words, but often make little sense physically, making them perfect for our task. Further, the generated endings are marked as "gibberish" by humans only 9.1% of the time (Sec 3.5); in that case the ending is filtered out.

[3] To ensure that the LM generates unique endings, we split the data into five validation folds and train five separate LMs, one for each set of training folds. This means that each LM never sees the found endings during training.

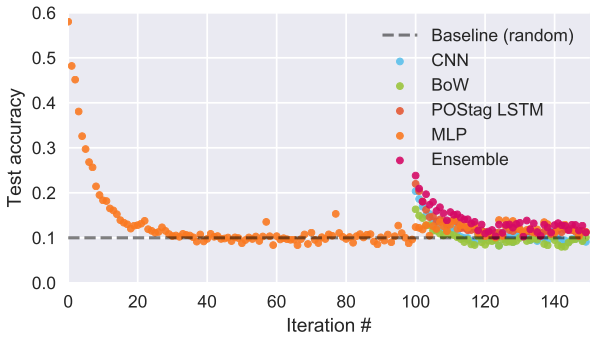


Figure 2: Test accuracy by AF iteration, under the negatives given by $\mathcal{A}$. The accuracy drops from around 60% to close to random chance. For efficiency, the first 100 iterations only use the MLP.

3.4 Stylistic models for adversarial filtering

In creating SWAG, we designed the model family f to pick up on low-level *stylistic features* that we posit should not be predictive of whether an event happens next in a video. These stylistic features are an obvious case of annotation artifacts (Cai et al., 2017; Schwartz et al., 2017).[4] Our final classifier is an ensemble of four stylistic models:

1. A multilayer perceptron (MLP) given LM perplexity features and context/ending lengths.
2. A bag-of-words model that averages the word embeddings of the second sentence as features.
3. A one-layer CNN, with filter sizes ranging from 2-5, over the second sentence.
4. A bidirectional LSTM over the 100 most common words in the second sentence; uncommon words are replaced by their POS tags.

We ensemble the models by concatenating their final representations and passing it through an MLP. On every adversarial iteration, the ensemble is trained jointly to minimize cross-entropy.

The accuracies of these models (at each iteration, evaluated on a 20% split of the test dataset before indices of $\mathcal{A}$ get remapped) are shown in Figure 2. Performance decreases from 60% to close to random chance; moreover, confusing the perplexity-based MLP is not sufficient to lower performance of the ensemble. Only once the other stylistic models are added does the ensemble accuracy drop substantially, suggesting that our approach is effective at reducing stylistic artifacts.

[4]A broad definition of annotation artifacts might include aspects besides lexical/stylistic features: for instance, certain events are less likely semantically regardless of the context (e.g. riding a horse using a hose). For this work, we erred more conservatively and only filtered based on style.

Imagine that you are watching a video clip. The clip has a caption, but it is missing the final phrase. Please choose the best 2 caption endings, and classify each as:

- **likely**, if it completes the caption in a reasonable way;
- **unlikely**, if it sounds ridiculous or impossible;
- **gibberish** if it has such serious errors that it doesn't feel like a valid English sentence.

Example: Someone is shown sitting on a fence and talking to the camera while pointing out horses. Someone

- stands in front of a podium. (**likely**, **second best**)
- rides a horse using a hose. (**unlikely**)
- is shown riding a horse. (**likely**, **best**)
- , the horse in a plaza field. (**gibberish**)

Figure 3: Mechanical Turk instructions (abridged).

3.5 Human verification

The final data-collection step is to have humans verify the data. Workers on Amazon Mechanical Turk were given the caption context, as well as six candidate endings: one found ending and five adversarially-sampled endings. The task was twofold: Turkers ranked the endings independently as likely, unlikely, or gibberish, and selected the best and second best endings (Fig 3).

We obtained the correct answers to each context in two ways. If a Turker ranks the found ending as either best or second best (73.7% of the time), we add the found ending as a gold example, with negatives from the generations not labelled best or gibberish. Further, if a Turker ranks a generated ending as best, and the found ending as second best, then we have reason to believe that the generation is good. This lets us add an additional training example, consisting of the generated best ending as the gold, and remaining generations as negatives.[5] Examples with $\leq$3 non-gibberish endings were filtered out.[6]

We found after 1000 examples that the annotators tended to have high agreement, also generally choosing found endings over generations (see Table 2). Thus, we collected the remaining 112k examples with one annotator each, periodically verifying that annotators preferred the found endings.

4 Experiments

In this section, we evaluate the performance of various NLI models on SWAG. Recall that models

[5]These two examples share contexts. To prevent biasing the test and validation sets, we didn't perform this procedure on answers from the evaluation sets' context.

[6]To be data-efficient, we reannotated filtered-out examples by replacing gibberish endings, as well as generations that outranked the found ending, with candidates from $\mathcal{A}$.

	Label distribution by ending type		Inter-annotator agreement	
Labels	Found end	Gen. end	α	ppa
Best	53.5%	9.3%		
Second Best	20.2%	15.9%	0.43	72%
Neither	26.3%	74.8%		
Likely	80.3%	33.3%		
Unlikely	19.0%	57.5%	0.39	64%
Gibberish	0.7%	9.1%		

Table 2: Annotators tend to label the found ending as likely and within the top 2 (column 2), in other cases the example is filtered out. Both label groups have high inter-annotator agreement, in terms of Krippendorff's α and pairwise percent agreement.

for our dataset take the following form: given a sentence and a noun phrase as context $\boldsymbol{c} = (\boldsymbol{s}, \boldsymbol{n})$, as well as a list of possible verb phrase endings $\boldsymbol{V} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_4\}$, a model f_θ must select a verb $\hat{i}$ that hopefully matches i_{gold}:

$$\hat{i} = \underset{i}{\operatorname{argmax}}\, f_\theta(\boldsymbol{s}, \boldsymbol{n}, \boldsymbol{v}_i) \quad (4)$$

To study the amount of bias in our dataset, we also consider models that take as input just the ending verb phrase $\boldsymbol{v}_i$, or the entire second sentence $(\boldsymbol{n}, \boldsymbol{v}_i)$. For our learned models, we train f by minimizing multi-class cross-entropy. We consider three different types of word representations: 300d GloVe vectors from Common Crawl (Pennington et al., 2014), 300d Numberbatch vectors retrofitted using ConceptNet relations (Speer et al., 2017), and 1024d ELMo contextual representations that show improvement on a variety of NLP tasks, including standard NLI (Peters et al., 2018). We follow the final dataset split (see Section 2) using two training approaches: training on the found data, and the found and highly-ranked generated data. See the appendix for more details.

4.1 Unary models

The following models predict labels from *a single span* of text as input; this could be the ending only, the second sentence only, or the full passage.

a. **fastText** (Joulin et al., 2017): This library models a single span of text as a bag of n-grams, and tries to predict the probability of an ending being correct or incorrect independently.[7]

b. **Pretrained sentence encoders** We consider two types of pretrained RNN sentence encoders, SkipThoughts (Kiros et al., 2015) and InferSent (Conneau et al., 2017). SkipThoughts was trained by predicting adjacent sentences in book data, whereas InferSent was trained on supervised NLI data. For each second sentence (or just the ending), we feed the encoding into an MLP.

c. **LSTM sentence encoder** Given an arbitrary span of text, we run a two-layer BiLSTM over it. The final hidden states are then max-pooled to obtain a fixed-size representation, which is then used to predict the potential for that ending.

4.2 Binary models

The following models predict labels from *two spans* of text. We consider two possibilties for these models: using just the second sentence, where the two text spans are $\boldsymbol{n}, \boldsymbol{v}_i$, or using the context and the second sentence, in which case the spans are $\boldsymbol{s}, (\boldsymbol{n}, \boldsymbol{v}_i)$. The latter case includes many models developed for the NLI task.

d. **Dual Bag-of-Words** For this baseline, we treat each sentence as a bag-of-embeddings $(\mathbf{c}, \mathbf{v}_i)$. We model the probability of picking an ending i using a bilinear model: $\text{softmax}_i(\mathbf{c}\mathbf{W}\mathbf{v}_i^T)$.[8]

e. **Dual pretrained sentence encoders** Here, we obtain representations from SkipThoughts or InferSent for each span, and compute their pairwise compatibility using either 1) a bilinear model or 2) an MLP from their concatenated representations.

f. **SNLI inference** Here, we consider two models that do well on SNLI (Bowman et al., 2015): Decomposable Attention (Parikh et al., 2016) and ESIM (Chen et al., 2017). We use pretrained versions of these models (with ELMo embeddings) on SNLI to obtain 3-way entailment, neutral, and contradiction probabilities for each example. We then train a log-linear model using these 3-way probabilities as features.

g. **SNLI models (retrained)** Here, we train ESIM and Decomposable Attention on our dataset: we simply change the output layer size to 1 (the potential of an ending $\boldsymbol{v}_i$) with a softmax over i.

4.3 Other models

We also considered the following models:

h. **Length**: Although length was used by the adversarial classifier, we want to verify that human validation didn't reintroduce a length bias. For this baseline, we always choose the shortest ending.

i. **ConceptNet** As our task requires world knowledge, we tried a rule-based system on top of the

[7] The fastText model is trained using binary cross-entropy; at test time we extract the prediction by selecting the ending with the highest positive likelihood under the model.

[8] We also tried using an MLP, but got worse results.

			Ending only				2nd sentence only				Context+2nd sentence			
			found only		found+gen		found only		found+gen		found only		found+gen	
		Model	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test
	misc	Random	25.0	25.0	25.0	25.0	25.0	25.0	25.0	25.0	25.0	25.0	25.0	25.0
		Length	26.7	27.0	26.7	27.0								
		ConceptNet					26.0	26.0	26.0	26.0				
Unary models		fastText	27.5	26.9	29.9	29.0	29.2	27.8	29.8	29.0	29.4	28.0	30.3	29.8
	Sentence encoders	SkipThoughts	32.4	32.1	32.2	31.8	33.0	32.4	32.8	32.3				
		InferSent	30.6	30.2	32.0	31.9	33.2	32.0	34.0	32.6				
	LSTM sequence model	LSTM+GloVe	31.9	31.8	32.9	32.4	32.7	32.4	34.3	33.5	43.1	43.6	45.6	45.7
		LSTM+Numberbatch	32.4	32.6	32.3	31.9	31.9	31.9	34.1	32.8	39.9	40.2	41.2	40.5
		LSTM+ELMo	**43.6**	**42.9**	**43.3**	**42.3**	**47.4**	**46.7**	**46.3**	**46.0**	51.4	50.6	51.3	50.4
Binary models	DualBoW	DualBoW+GloVe					31.3	31.3	31.9	31.2	34.5	34.7	32.9	33.1
		DualBoW+Numberbatch					31.9	31.4	31.6	31.3	35.1	35.1	34.2	34.1
	Dual sentence encoders	SkipThoughts-MLP					34.6	33.9	36.2	35.5	33.4	32.3	37.4	36.4
		SkipThoughts-Bilinear					36.0	35.7	34.7	34.5	36.5	35.6	35.3	34.9
		InferSent-MLP					32.9	32.1	32.8	32.7	35.9	36.2	39.5	39.4
		InferSent-Bilinear					32.0	31.3	31.6	31.3	40.5	40.3	39.0	38.4
	SNLI inference	SNLI-ESIM									36.4	36.1	36.2	36.0
		SNLI-DecompAttn									35.8	35.8	35.8	35.7
	SNLI models (retrained)	DecompAttn+GloVe					29.8	30.3	31.1	31.7	47.4	47.6	48.5	48.6
		DecompAttn+Numberbatch					32.4	31.7	32.5	31.9	47.4	48.0	48.0	48.3
		DecompAttn+ELMo					43.4	43.4	40.6	40.3	47.7	47.3	46.0	45.4
		ESIM+GloVe					34.8	35.1	36.3	36.7	51.9	52.7	52.5	52.5
		ESIM+Numberbatch					33.1	32.6	33.0	32.4	46.5	46.4	44.0	44.6
		ESIM+ELMo					46.0	45.7	45.9	44.8	**59.1**	**59.2**	**58.7**	**58.5**
	Human	1 turker										82.8		
		3 turkers										85.1		
		5 turkers										**88.0**		
		Expert										85.0		

Table 3: Performance of all models in accuracy (%). All models substantially underperform humans, although performance increases as more context is provided (left to right). We optionally train on found endings only, or found and human-validated generated endings (found+gen).

ConceptNet knowledge base (Speer et al., 2017). For an ending sentence, we use the spaCy dependency parser to extract the head verb and its dependent object. The ending score is given by the number of ConceptNet causal relations[9] between synonyms of the verb and synonyms of the object.

j. Human performance To benchmark human performance, five Mechanical Turk workers were asked to answer 100 dataset questions, as did an 'expert' annotator (the first author of this paper). Predictions were combined using a majority vote.

4.4 Results

We present our results in Table 3. The best model that only uses the ending is the LSTM sequence model with ELMo embeddings, which obtains 43.6%. This model, as with most models studied, greatly improves with more context: by 3.1% when given the initial noun phrase, and by an additional 4% when also given the first sentence.

Further improvement is gained from models that compute pairwise representations of the inputs. While the simplest such model, DualBoW, obtains only 35.1% accuracy, combining InferSent sentence representations gives 40.5% accuracy (InferSent-Bilinear). The best results come from pairwise NLI models: when fully trained on SWAG, ESIM+ELMo obtains 59.2% accuracy.

When comparing machine results to human results, we see there exists a lot of headroom. Though there likely is some noise in the task, our results suggest that humans (even untrained) converge to a consensus. Our in-house "expert" annotator is outperformed by an ensemble of 5 Turk workers (with 88% accuracy); thus, the effective upper bound on our dataset is likely even higher.

5 Analysis

5.1 SWAG versus existing NLI datasets

The past few years have yielded great advances in NLI and representation learning, due to the availability of large datasets like SNLI and MultiNLI

[9] We used the relations 'Causes', 'CapableOf', 'ReceivesAction', 'UsedFor', and 'HasSubevent'. Though their coverage is low (30.4% of questions have an answer with $\geq$1 causal relation), the more frequent relations in ConceptNet, such as 'IsA', at best only indirectly relate to our task.

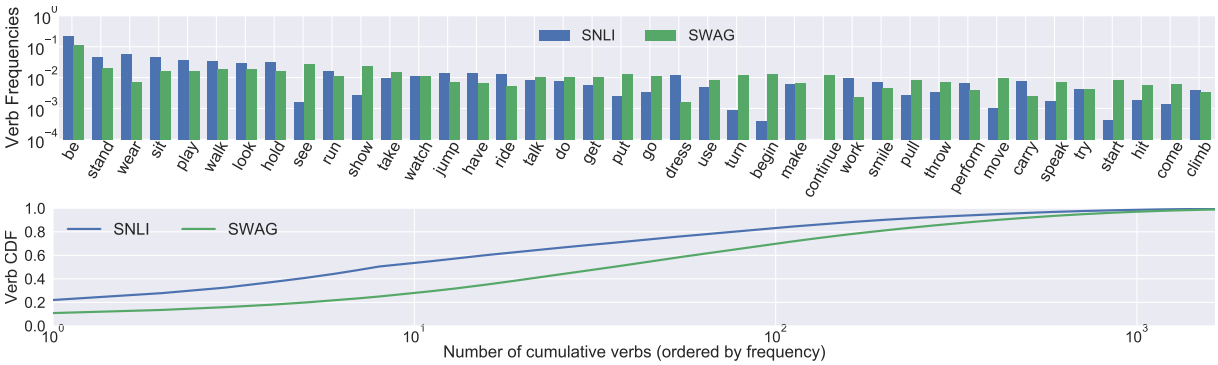


Figure 4: Top: Distribution of the 40 top verbs in the union of SNLI and SWAG. Our dataset shows a greater variety of dynamic verbs, such as "move", as well as temporal verbs such as "start" and "come." "Continue" is cut off for SNLI (it has frequency $6 \cdot 10^{-5}$). Bottom: CDF for verbs in SNLI and SWAG.

(Bowman et al., 2015; Williams et al., 2018). With the release of SWAG, we hope to continue this trend, particularly as our dataset largely has the same input/output format as other NLI datasets. We observe three key differences between our dataset and others in this space:

First, as noted in Section 1, SWAG requires a unique type of temporal reasoning. A state-of-the-art NLI model such as ESIM, when bottlenecked through the SNLI notion of entailment (SNLI-ESIM), only obtains 36.1% accuracy.[10] This implies that these datasets necessitate different (and complementary) forms of reasoning.

Second, our use of videos results in wide coverage of dynamic and temporal situations Compared with SNLI, with contexts from Flickr30K (Plummer et al., 2017) image captions, SWAG has more active verbs like 'pull' and 'hit,' and fewer static verbs like 'sit' and 'wear' (Figure 4).[11]

Third, our dataset suffers from few lexical biases. Whereas fastText, a bag of n-gram model, obtains 67.0% accuracy on SNLI versus a 34.3% baseline (Gururangan et al., 2018), fastText obtains only 29.0% accuracy on SWAG.[12]

5.2 Error analysis

We sought to quantify how human judgments differ from the best studied model, ESIM+ELMo. We randomly sampled 100 validation questions that ESIM+ELMo answered incorrectly, for each extracting both the gold ending and the model's preferred ending. We asked 5 Amazon Mechanical Turk workers to pick the better ending (of which they preferred the gold endings 94% of the time) and to select one (or more) multiple choice reasons explaining why the chosen answer was better.

Reason	Explanation	Freq.
Situational	The good ending is better *in context*.	53.7%
Plausibility	The bad ending is implausible *regardless of context*.	14.4%
Novelty	The bad ending seems redundant; it is entailed by the context.	1.8%
Weirdness	The bad ending is semantically or grammatically malformed, e.g. 'the man is getting out of the horse.'	18.1%
Ambiguous	Both endings seem equally likely.	12.0%

Table 4: Justifications for ranking the gold answer over a wrong answer chosen by ESIM+ELMo.

The options, and the frequencies, are outlined in Table 4. The most common reason for the turkers preferring the correct answer is situational (52.3% of the time), followed by weirdness (17.5%) and plausibility (14.4%). This suggests that ESIM+ELMo already does a good job at filtering out weird and implausible answers, with the main bottleneck being grounded physical understanding. The ambiguous percentage is also relatively low (12.0%), implying significant headroom.

5.3 Qualitative examples

Last, we show several qualitative examples in Table 5. Though models can do decently well by identifying complex alignment patterns between the two sentences (e.g. being "up a tree" implies that "tree" is the end phrase), the incorrect model predictions suggest this strategy is insuffi-

[10] The weights of SNLI-ESIM pick up primarily on entailment probability (0.59), as with neutral (0.46), while contradiction is negatively correlated (-.42).

[11] Video data has other language differences; notably, character names in LSMDC were replaced by 'someone'

[12] The most predictive individual words on SWAG are infrequent in number: 'dotted' with P(+|dotted) = 77% with 10.3 counts, and P(−|similar) = 81% with 16.3 counts. (Counts from negative endings were discounted 3x, as there are 3 times as many negative endings as positive endings).

A waiter brings a fork. The waiter **a) starts to step away. (74.76%)** b) adds spaghetti to the table. (21.57%) c) brings a bunch of pie to the food (2.67%) d) drinks from the mug in the bowl. (0.98%)	He is up a tree. Someone **a) stands underneath the tree. (97.44%)** b) is at a pool table holding a cup. (1.14%) c) grabs a flower from a paper. (0.96%) d) is eating some cereal. (0.45%)
An old man rides a small bumper car. Several people a) get in the parking lot. (76.58%) b) wait in the car. (15.28%) **c) get stuck with other bumper cars. (6.75%)** d) are running down the road. (1.39%)	He pours the raw egg batter into the pan. He a) drops the tiny pan onto a plate. (93.48%) **b) lifts the pan and moves it around to shuffle the eggs. (4.94%)** c) stirs the dough into a kite. (1.53%) d) swirls the stir under the adhesive. (0.05%)

Table 5: Example questions answered by the best model, ESIM+Elmo, sorted by model probability. Correct model predictions are in **blue**, incorrect model predictions are red. The right answers are **bolded**.

cient. For instance, answering "An old man rides a small bumper car" requires knowledge about *bumper cars* and how they differ from regular cars: bumper cars are tiny, don't drive on roads, and don't work in parking lots, eliminating the alternatives. However, this knowledge is difficult to extract from existing corpora: for instance, the ConceptNet entry for Bumper Car has only a single relation: bumper cars are a type of vehicle. Other questions require intuitive physical reasoning: e.g, for "he pours the raw egg batter into the pan," about what happens next in making an omelet.

5.4 Where to go next?

Our results suggest that SWAG is a challenging testbed for NLI models. However, the adversarial models used to filter the dataset are purely stylistic and focus on the second sentence; thus, subtle artifacts still likely remain in our dataset. These patterns are ostensibly picked up by the NLI models (particularly when using ELMo features), but the large gap between machine and human performance suggests that more is required to solve the dataset. As models are developed for commonsense inference, and more broadly as the field of NLP advances, we note that AF can be used again to create a more adversarial version of SWAG using better language models and AF models.

6 Related Work

Entailment NLI There has been a long history of NLI benchmarks focusing on linguistic entailment (Cooper et al., 1996; Dagan et al., 2006; Marelli et al., 2014; Bowman et al., 2015; Lai et al., 2017; Williams et al., 2018). Recent NLI datasets in particular have supported learning broadly-applicable sentence representations (Conneau et al., 2017); moreover, models trained on these datasets were used as components for performing better video captioning (Pasunuru and Bansal, 2017), summarization (Pasunuru and Bansal, 2018), and generation (Holtzman et al., 2018), confirming the importance of NLI research. The NLI task requires a variety of commonsense knowledge (LoBue and Yates, 2011), which our work complements. However, previous datasets for NLI have been challenged by unwanted annotation artifacts, (Gururangan et al., 2018; Poliak et al., 2018) or scale issues. Our work addresses these challenges by constructing a new NLI benchmark focused on grounded commonsense reasoning, and by introducing an adversarial filtering mechanism that substantially reduces known and easily detectable annotation artifacts.

Commonsense NLI Several datasets have been introduced to study NLI beyond linguistic entailment: for inferring likely causes and endings given a sentence (COPA; Roemmele et al., 2011), for choosing the most sensible ending to a short story (RocStories; Mostafazadeh et al., 2016; Sharma et al., 2018), and for predicting likelihood of a hypothesis by regressing to an ordinal label (JOCI; (Zhang et al., 2017)). These datasets are relatively small: 1k examples for COPA and 10k cloze examples for RocStories.[13] JOCI increases the scale by generating the hypotheses using a knowledge graph or a neural model. In contrast to JOCI where the task was formulated as a regression task on the degree of plausibility of the hypothesis, we frame commonsense inference as a multiple choice question to reduce the potential ambiguity in the labels and to allow for direct comparison between machines and humans. In addition, SWAG's use of adversarial filtering increases diversity of situations and counterfactual generation quality.

[13] For RocStories, this was by design to encourage learning from the larger corpus of 98k sensible stories.

Last, another related task formulation is sentence completion or cloze, where the task is to predict a single word that is removed from a given context (Zweig and Burges, 2011; Paperno et al., 2016).[14] Our work in contrast requires longer textual descriptions to reason about.

Vision datasets Several resources have been introduced to study temporal inference in vision. The Visual Madlibs dataset has 20k image captions about hypothetical next/previous events (Yu et al., 2015); similar to our work, the test portion is multiple-choice, with counterfactual answers retrieved from similar images and verified by humans. The question of 'what will happen next?' has also been studied in photo albums (Huang et al., 2016), videos of team sports, (Felsen et al., 2017) and egocentric dog videos (Ehsani et al., 2018). Last, annotation artifacts are also a recurring problem for vision datasets such as Visual Genome (Zellers et al., 2018) and Visual QA (Jabri et al., 2016); recent work was done to create a more challenging VQA dataset by annotating complementary image pairs (Goyal et al., 2016).

Reducing gender/racial bias Prior work has sought to reduce demographic biases in word embeddings (Zhang et al., 2018) as well as in image recognition models (Zhao et al., 2017). Our work has focused on producing a dataset with minimal annotation artifacts, which in turn helps to avoid some gender and racial biases that stem from elicitation (Rudinger et al., 2017). However, it is not perfect in this regard, particularly due to biases in movies (Schofield and Mehr, 2016; Sap et al., 2017). Our methodology could potentially be extended to construct datasets free of (possibly intersectional) gender or racial bias.

Physical knowledge Prior work has studied learning grounded knowledge about objects and verbs: from knowledge bases (Li et al., 2016), syntax parses (Forbes and Choi, 2017), word embeddings (Lucy and Gauthier, 2017), and images and dictionary definitions (Zellers and Choi, 2017). An alternate thread of work has been to learn scripts: high-level representations of event chains (Schank and Abelson, 1975; Chambers and Jurafsky, 2009). SWAG evaluates both of these strands.

[14] Prior work on sentence completion filtered negatives with heuristics based on LM perplexities. We initially tried something similar, but found the result to still be gameable.

7 Conclusion

We propose a new challenge of physically situated commonsense inference that broadens the scope of natural language inference (NLI) with commonsense reasoning. To support research toward commonsense NLI, we create a large-scale dataset SWAG with 113k multiple-choice questions. Our dataset is constructed using Adversarial Filtering (AF), a new paradigm for robust and cost-effective dataset construction that allows datasets to be constructed at scale while automatically reducing annotation artifacts that can be easily detected by a committee of strong baseline models. Our adversarial filtering paradigm is general, allowing potential applications to other datasets that require human composition of question answer pairs.

Acknowledgements

We thank the anonymous reviewers, members of the ARK and xlab at the University of Washington, researchers at the Allen Institute for AI, and Luke Zettlemoyer for their helpful feedback. We also thank the Mechanical Turk workers for doing a fantastic job with the human validation. This work was supported by the National Science Foundation Graduate Research Fellowship (DGE-1256082), the NSF grant (IIS-1524371, 1703166), the DARPA CwC program through ARO (W911NF-15-1-0543), the IARPA DIVA program through D17PC00343, and gifts by Google and Facebook. The views and conclusions contained herein are those of the authors and should not be interpreted as representing endorsements of IARPA, DOI/IBC, or the U.S. Government.

A Appendix

A.1 More detail about video datasets

As mentioned in the main paper, we obtained contexts and found endings from video data. The videos in the ActivityNet dataset are already broken up into into clips. However, the LSMDC dataset contains captions for the entire movie, so it is possible that temporally adjacent captions describe events that are far apart in time. Thus, we don't include any pair of captions that have a time-difference of more than 25 seconds.

In addition to the datasets we used, we also considered the DiDeMo dataset, which consists of (often several) referring expressions in a video (Hen-

dricks et al., 2017). However, many of the referring expressions are themselves sentence fragments, (e.g. "first time we see people" so we ultimately did not use this dataset.) Additionally, we considered the Visual Madlibs dataset (Yu et al., 2015), as it contains 10k hypothetical captions written by Mechanical Turk workers about what might happen *next* given an image. However, these captions are fundamentally different from the rest of the data (as they're about what *might*) happen next; as a result, they use different types of language. They also have different tenses versus the other datasets that we considered (e.g. past tense), as a result of the "Mad-libs" style of data collection.

A.2 Details of the language model

Our language model follows standard best practices: the input and output embedding layers are tied (Inan et al., 2017; Press and Wolf, 2017), all embedding and hidden layers are set to 512, and we used recurrent dropout (Gal and Ghahramani, 2016) on the hidden states and embedding layer. We additionally train a *backwards* language model alongside the forward language model, and they share embedding parameters. This adds extra supervision to the embedding layer and gives us another way to score candidate generations. We first pretrain the language model for two epochs on pairs of two sentences in the Toronto Books dataset (Zhu et al., 2015), and then train on sentence pairs from ActivityNet Captions and LSMDC, validating on held-out perplexity. For optimization, we use Adam (Kingma and Ba, 2015) with a learning rate of 10^{-3} and clip gradients to norm 1.0.

All of the above details were validated using perplexity on a held-out set of the video datasets during early experimentation. Our final development set forward perplexity was 31.2 and backward perplexity was 30.4. We tried more complicated language modeling architectures, such as from (Józefowicz et al., 2016), but ended up not seeing an improvement due to overfitting.

A.3 Language model features for the MLP, during adversarial filtering

We obtained LM perplexity features to be used during adversarial filtering in the following ways, using both directions of the bidirectional language model. We extract perplexities for the context by itself (going forward), the ending given the context (going forward), the context given the ending (going backward), and the ending by itself (going backward). We also extract the probability of the final generated token going forward, since sentences sometimes reach the length limit of 25 tokens and end unnaturally.

A.4 Refinining the generated answers to four distractors

In the main paper, we noted that we started with 1023 negatives per example, which the adversarial filtering process filtered down to 9. Five of these were passed to mechanical turk workers, and we were left with anywhere between 0 and 4 of these per example as "distractors." (Note that we always were filtering out the second best option that the was selected by the turkers). This means that for many of our examples (62%) we actually have a fourth distractor. In these cases, we sorted the distractors by their "unlikely/likely" score, so that the fourth distractor was the one deemed most likely. We still provided the fourth distractor in the training set to be possibly used in future work, however we didn't train on it for simplicity.

A.5 More information about Mechanical turk

We used several tricks to keep the interannotator agreement high (with a pairwise percent agreement of 79% at classifying an ending as either in the Top 2). First, we had a screening HIT where turkers were given detailed instructions for the task, and only the best-scoring turk workers qualified for the remaining HITs. Second, we periodically dequalified turkers that had a low agreement with the gold endings: any turk worker with an accuracy of less than 55% of classifying the "gold" ending as the best or second best, over 10 or more HITs, had the qualification taken away. We also gave small bonuses to turkers with high accuracy.

During our crowdsourcing, we tried to pay the Turkers a fair wage (median $8.57 per hour) and they left positive comments for us on TurkOpticon and TurkerView. The total dataset cost was $23,000, or an average of 20 cents per example.

A.6 Implementation details of the models considered

We implemented the neural models in PyTorch using the AllenNLP library (Gardner et al., 2018). Our experiments use the Adam optimizer (Kingma and Ba, 2015), with a learning rate of 10^{-3} and

Questions with only generated endings	25,618
Questions with one original ending	87,939
Questions in total	113,557
Sentence pairs from ActivityNet	51,439
Sentence pairs from LSMDC	62,118
Unique contexts	92,221
Unique endings	452,683

Table 6: Statistics of SWAG.

Freq	Topic words
5.0%	ball, pull, hit, wall, inside, time, game, rope, team
4.9%	window, red, long, drink, bowl, ingredient, mix
6.1%	arm, speak, appear, climb, tree, roll, like, roof, edge
4.0%	water, bar, board, blue, boat, fly, river, join, dive
5.3%	eye, smile, close, little, lean, cover, remove, lip
4.6%	walk, outside, street, wave, pass, beach, sidewalk
5.7%	field, drop, slide, drive, right, kick, park, road, chest
4.7%	watch, dog, flip, stick, land, demonstrate, trick, mat
4.5%	dance, lift, try, line, snow, gun, catch, hill, bend
4.6%	fall, crowd, pour, shake, finish, raise, grass, wooden
5.9%	perform, spin, house, stage, routine, fence, bow

Table 7: A visualization of the diversity of the dataset, using a topic model (Blei et al., 2003).

gradient clipping, except for Decomposable Attention and ESIM, where we use the AllenNLP default configurations.

A.7 More info about dataset diversity

The final dataset has a vocabulary size of 21000. We also visualize the coverage of the dataset with a Topic model (see Table 7).

A.8 Comparing the distribution of verbs with MultiNLI

We also produced an extension to Figure 4 of the main paper, that involves verbs from MultiNLI, in Figure 5. We ended up not including it in the paper because we wanted to focus our comparison between SNLI and SWAG (as they are both grounded datasets). Interestingly, we find that SWAG has a less skewed cumulative distribution of verbs up to around 120, when afterwards MultiNLI has a slightly less skewed distribution. This is possibly due to the broader set of domains considered by MultiNLI, whereas we consider videos (which is also a broad domain! but still underrepresents words highly used in newswire text, for instance.)

A.9 More examples

We have more qualitative examples in Table 8.

References

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 632–642.

Zheng Cai, Lifu Tu, and Kevin Gimpel. 2017. Pay attention to the ending: Strong neural baselines for the roc story cloze task. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 616–622.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised Learning of Narrative Schemas and Their Participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 602–610, Stroudsburg, PA, USA. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1657–1668.

Gennaro Chierchia and Sally McConnell-Ginet. 2000. *Meaning and Grammar (2Nd Ed.): An Introduction to Semantics*. MIT Press, Cambridge, MA, USA.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680.

Robin Cooper, Dick Crouch, JV Eijckl, Chris Fox, JV Genabith, J Japars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, et al. 1996. A framework for computational semantics (fracas). Technical report, Technical report, The FraCaS Consortium.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

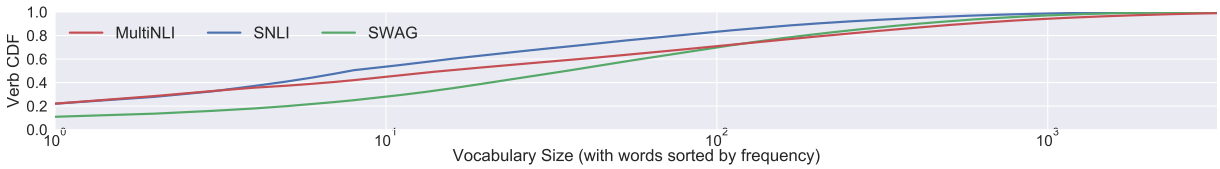


Figure 5: Bottom: CDF for verbs in SNLI, SWAG, and MultiNLI.

The lady demonstrates wrapping gifts using her feet. The lady a) shows us the different shapes of the ornaments. (99.67%) b) continues playing when the lady talks to the camera. (0.26%) c) takes the desserts from the box and continues talking to the camera . (0.07%) **d) cuts the paper with scissors. (0.01%)**	In a cafeteria, someone holds a combination tray and bowl in one hand. With the other, he a) heads into his own study. (80.67%) b) glances around and studies the photo of the blonde someone. (8.45%) **c) struggles to serve himself food with chopsticks. (6.82%)** d) opens the wall , revealing an expanse of bed within. (4.06%)
As he approaches , his kayak flips upside-down. As the view follows him, we a) see silhouetted black clouds making him zoom out of the trees, catching smoke. (42.54%) b) drift over a busy city street , like down buildings on the tarmac. (41.41%) c) find someone climbing into a tawny grave atop a road drawn among german soldiers. (13.73%) **d) notice another man seated on the rocks to the right in red with a white helmet. (2.32%)**	A man is bending over a sink. He a) takes a rag from over the sink, putting it in his mouth. (89.54%) **b) is spraying a small dog with a hose. (6.07%)** c) is carrying a shaving machine with a pressure washer. (4.29%) d) is putting a pair of shaving glass on the side of his face. (0.10%)
People are walking next to the camels leading them. A building a) is shown riding the camels. (90.72%) **b) is shown in the background. (8.39%)** c) with a rifle is leading them. (0.87%) d) is then shown for several clip. (0.01%)	A hockey game is in progress. two hockey players a) walked together in the middle of a field. (48.11%) b) walk past with a goal. (44.00%) c) sit around a rope watching the other team. (5.30%) **d) ram into each other and begin fighting. (2.58%)**
Meanwhile, someone parries another giant 's attacks. The giant a) strikes a fight and thuds into someone as he rushes in, who briefly flees. (89.96%) **b) knocks someone 's sword out of his hand. (5.25%)** c) spins him across the bars. (4.55%) d) throws stick to the bat, dragging around. (0.24%)	A lady pours ice in a glass. The lady a) pours ice into the glass. (65.14%) b) measures the contents of the glass. (33.56%) c) pours lemon mixture into a glass and pours liquids into asian juice. (0.87%) **d) adds 3 liquors and lemon juice. (0.43%)**
The stars emerge from behind the clouds. Someone a) backs away from the windows of the clip, as lightning billows over the sky. (96.59%) b) walks back across the room with nothing of his own. (1.82%) **c) stands on his boat and looks at a deep orange and red sunset. (1.47%)** d) shoots the man 's shoulder sideways, but neither do anything for a few seconds. (0.12%)	Someone stands waiting with the bridesmaids. Everyone a) seems to be ecstatic. (78.33%) **b) looks around as someone walks down the aisle, arm-in-arm with someone 's uncle. (8.97%)** c) holds someone 's eyebrow. (8.84%) d) looks at her anxiously as someone walks and sits in his seat. (3.85%)

Table 8: More (incorrect) questions answered by the best model, ESIM+Elmo, sorted by model probability. The right answers are **bolded**.

Kiana Ehsani, Hessam Bagherinezhad, Joseph Redmon, Roozbeh Mottaghi, and Ali Farhadi. 2018. Who let the dogs out? modeling dog behavior from visual data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Panna Felsen, Pulkit Agrawal, and Jitendra Malik. 2017. What will happen next? forecasting player moves in sports videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3342–3351.

Maxwell Forbes and Yejin Choi. 2017. Verb physics: Relative physical knowledge of actions and objects. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 266–276.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent

neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

JJ Gibson. 1979. The ecological approach to visual perception. *Houghton Mifflin Comp*.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking nli systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia. Association for Computational Linguistics.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2016. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. *arXiv preprint arXiv:1612.00837*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112. Association for Computational Linguistics.

Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. 2015. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970.

Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. 2017. Localizing moments in video with natural language. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1638–1649. Association for Computational Linguistics.

Ting-Hao Kenneth Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, et al. 2016. Visual storytelling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1233–1239.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *ICLR*. ICLR.

Allan Jabri, Armand Joulin, and Laurens van der Maaten. 2016. Revisiting visual question answering baselines. In *European conference on computer vision*, pages 727–739. Springer.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 427–431.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*. ICLR.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. 2017. Dense-Captioning Events in Videos. In *International Conference on Computer Vision (ICCV)*.

Alice Lai, Yonatan Bisk, and Julia Hockenmaier. 2017. Natural language inference from multiple premises. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 100–109, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. 2016. Commonsense knowledge base completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1445–1455, Berlin, Germany. Association for Computational Linguistics.

Peter LoBue and Alexander Yates. 2011. Types of common-sense knowledge needed for recognizing textual entailment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 329–334. Association for Computational Linguistics.

Li Lucy and Jon Gauthier. 2017. Are distributional representations ready for the real world? evaluating word vectors for grounded perceptual meaning. In *Proceedings of the First Workshop on Language Grounding for Robotics*, pages 76–85.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella bernardi, and Roberto Zamparelli. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland. European Language Resources Association (ELRA). ACL Anthology Identifier: L14-1314.

Nasrin Mostafazadeh, Nathanael Chambers, Xiadong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. In *NAACL*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernandez. 2016. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255.

Ramakanth Pasunuru and Mohit Bansal. 2017. Multi-task video captioning with video and entailment generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1273–1283, Vancouver, Canada. Association for Computational Linguistics.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 646–653. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2017. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. *Int. J. Comput. Vision*, 123(1):74–93.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis Only Baselines in Natural Language Inference. In *Joint Conference on Lexical and Computational Semantics (StarSem)*.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 157–163.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of Plausible Alternatives: An Evaluation of Commonsense Causal Reasoning. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*.

Anna Rohrbach, Atousa Torabi, Marcus Rohrbach, Niket Tandon, Christopher Pal, Hugo Larochelle, Aaron Courville, and Bernt Schiele. 2017. Movie Description. *International Journal of Computer Vision*, 123(1):94–120.

Rachel Rudinger, Chandler May, and Benjamin Van Durme. 2017. Social bias in elicited natural language inferences. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 74–79.

Maarten Sap, Marcella Cindy Prasettio, Ari Holtzman, Hannah Rashkin, and Yejin Choi. 2017. Connotation frames of power and agency in modern films. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2329–2334.

Roger C. Schank and Robert P. Abelson. 1975. Scripts, plans, and knowledge. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'75, pages 151–157, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Alexandra Schofield and Leo Mehr. 2016. Gender-distinguishing features in film dialogue. In *Proceedings of the Fifth Workshop on Computational Linguistics for Literature*, pages 32–39.

Roy Schwartz, Maarten Sap, Ioannis Konstas, Li Zilles, Yejin Choi, and Noah A. Smith. 2017. The effect of different writing tasks on linguistic style: A case study of the ROC story cloze task. In *Proc. of CoNLL*.

Rishi Sharma, James Allen, Omid Bakhshandeh, and Nasrin Mostafazadeh. 2018. Tackling the story ending biases in the story cloze test. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 752–757.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI Conference on Artificial Intelligence*, pages 4444–4451.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 818–827.

Vladimir Vapnik. 2000. *The Nature of Statistical Learning Theory*, 2 edition. Information Science and Statistics. Springer-Verlag, New York.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Licheng Yu, Eunbyung Park, Alexander C. Berg, and Tamara L. Berg. 2015. Visual Madlibs: Fill in the blank Image Generation and Question Answering. *ICCV*.

Rowan Zellers and Yejin Choi. 2017. Zero-shot activity recognition with verb attribute induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. 2018. Neural motifs: Scene graph parsing with global context. In *Conference on Computer Vision and Pattern Recognition*.

Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating unwanted biases with adversarial learning. In *Conference on Artificial Intelligence, Ethics and Society*.

Sheng Zhang, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2017. Ordinal Common-sense Inference. *Transactions of the Association for Computational Linguistics*, 5:379–395.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2979–2989.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *arXiv preprint arXiv:1506.06724*.

Geoffrey Zweig and Christopher JC Burges. 2011. The microsoft research sentence completion challenge. Technical report, Citeseer.

Meta-Learning for Low-Resource Neural Machine Translation

Jiatao Gu*[†], Yong Wang*[†], Yun Chen[†], Kyunghyun Cho[‡] and Victor O.K. Li[†]

[†]The University of Hong Kong
[‡]New York University, CIFAR Azrieli Global Scholar
[†]{jiataogu, wangyong, vli}@eee.hku.hk
[†]yun.chencreek@gmail.com
[‡]kyunghyun.cho@nyu.edu

Abstract

In this paper, we propose to extend the recently introduced model-agnostic meta-learning algorithm (MAML, Finn et al., 2017) for low-resource neural machine translation (NMT). We frame low-resource translation as a meta-learning problem, and we learn to adapt to low-resource languages based on multilingual high-resource language tasks. We use the universal lexical representation (Gu et al., 2018b) to overcome the input-output mismatch across different languages. We evaluate the proposed meta-learning strategy using eighteen European languages (Bg, Cs, Da, De, El, Es, Et, Fr, Hu, It, Lt, Nl, Pl, Pt, Sk, Sl, Sv and Ru) as source tasks and five diverse languages (Ro, Lv, Fi, Tr and Ko) as target tasks. We show that the proposed approach significantly outperforms the multilingual, transfer learning based approach (Zoph et al., 2016) and enables us to train a competitive NMT system with only a fraction of training examples. For instance, the proposed approach can achieve as high as 22.04 BLEU on Romanian-English WMT'16 by seeing only 16,000 translated words ($\sim$ 600 parallel sentences).

1 Introduction

Despite the massive success brought by neural machine translation (NMT, Sutskever et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017), it has been noticed that the vanilla NMT often lags behind conventional machine translation systems, such as statistical phrase-based translation systems (PBMT, Koehn et al., 2003), for low-resource language pairs (see, e.g., Koehn and Knowles, 2017). In the past few years, various approaches have been proposed to address this issue. The first attempts at tackling this problem exploited the availability of monolingual corpora (Gulcehre et al., 2015; Sennrich et al., 2015; Zhang and Zong, 2016). It was later followed by approaches based on multilingual translation, in which the goal was to exploit knowledge from high-resource language pairs by training a single NMT system on a mix of high-resource and low-resource language pairs (Firat et al., 2016a,b; Lee et al., 2016; Johnson et al., 2016; Ha et al., 2016b). Its variant, transfer learning, was also proposed by Zoph et al. (2016), in which an NMT system is pretrained on a high-resource language pair before being fine-tuned on a target low-resource language pair.

In this paper, we follow up on these latest approaches based on multilingual NMT and propose a meta-learning algorithm for low-resource neural machine translation. We start by arguing that the recently proposed model-agnostic meta-learning algorithm (MAML, Finn et al., 2017) could be applied to low-resource machine translation by viewing language pairs as separate tasks. This view enables us to use MAML to find the initialization of model parameters that facilitate fast adaptation for a new language pair with a minimal amount of training examples (§3). Furthermore, the vanilla MAML however cannot handle tasks with mismatched input and output. We overcome this limitation by incorporating the universal lexical representation (Gu et al., 2018b) and adapting it for the meta-learning scenario (§3.3).

We extensively evaluate the effectiveness and generalizing ability of the proposed meta-learning algorithm on low-resource neural machine translation. We utilize 17 languages from Europarl and Russian from WMT as the source tasks and test the meta-learned parameter initialization against five target languages (Ro, Lv, Fi, Tr and Ko), in all cases translating to English. Our experiments using only up to 160k tokens in each of the target task reveal that the proposed meta-learning approach outperforms the multilingual translation

* Equal contribution.

approach across all the target language pairs, and the gap grows as the number of training examples decreases.

2 Background

Neural Machine Translation (NMT) Given a source sentence $X = \{x_1, ..., x_{T'}\}$, a neural machine translation model factors the distribution over possible output sentences $Y = \{y_1, ..., y_T\}$ into a chain of conditional probabilities with a left-to-right causal structure:

$$p(Y|X;\theta) = \prod_{t=1}^{T+1} p(y_t|y_{0:t-1}, x_{1:T'};\theta), \quad (1)$$

where special tokens y_0 ($\langle \text{bos} \rangle$) and y_{T+1} ($\langle \text{eos} \rangle$) are used to represent the beginning and the end of a target sentence. These conditional probabilities are parameterized using a neural network. Typically, an encoder-decoder architecture (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015) with a RNN-based decoder is used. More recently, architectures without any recurrent structures (Gehring et al., 2017; Vaswani et al., 2017) have been proposed and shown to speed up training while achieving state-of-the-art performance.

Low Resource Translation NMT is known to easily over-fit and result in an inferior performance when the training data is limited (Koehn and Knowles, 2017). In general, there are two ways for handling the problem of low resource translation: (1) utilizing the resource of unlabeled monolingual data, and (2) sharing the knowledge between low- and high-resource language pairs. Many research efforts have been spent on incorporating the monolingual corpora into machine translation, such as multi-task learning (Gulcehre et al., 2015; Zhang and Zong, 2016), back-translation (Sennrich et al., 2015), dual learning (He et al., 2016) and unsupervised machine translation with monolingual corpora only for both sides (Artetxe et al., 2017b; Lample et al., 2017; Yang et al., 2018).

For the second approach, prior researches have worked on methods to exploit the knowledge of auxiliary translations, or even auxiliary tasks. For instance, Cheng et al. (2016); Chen et al. (2017); Lee et al. (2017); Chen et al. (2018) investigate the use of a pivot to build a translation path between two languages even without any directed resource. The pivot can be a third language or even an image in multimodal domains. When pivots are not easy to obtain, Firat et al. (2016a); Lee et al. (2016); Johnson et al. (2016) have shown that the structure of NMT is suitable for multilingual machine translation. Gu et al. (2018b) also showed that such a multilingual NMT system could improve the performance of low resource translation by using a universal lexical representation to share embedding information across languages.

All the previous work for multilingual NMT assume the joint training of multiple high-resource languages naturally results in a universal space (for both the input representation and the model) which, however, is not necessarily true, especially for very low resource cases.

Meta Learning In the machine learning community, meta-learning, or learning-to-learn, has recently received interests. Meta-learning tries to solve the problem of "fast adaptation on new training data." One of the most successful applications of meta-learning has been on few-shot (or one-shot) learning (Lake et al., 2015), where a neural network is trained to readily learn to classify inputs based on only one or a few training examples. There are two categories of meta-learning:

1. learning a meta-policy for updating model parameters (see, e.g., Andrychowicz et al., 2016; Ha et al., 2016a; Mishra et al., 2017)

2. learning a good parameter initialization for fast adaptation (see, e.g., Finn et al., 2017; Vinyals et al., 2016; Snell et al., 2017).

In this paper, we propose to use a meta-learning algorithm for low-resource neural machine translation based on the second category. More specifically, we extend the idea of model-agnostic meta-learning (MAML, Finn et al., 2017) in the multilingual scenario.

3 Meta Learning for Low-Resource Neural Machine Translation

The underlying idea of MAML is to use a set of source tasks $\left\{\mathcal{T}^1, \ldots, \mathcal{T}^K\right\}$ to find the initialization of parameters θ^0 from which learning a target task $\mathcal{T}^0$ would require only a small number of training examples. In the context of machine translation, this amounts to using many high-resource language pairs to find good initial parameters and training a new translation model on a low-resource language starting from the found initial parame-

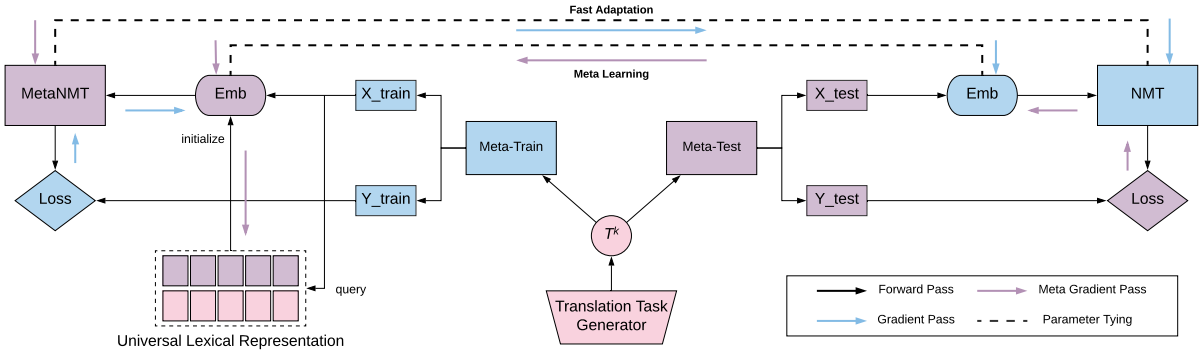


Figure 1: The graphical illustration of the training process of the proposed MetaNMT. For each episode, one task (language pair) is sampled for meta-learning. The boxes and arrows in blue are mainly involved in language-specific learning (§3.1), and those in purple in meta-learning (§3.2).

ters. This process can be understood as

$$\theta^* = \text{Learn}(\mathcal{T}^0; \text{MetaLearn}(\mathcal{T}^1, \ldots, \mathcal{T}^K)).$$

That is, we *meta-learn* the initialization from auxiliary tasks and continue to *learn* the target task. We refer the proposed meta-learning method for NMT to MetaNMT. See Fig. 1 for the overall illustration.

3.1 Learn: language-specific learning

Given any initial parameters θ^0 (which can be either random or meta-learned),

the prior distribution of the parameters of a desired NMT model can be defined as an isotropic Guassian:

$$\theta_i \sim \mathcal{N}(\theta_i^0, 1/\beta),$$

where $1/\beta$ is a variance. With this prior distribution, we formulate the language-specific learning process $\text{Learn}(D_\mathcal{T}; \theta^0)$ as maximizing the log-posterior of the model parameters given data $D_\mathcal{T}$:

$$\begin{aligned}\text{Learn}(D_\mathcal{T}; \theta^0) &= \arg\max_\theta \mathcal{L}^{D_\mathcal{T}}(\theta) \\ &= \arg\max_\theta \sum_{(X,Y)\in D_\mathcal{T}} \log p(Y|X,\theta) - \beta\|\theta - \theta^0\|^2,\end{aligned}$$

where we assume $p(X|\theta)$ to be uniform. The first term above corresponds to the maximum likelihood criterion often used for training a usual NMT system. The second term discourages the newly learned model from deviating too much from the initial parameters, alleviating the issue of overfitting when there is not enough training data. In practice, we solve the problem above by maximizing the first term with gradient-based optimization and early-stopping after only a few update steps.

Thus, in the low-resource scenario, finding a good initialization θ^0 strongly correlates the final performance of the resulting model.

3.2 MetaLearn

We find the initialization θ^0 by repeatedly simulating low-resource translation scenarios using auxiliary, high-resource language pairs. Following Finn et al. (2017), we achieve this goal by defining the meta-objective function as

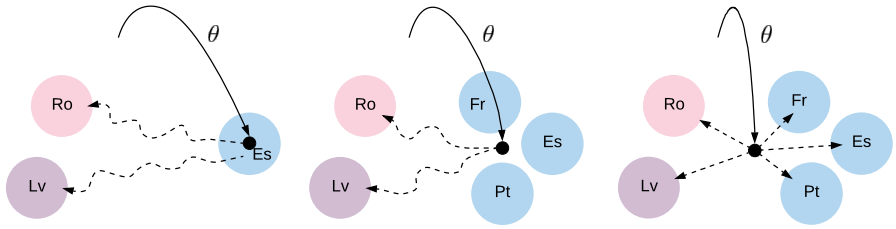
$$\begin{aligned}\mathcal{L}(\theta) =& \mathbb{E}_k \mathbb{E}_{D_{\mathcal{T}^k}, D'_{\mathcal{T}^k}} \\ & \left[\sum_{(X,Y)\in D'_{\mathcal{T}^k}} \log p(Y|X; \text{Learn}(D_{\mathcal{T}^k}; \theta)) \right],\end{aligned} \tag{2}$$

where $k \sim \mathcal{U}(\{1, \ldots, K\})$ refers to one meta-learning episode, and $D_\mathcal{T}$, $D'_\mathcal{T}$ follow the uniform distribution over $\mathcal{T}$'s data.

We maximize the meta-objective function using stochastic approximation (Robbins and Monro, 1951) with gradient descent. For each episode, we uniformly sample one source task at random, $\mathcal{T}^k$. We then sample two subsets of training examples independently from the chosen task, $D_{\mathcal{T}^k}$ and $D'_{\mathcal{T}^k}$. We use the former to *simulate* language-specific learning and the latter to *evaluate* its outcome. Assuming a single gradient step is taken only the with learning rate η, the simulation is:

$$\theta'_k = \text{Learn}(D_{\mathcal{T}^k}; \theta) = \theta - \eta \nabla_\theta \mathcal{L}^{D_{\mathcal{T}^k}}(\theta).$$

Once the simulation of learning is done, we evaluate the updated parameters θ'_k on $D'_{\mathcal{T}^k}$, The gradient computed from this evaluation, which we refer to as *meta-gradient*, is used to update the



(a) Transfer Learning (b) Multilingual Transfer Learning (c) Meta Learning

Figure 2: An intuitive illustration in which we use solid lines to represent the learning of initialization, and dashed lines to show the path of fine-tuning.

meta model θ. It is possible to aggregate multiple episodes of source tasks before updating θ:

$$\theta \leftarrow \theta - \eta' \sum_k \nabla_\theta \mathcal{L}^{D'_{\mathcal{T}^k}}(\theta'_k),$$

where η' is the meta learning rate.

Unlike a usual learning scenario, the resulting model θ^0 from this meta-learning procedure is not necessarily a good model on its own. It is however a good starting point for training a good model using only a few steps of learning. In the context of machine translation, this procedure can be understood as finding the initialization of a neural machine translation system that could quickly adapt to a new language pair by simulating such a fast adaptation scenario using many high-resource language pairs.

Meta-Gradient We use the following approximation property

$$H(x)v \approx \frac{\nabla(x + \nu v) - \nabla(x)}{\nu}$$

to approximate the meta-gradient:[1]

$$\begin{aligned}
&\nabla_\theta \mathcal{L}^{D'}(\theta') = \nabla_{\theta'} \mathcal{L}^{D'}(\theta') \nabla_\theta (\theta - \eta \nabla_\theta \mathcal{L}^D(\theta)) \\
&= \nabla_{\theta'} \mathcal{L}^{D'}(\theta') - \eta \nabla_{\theta'} \mathcal{L}^{D'}(\theta') H_\theta(\mathcal{L}^D(\theta)) \\
&\approx \nabla_{\theta'} \mathcal{L}^{D'}(\theta') - \frac{\eta}{\nu} \left[\nabla_\theta \mathcal{L}^D(\theta) \Big|_{\hat{\theta}} - \nabla_\theta \mathcal{L}^D(\theta) \Big|_{\theta} \right],
\end{aligned}$$

where ν is a small constant and

$$\hat{\theta} = \theta + \nu \nabla_{\theta'} \mathcal{L}^{D'}(\theta').$$

In practice, we find that it is also possible to ignore the second-order term, ending up with the following simplified update rule:

$$\nabla_\theta \mathcal{L}^{D'}(\theta') \approx \nabla_{\theta'} \mathcal{L}^{D'}(\theta'). \quad (3)$$

[1] We omit the subscript k for simplicity.

Related Work: Multilingual Transfer Learning The proposed MetaNMT differs from the existing framework of multilingual translation (Lee et al., 2016; Johnson et al., 2016; Gu et al., 2018b) or transfer learning (Zoph et al., 2016). The latter can be thought of as solving the following problem:

$$\max_\theta \mathcal{L}^{\text{multi}}(\theta) = \mathbb{E}_k \left[\sum_{(X,Y) \in D_k} \log p(Y|X;\theta) \right],$$

where D_k is the training set of the k-th task, or language pair. The target low-resource language pair could either be a part of joint training or be trained separately starting from the solution θ^0 found from solving the above problem.

The major difference between the proposed MetaNMT and these multilingual transfer approaches is that the latter do not consider how learning happens with the target, low-resource language pair. The former explicitly incorporates the learning process within the framework by simulating it repeatedly in Eq. (2). As we will see later in the experiments, this results in a substantial gap in the final performance on the low-resource task.

Illustration In Fig. 2, we contrast transfer learning, multilingual learning and meta-learning using three source language pairs (Fr-En, Es-En and Pt-En) and two target pairs (Ro-En and Lv-En). Transfer learning trains an NMT system specifically for a source language pair (Es-En) and finetunes the system for each target language pair (Ro-En, Lv-En). Multilingual learning often trains a single NMT system that can handle many different language pairs (Fr-En, Pt-En, Es-En), which may or may not include the target pairs (Ro-En, Lv-En). If not, it finetunes the system for each target pair, similarly to transfer learning. Both of these however aim at directly solving the source tasks. On the other hand, meta-learning trains the NMT system to be *useful for fine-tuning* on various tasks including the source and target tasks. This is done by repeatedly simulating the learning process on

low-resource languages using many high-resource language pairs (Fr-En, Pt-En, Es-En).

3.3 Unified Lexical Representation

I/O mismatch across language pairs One major challenge that limits applying meta-learning for low resource machine translation is that the approach outlined above assumes the input and output spaces are shared across all the source and target tasks. This, however, does not apply to machine translation in general due to the vocabulary mismatch across different languages. In multilingual translation, this issue has been tackled by using a vocabulary of sub-words (Sennrich et al., 2015) or characters (Lee et al., 2016) shared across multiple languages. This surface-level sharing is however limited, as it cannot be applied to languages exhibiting distinct orthography (e.g., Indo-Euroepan languages vs. Korean.)

Universal Lexical Representation (ULR) We tackle this issue by dynamically building a vocabulary specific to each language using a key-value memory network (Miller et al., 2016; Gulcehre et al., 2018), as was done successfully for low-resource machine translation recently by Gu et al. (2018b). We start with multilingual word embedding matrices $\epsilon^k_{\text{query}} \in \mathbb{R}^{|V_k| \times d}$ pretrained on large monolingual corpora, where V_k is the vocabulary of the k-th language. These embedding vectors can be obtained with small dictionaries of seed word pairs (Artetxe et al., 2017a; Smith et al., 2017) or in a fully unsupervised manner (Zhang et al., 2017; Conneau et al., 2018). We take one of these languages k' to build universal lexical representation consisting of a universal embedding matrix $\epsilon_u \in \mathbb{R}^{M \times d}$ and a corresponding key matrix $\epsilon_{\text{key}} \in \mathbb{R}^{M \times d}$, where $M < |V'_k|$. Both $\epsilon^k_{\text{query}}$ and ϵ_{key} are fixed during meta-learning. We then compute the language-specific embedding of token x from the language k as the convex sum of the universal embedding vectors by

$$\epsilon^0[x] = \sum_{i=1}^{M} \alpha_i \epsilon_u[i],$$

where $\alpha_i \propto \exp\left\{-\frac{1}{\tau}\epsilon_{\text{key}}[i]^\top A \epsilon^k_{\text{query}}[x]\right\}$ and τ is set to 0.05. This approach allows us to handle languages with different vocabularies using a fixed number of shared parameters (ϵ_u, ϵ_{key} and A.)

Learning of ULR It is not desirable to update the universal embedding matrix ϵ_u when fine-tuning on a small corpus which contains a limited set of unique tokens in the target language, as it could adversely influence the other tokens' embedding vectors. We thus estimate the change to each embedding vector induced by language-specific learning by a separate parameter $\Delta\epsilon^k[x]$:

$$\epsilon^k[x] = \epsilon^0[x] + \Delta\epsilon^k[x].$$

During language-specific learning, the ULR $\epsilon^0[x]$ is held constant, while only $\Delta\epsilon^k[x]$ is updated, starting from an all-zero vector. On the other hand, we hold $\Delta\epsilon^k[x]$'s constant while updating ϵ_u and A during the meta-learning stage.

	# of sents.	# of En tokens	Dev	Test
Ro-En	0.61 M	16.66 M	–	31.76
Lv-En	4.46 M	67.24 M	20.24	15.15
Fi-En	2.63 M	64.50 M	17.38	20.20
Tr-En	0.21 M	5.58 M	15.45	13.74
Ko-En	0.09 M	2.33 M	6.88	5.97

Table 1: Statistics of full datasets of the target language pairs. BLEU scores on the dev and test sets are reported from a supervised Transformer model with the same architecture.

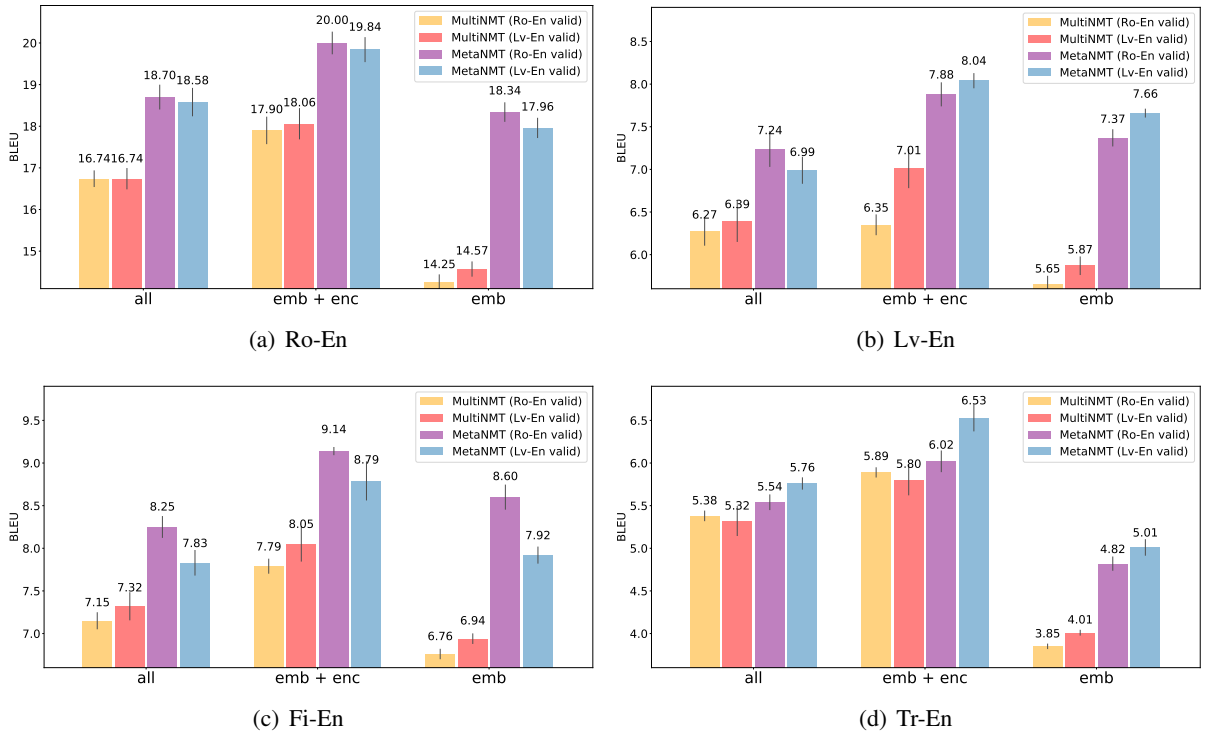
4 Experimental Settings

4.1 Dataset

Target Tasks We show the effectiveness of the proposed meta-learning method for low resource NMT with extremely limited training examples on five diverse target languages: Romanian (Ro) from WMT'16,[2] Latvian (Lv), Finnish (Fi), Turkish (Tr) from WMT'17,[3] and Korean (Ko) from Korean Parallel Dataset.[4] We use the officially provided train, dev and test splits for all these languages. The statistics of these languages are presented in Table 1. We simulate the low-resource translation scenarios by randomly sub-sampling the training set with different sizes.

Source Tasks We use the following languages from Europarl[5]: Bulgarian (Bg), Czech (Cs), Danish (Da), German (De), Greek (El), Spanish (Es), Estonian (Et), French (Fr), Hungarian (Hu), Italian (It), Lithuanian (Lt), Dutch (Nl), Polish (Pl), Portuguese (Pt), Slovak (Sk), Slovene (Sl) and

[2] http://www.statmt.org/wmt16/translation-task.html
[3] http://www.statmt.org/wmt17/translation-task.html
[4] https://sites.google.com/site/koreanparalleldata/
[5] http://www.statmt.org/europarl/



(a) Ro-En (b) Lv-En

(c) Fi-En (d) Tr-En

Figure 3: BLEU scores reported on test sets for {Ro, Lv, Fi, Tr} to En, where each model is first learned from 6 source tasks (Es, Fr, It, Pt, De, Ru) and then fine-tuned on randomly sampled training sets with around 16,000 English tokens per run. The error bars show the standard deviation calculated from 5 runs.

Swedish (Sv), in addition to Russian (Ru)[6] to learn the intilization for fine-tuning. In our experiments, different combinations of source tasks are explored to see the effects from the source tasks.

Validation We pick either Ro-En or Lv-En as a validation set for meta-learning and test the generalization capability on the remaining target tasks. This allows us to study the strict form of meta-learning, in which target tasks are unknown during both training and model selection.

Preprocessing and ULR Initialization As described in §3.3, we initialize the query embedding vectors $\epsilon^k_{\text{query}}$ of all the languages. For each language, we use the monolingual corpora built from Wikipedia[7] and the parallel corpus. The concatenated corpus is first tokenized and segmented using byte-pair encoding (BPE, Sennrich et al., 2016), resulting in $40,000$ subwords for each language. We then estimate word vectors using fastText (Bojanowski et al., 2016) and align them across all the languages in an unsupervised way using MUSE (Conneau et al., 2018) to get multilingual word vectors. We use the multilingual word vectors of the 20,000 most frequent words in English to form the universal embedding matrix ϵ_u.

4.2 Model and Learning

Model We utilize the recently proposed Transformer (Vaswani et al., 2017) as an underlying NMT system. We implement Transformer in this paper based on (Gu et al., 2018a)[8] and modify it to use the universal lexical representation from §3.3. We use the default set of hyperparameters ($d_{\text{model}} = d_{\text{hidden}} = 512$, $n_{\text{layer}} = 6$, $n_{\text{head}} = 8$, $n_{\text{batch}} = 4000$, $t_{\text{warmup}} = 16000$) for all the language pairs and across all the experimental settings. We refer the readers to (Vaswani et al., 2017; Gu et al., 2018a) for the details of the model. However, since the proposed meta-learning method is model-agnostic, it can be easily extended to any other NMT architectures, e.g. RNN-based sequence-to-sequence models with attention (Bahdanau et al., 2015).

[6] A subsample of approximately 2M pairs from WMT'17.

[7] We use the most recent Wikipedia dump (2018.5) from https://dumps.wikimedia.org/backup-index.html.

[8] https://github.com/salesforce/nonauto-nmt

Meta-Train	Ro-En		Lv-En		Fi-En		Tr-En		Ko-En	
	zero	finetune	zero	finetune	zero	finetune	zero	finetune	zero	finetune
–		$00.00 \pm .00$		$0.00 \pm .00$		$0.00 \pm .00$		$0.00 \pm .00$		$0.00 \pm .00$
Es	9.20	$15.71 \pm .22$	2.23	$4.65 \pm .12$	2.73	$5.55 \pm .08$	1.56	$4.14 \pm .03$	0.63	$1.40 \pm .09$
Es Fr	12.35	$17.46 \pm .41$	2.86	$5.05 \pm .04$	3.71	$6.08 \pm .01$	2.17	$4.56 \pm .20$	0.61	$1.70 \pm .14$
Es Fr It Pt	13.88	$18.54 \pm .19$	3.88	$5.63 \pm .11$	4.93	$6.80 \pm .04$	2.49	$4.82 \pm .10$	0.82	$1.90 \pm .07$
De Ru	10.60	$16.05 \pm .31$	5.15	$7.19 \pm .17$	6.62	$7.98 \pm .22$	3.20	$6.02 \pm .11$	1.19	$2.16 \pm .09$
Es Fr It Pt De Ru	15.93	$20.00 \pm .27$	6.33	$7.88 \pm .14$	7.89	$9.14 \pm .05$	3.72	$6.02 \pm .13$	1.28	$2.44 \pm .11$
All	18.12	$\mathbf{22.04 \pm .23}$	9.58	$\mathbf{10.44 \pm .17}$	11.39	$\mathbf{12.63 \pm .22}$	5.34	$\mathbf{8.97 \pm .08}$	1.96	$\mathbf{3.97 \pm .10}$
Full Supervised	31.76		15.15		20.20		13.74		5.97	

Table 2: BLEU Scores w.r.t. the source task set for all five target tasks.

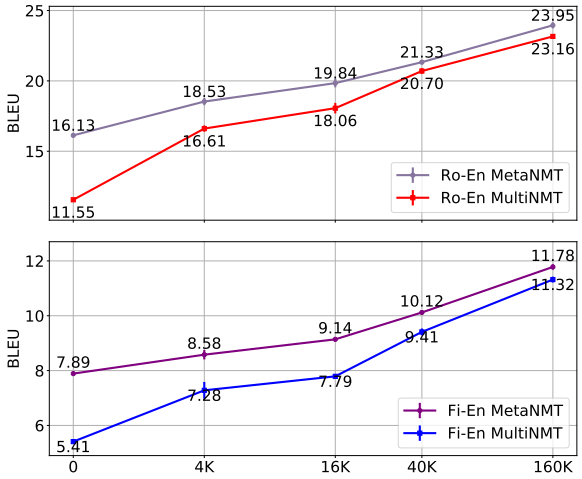


Figure 4: BLEU Scores w.r.t. the size of the target task's training set.

Learning We meta-learn using various sets of source languages to investigate the effect of source task choice. For each episode, by default, we use a single gradient step of language-specific learning with Adam (Kingma and Ba, 2014) per computing the meta-gradient, which is computed by the first-order approximation in Eq. (3).

For each target task, we sample training examples to form a low-resource task. We build tasks of 4k, 16k, 40k and 160k English tokens for each language. We randomly sample the training set five times for each experiment and report the average score and its standard deviation. Each fine-tuning is done on a training set, early-stopped on a validation set and evaluated on a test set. In default without notation, datasets of 16k tokens are used.

Fine-tuning Strategies The transformer consists of three modules; embedding, encoder and decoder. We update all three modules during meta-learning, but during fine-tuning, we can selectively tune only a subset of these modules. Following (Zoph et al., 2016), we consider three fine-tuning strategies; (1) fine-tuning all the modules (all), (2) fine-tuning the embedding and encoder, but freezing the parameters of the decoder (emb+enc) and (3) fine-tuning the embedding only (emb).

5 Results

vs. Multilingual Transfer Learning We meta-learn the initial models on all the source tasks using either Ro-En or Lv-En as a validation task. We also train the initial models to be multilingual translation systems. We fine-tune them using the four target tasks (Ro-En, Lv-En, Fi-En and Tr-En; 16k tokens each) and compare the proposed meta-learning strategy and the multilingual, transfer learning strategy. As presented in Fig. 3, the proposed learning approach significantly outperforms the multilingual, transfer learning strategy across all the target tasks regardless of which target task was used for early stopping. We also notice that the emb+enc strategy is most effective for both meta-learning and transfer learning approaches. With the proposed meta-learning and emb+enc fine-tuning, the final NMT systems trained using only a fraction of all available training examples achieve 2/3 (Ro-En) and 1/2 (Lv-En, Fi-En and Tr-En) of the BLEU score achieved by the models trained with full training sets.

vs. Statistical Machine Translation We also test the same Ro-En datasets with $16,000$ target tokens using the default setting of Phrase-based MT (Moses) with the dev set for adjusting the parameters and the test set for calculating the final performance. We obtain $4.79(\pm 0.234)$ BLEU point, which is higher than the standard NMT performance (0 BLEU). It is however still lower than both the multi-NMT and meta-NMT.

Impact of Validation Tasks Similarly to training any other neural network, meta-learning still requires early-stopping to avoid overfitting to a

specific set of source tasks. In doing so, we observe that the choice of a validation task has non-negligible impact on the final performance. For instance, as shown in Fig. 3, Fi-En benefits more when Ro-En is used for validation, while the opposite happens with Tr-En. The relationship between the task similarity and the impact of a validation task must be investigated further in the future.

Training Set Size We vary the size of the target task's training set and compare the proposed meta-learning strategy and multilingual, transfer learning strategy. We use the emb+enc fine-tuning on Ro-En and Fi-En. Fig. 4 demonstrates that the meta-learning approach is more robust to the drop in the size of the target task's training set. The gap between the meta-learning and transfer learning grows as the size shrinks, confirming the effectiveness of the proposed approach on extremely low-resource language pairs.

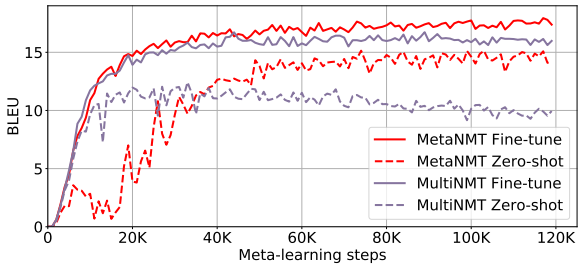


Figure 5: The learning curves of BLEU scores on the validation task (Ro-En).

Impact of Source Tasks In Table 2, we present the results on all five target tasks obtained while varying the source task set. We first see that it is always beneficial to use more source tasks. Although the impact of adding more source tasks varies from one language to another, there is up to 2× improvement going from one source task to 18 source tasks (Lv-En, Fi-En, Tr-En and Ko-En). The same trend can be observed even without any fine-tuning (i.e., unsupervised translation, (Lample et al., 2017; Artetxe et al., 2017b)). In addition, the choice of source languages has different implications for different target languages. For instance, Ro-En benefits more from {Es, Fr, It, Pt} than from {De, Ru}, while the opposite effect is observed with all the other target tasks.

Training Curves The benefit of meta-learning over multilingual translation is clearly demonstrated when we look at the training curves in Fig. 5. With the multilingual, transfer learning approach, we observe that training rapidly saturates and eventually degrades, as the model overfits to the source tasks. MetaNMT on the other hand continues to improve and never degrades, as the meta-objective ensures that the model is adequate for fine-tuning on target tasks rather than for solving the source tasks.

Sample Translations We present some sample translations from the tested models in Table 3. Inspecting these examples provides the insight into the proposed meta-learning algorithm. For instance, we observe that the meta-learned model without any fine-tuning produces a word-by-word translation in the first example (Tr-En), which is due to the successful use of the universal lexcial representation and the meta-learned initialization. The system however cannot reorder tokens from Turkish to English, as it has not seen any training example of Tr-En. After seeing around 600 sentence pairs (16K English tokens), the model rapidly learns to correctly reorder tokens to form a better translation. A similar phenomenon is observed in the Ko-En example. These cases could be found across different language pairs.

6 Conclusion

In this paper, we proposed a meta-learning algorithm for low-resource neural machine translation that exploits the availability of high-resource languages pairs. We based the proposed algorithm on the recently proposed model-agnostic meta-learning and adapted it to work with multiple languages that do not share a common vocabulary using the technique of universal lexcal representation, resulting in MetaNMT. Our extensive evaluation, using 18 high-resource source tasks and 5 low-resource target tasks, has shown that the proposed MetaNMT significantly outperforms the existing approach of multilingual, transfer learning in low-resource neural machine translation across all the language pairs considered.

The proposed approach opens new opportunities for neural machine translation. First, it is a principled framework for incorporating various extra sources of data, such as source- and target-side monolingual corpora. Second, it is a generic framework that can easily accommodate existing and future neural machine translation systems.

Source (Tr)	google mülteciler için 11 milyon dolar toplamak üzere bağış eşleştirme kampanyasını başlattı .
Target	google launches donation-matching campaign to raise $ 11 million for refugees .
Meta-0	google refugee fund for usd 11 million has launched a campaign for donation .
Meta-16k	google has launched a campaign to collect $ 11 million for refugees .
Source (Ko)	이번에 체포되어 기소된 사람들 중에는 퇴역한 군 고위관리 , 언론인 , 정치인 , 경제인 등이 포함됐다
Target	among the suspects are retired military officials , journalists , politicians , businessmen and others .
Meta-0	last year , convicted people , among other people , of a high-ranking army of journalists in economic and economic policies , were included .
Meta-16k	the arrested persons were included in the charge , including the military officials , journalists , politicians and economists .

Table 3: Sample translations for Tr-En and Ko-En highlight the impact of fine-tuning which results in syntactically better formed translations. We highlight tokens of interest in terms of reordering.

Acknowledgement

This research was supported in part by the Facebook Low Resource Neural Machine Translation Award. This work was also partly supported by Samsung Advanced Institute of Technology (Next Generation Deep Learning: from pattern recognition to AI) and Samsung Electronics (Improving Deep Learning using Latent Structure). KC thanks support by eBay, TenCent, NVIDIA and CIFAR.

References

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. 2016. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017a. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 451–462.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017b. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Yun Chen, Yang Liu, Yong Cheng, and Victor OK Li. 2017. A teacher-student framework for zero-resource neural machine translation. *arXiv preprint arXiv:1705.00753*.

Yun Chen, Yang Liu, and Victor OK Li. 2018. Zero-resource neural machine translation with multi-agent communication game. *arXiv preprint arXiv:1802.03116*.

Yong Cheng, Yang Liu, Qian Yang, Maosong Sun, and Wei Xu. 2016. Neural machine translation with pivot languages. *arXiv preprint arXiv:1611.04928*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–Decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. *International Conference on Learning Representations*.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016a. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *NAACL*.

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T Yarman Vural, and Kyunghyun Cho. 2016b. Zero-resource translation with multi-lingual neural machine translation. In *EMNLP*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2018a. Non-autoregressive neural machine translation. *ICLR*.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor OK Li. 2018b. Universal neural machine translation for extremely low resource languages. *arXiv preprint arXiv:1802.05368*.

Caglar Gulcehre, Sarath Chandar, Kyunghyun Cho, and Yoshua Bengio. 2018. Dynamic neural turing machine with continuous and discrete addressing schemes. *Neural computation*, 30(4):857–884.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

David Ha, Andrew Dai, and Quoc V Le. 2016a. Hypernetworks. *arXiv preprint arXiv:1609.09106*.

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2016b. Toward multilingual neural machine translation with universal encoder and decoder. *arXiv preprint arXiv:1611.04798*.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2016. Fully character-level neural machine translation without explicit segmentation. *arXiv preprint arXiv:1610.03017*.

Jason Lee, Kyunghyun Cho, Jason Weston, and Douwe Kiela. 2017. Emergent translation in multi-agent communication. *arXiv preprint arXiv:1710.06922*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2017. Meta-learning with temporal convolutions. *arXiv preprint arXiv:1707.03141*.

Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for wmt 16. *arXiv preprint arXiv:1606.02891*.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4080–4090.

Ilya Sutskever, Oriol Vinyals, and Quôc Lê. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Unsupervised neural machine translation with weight sharing. *arXiv preprint arXiv:1804.09057*.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Earth mover's distance minimization for unsupervised bilingual lexicon induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1934–1945. Association for Computational Linguistics.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. *arXiv preprint arXiv:1604.02201*.

Dissecting Contextual Word Embeddings: Architecture and Representation

Matthew E. Peters[*1], **Mark Neumann**[*1], **Luke Zettlemoyer**[2], **Wen-tau Yih**[1]
[1]Allen Institute for Artificial Intelligence, Seattle, WA, USA
[2]Paul G. Allen Computer Science & Engineering, University of Washington
{matthewp,markn,scottyih}@allenai.org lsz@cs.washington.edu

Abstract

Contextual word representations derived from pre-trained bidirectional language models (biLMs) have recently been shown to provide significant improvements to the state of the art for a wide range of NLP tasks. However, many questions remain as to how and why these models are so effective. In this paper, we present a detailed empirical study of how the choice of neural architecture (e.g. LSTM, CNN, or self attention) influences both end task accuracy and qualitative properties of the representations that are learned. We show there is a tradeoff between speed and accuracy, but all architectures learn high quality contextual representations that outperform word embeddings for four challenging NLP tasks. Additionally, all architectures learn representations that vary with network depth, from exclusively morphological based at the word embedding layer through local syntax based in the lower contextual layers to longer range semantics such coreference at the upper layers. Together, these results suggest that unsupervised biLMs, independent of architecture, are learning much more about the structure of language than previously appreciated.

1 Introduction

Language model pre-training has been shown to substantially improve performance for many NLP tasks including question answering, coreference resolution, and semantic role labeling (Peters et al., 2018), text classification (Dai and Le, 2015; Howard and Ruder, 2018), sequence tagging (Peters et al., 2017), sequence–to–sequence learning (Ramachandran et al., 2017), and constituency parsing (Kitaev and Klein, 2018; Joshi et al., 2018). Despite large gains (typical relative error reductions range from 10–25%), we do not yet fully understand why or how pre-training works in practice. In this paper, we take a step towards such understanding by empirically studying how the choice of neural architecture (e.g. LSTM, CNN, or self attention) influences both direct end-task accuracies and how contextualized word representations encode notions of syntax and semantics.

Previous work on learning contextual representations has used LSTM-based biLMs, but there is no prior reason to believe this is the best possible architecture. More computationally efficient networks have been introduced for sequence modeling including including gated CNNs for language modeling (Dauphin et al., 2017) and feed forward self-attention based approaches for machine translation (Transformer; Vaswani et al., 2017). As RNNs are forced to compress the entire history into a hidden state vector before making predictions while CNNs with a large receptive field and the Transformer may directly reference previous tokens, each architecture will represent information in a different manner.

Given such differences, we study whether more efficient architectures can also be used to learn high quality contextual vectors. We show empirically that all three approaches provide large improvements over traditional word vectors when used in state-of-the-art models across four benchmark NLP tasks. We do see the expected tradeoff between speed and accuracy between LSTMs and the other alternatives, but the effect is relatively modest and all three networks work well in practice.

Given this result, it is important to better understand what the different networks learn. In a detailed quantitative evaluation, we probe the learned representations and show that, in every case, they represent a rich hierarchy of contextual information throughout the layers of the network in an analogous manner to how deep CNNs

[*]These authors contributed equally to this work.

trained for image classification learn a hierarchy of image features (Zeiler and Fergus, 2014). For example, we show that in contrast to traditional word vectors which encode some semantic information, the word embedding layer of deep biLMs focuses exclusively on word morphology. Moving upward in the network, the lowest contextual layers of biLMs focus on local syntax, while the upper layers can be used to induce more semantic content such as within-sentence pronominal coreferent clusters. We also show that the biLM activations can be used to form phrase representations useful for syntactic tasks. Together, these results suggest that large scale biLMs, independent of architecture, are learning much more about the structure of language than previous appreciated.

2 Contextual word representations from biLMs

To learn contextual word representations, we follow previous work by first training a biLM on a large text corpus (Sec. 2.1). Then, the internal layer activations from the biLM are transferred to downstream tasks (Sec. 2.3).

2.1 Bidirectional language models

Given a sequence of N tokens, $(t_1, t_2, ..., t_N)$, a biLM combines a forward and backward language model to jointly maximize the log likelihood of both directions:

$$\sum_{k=1}^{N} \Big(\log p(t_k \mid t_1, \ldots, t_{k-1}; \overrightarrow{\Theta}) + \log p(t_k \mid t_{k+1}, \ldots, t_N; \overleftarrow{\Theta}) \Big),$$

where $\overrightarrow{\Theta}$ and $\overleftarrow{\Theta}$ are the parameters of the forward and backward LMs respectively.

To compute the probability of the next token, state-of-the-art neural LMs first produce a context-insensitive token representation or word embedding, $\mathbf{x}_k$, (with either an embedding lookup or in our case a character aware encoder, see below). Then, they compute L layers of context-dependent representations $\overrightarrow{\mathbf{h}}_{k,i}$ where $i \in [1, L]$ using a RNN, CNN or feed forward network (see Sec. 3). The top layer output $\overrightarrow{\mathbf{h}}_{k,L}$ is used to predict the next token using a Softmax layer. The backward LM operates in an analogous manner to the forward LM. Finally, we can concatenate the forward and backward states to form L layers of contextual representations, or context vectors, at each token position: $\mathbf{h}_{k,i} = [\overrightarrow{\mathbf{h}}_{k,i}; \overleftarrow{\mathbf{h}}_{k,i}]$. When training, we tie the weights of the word embedding layers and Softmax in each direction but maintain separate weights for the contextual layers.

2.2 Character based language models

Fully character aware models (Kim et al., 2015) are considerably more parameter efficient then word based models but more computationally expensive then word embedding based methods when training. During inference, these differences can be largely eliminated by pre-computing embeddings for a large vocabulary and only falling back to the full character based method for rare words. Overall, for a large English language news benchmark, character aware models have slightly better perplexities then word based ones, although the differences tend to be small (Józefowicz et al., 2016).

Similar to Kim et al. (2015), our character-to-word encoder is a five-layer sub-module that first embeds single characters with an embedding layer then passes them through 2048 character n-gram CNN filters with max pooling, two highway layers (Srivastava et al., 2015), and a linear projection down to the model dimension.

2.3 Deep contextual word representations

After pre-training on a large data set, the internal representations from the biLM can be transferred to a downstream model of interest as contextual word representations. To effectively use all of the biLM layers, Peters et al. (2018) introduced ELMo word representations, whereby all of the layers are combined with a weighted average pooling operation, $\mathbf{ELMo}_k = \gamma \sum_{j=0}^{L} s_j \mathbf{h}_{k,j}$. The parameters $\mathbf{s}$ are optimized as part of the task model so that it may preferentially mix different types of contextual information represented in different layers of the biLM. In Sec. 4 we evaluate the relative effectiveness of ELMo representations from three different biLM architectures vs. pre-trained word vectors in four different state-of-the-art models.

3 Architectures for deep biLMs

The primary design choice when training deep biLMs for learning context vectors is the choice of the architecture for the contextual layers. However, it is unknown if the architecture choice is important for the quality of learned representations. To study this question, we consider two alterna-

Architecture	# layers	Perplexity	# params. (M)	Inference (ms) 1 sentence	Inference (ms) 64 sentences
LSTM	2	39.7	76 / 94	44 / 46	66 / 85
LSTM	4	37.5	151 / 153	85 / 86	102 / 118
Transformer	6	40.7	38 / 56	12 / 13	22 / 44
Gated CNN	16	44.5	67 / 85	9 / 11	29 / 55

Table 1: Characteristics of the different biLMs in this study. For each model, the table shows the number of layers used for the contextual representations, the averaged forward and backward perplexities on the 1 Billion Word Benchmark, the number of parameters (in millions, excluding softmax) and the inference speed (in milliseconds with a Titan X GPU, for sentences with 20 tokens, excluding softmax). For the number of parameters and inference speeds we list both the values for just the contextual layers and all layers needed to compute context vectors.

tives to LSTMs as described below. See the appendix for the hyperparameter details.

3.1 LSTM

Among the RNN variants, LSTMs have been shown to provide state-of-the-art performance for several benchmark language modeling tasks (Józefowicz et al., 2016; Merity et al., 2018; Melis et al., 2018). In particular, the LSTM with projection introduced by Sak et al. (2014) allows the model to use a large hidden state while reducing the total number of parameters. This is the architecture adopted by Peters et al. (2018) for computing ELMo representations. In addition to the pre-trained 2-layer biLM from that work,[1] we also trained a deeper 4-layer model to examine the impact of depth using the publicly available training code.[2] To reduce the training time for this large 4-layer model, we reduced the number of parameters in the character encoder by first projecting the character CNN filters down to the model dimension before the two highway layers.

3.2 Transformer

The Transformer, introduced by Vaswani et al. (2017), is a feed forward self-attention based architecture. In addition to machine translation, it has also provided strong results for Penn Treebank constituency parsing (Kitaev and Klein, 2018) and semantic role labeling (Tan et al., 2018). Each identical layer in the encoder first computes a multi-headed attention between a given token and all other tokens in the history, then runs a position wise feed forward network.

To adapt the Transformer for bidirectional language modeling, we modified a PyTorch based re-implementation (Klein et al., 2017)[3] to mask out future tokens for the forward language model and previous tokens for the backward language model, in a similar manner to the decoder masking in the original implementation. We adopted hyper-parameters from the "base" configuration in Vaswani et al. (2017), providing six layers of 512 dimensional representations for each direction.

Concurrent with our work, Radford et al. (2018) trained a large forward Transformer LM and fine tuned it for a variety of NLP tasks.

3.3 Gated CNN

Convolutional architectures have also been shown to provide competitive results for sequence modeling including sequence-to-sequence machine translation (Gehring et al., 2017). Dauphin et al. (2017) showed that architectures using Gated Linear Units (GLU) that compute hidden representations as the element wise product of a convolution and sigmoid gate provide perplexities comparable to large LSTMs on large scale language modeling tasks.

To adapt the Gated CNN for bidirectional language modeling, we closely followed the publicly available ConvSeq2Seq implementation,[4] modified to support causal convolutions (van den Oord et al., 2016) for both the forward and backward directions. In order to model a wide receptive field at the top layer, we used a 16-layer deep model, where each layer is a $[4, 512]$ residual block.

3.4 Pre-trained biLMs

Table 1 compares the biLMs used in the remainder of this study. All models were trained on the 1

[1] http://allennlp.org/elmo
[2] https://github.com/allenai/bilm-tf
[3] http://nlp.seas.harvard.edu/2018/04/03/attention.html
[4] https://github.com/pytorch/fairseq

Architecture	MultiNLI	SRL	Constituency Parsing	NER
GloVe	77.0 / 76.0	81.4	91.8	89.9 $\pm$ 0.35
LSTM 2-layer	79.6 / 79.3	84.6	**93.9**	**91.7 $\pm$ 0.26**
LSTM 4-layer	**80.1 / 79.7**	**84.7**	**93.9**	91.5 $\pm$ 0.12
Transformer	79.4 / 78.7	84.1	93.7	91.1 $\pm$ 0.26
Gated CNN	78.3 / 77.9	84.1	92.9	91.2 $\pm$ 0.14

Table 2: Test set performance comparison using different pre-trained biLM architectures. The performance metric is accuracy for MultiNLI and F_1 score for the other tasks. For MultiNLI, the table shows accuracy on both the matched and mismatched portions of the test set.

Billion Word Benchmark (Chelba et al., 2014) using a sampled softmax with 8192 negative samples per batch. Overall, the averaged forward and backward perplexities are comparable across the models with values ranging from 37.5 for the 4-layer LSTM to 44.5 for the Gated CNN. To our knowledge, this is the first time that the Transformer has been shown to provide competitive results for language modeling. While it is possible to reduce perplexities for all models by scaling up, our goal is to compare representations across architectures for biLMs of approximately equal skill, as measured by perplexity.

The Transformer and CNN based models are faster than the LSTM based ones for our hyperparameter choices, with speed ups of 3-5X for the contextual layers over the 2-layer LSTM model.[5] Speed ups are relatively faster in the single element batch scenario where the sequential LSTM is most disadvantaged, but are still 2.3-3X for a 64 sentence batch. As the inference speed for the character based word embeddings could be mostly eliminated in a production setting, the table lists timings for both the contextual layers and all layers of the biLM necessary to compute context vectors. We also note that the faster architectures will allow training to scale to large unlabeled corpora, which has been shown to improve the quality of biLM representations for syntactic tasks (Zhang and Bowman, 2018).

4 Evaluation as word representations

In this section, we evaluate the quality of the pre-trained biLM representations as ELMo-like contextual word vectors in state-of-the-art models across a suite of four benchmark NLP tasks. To do so, we ran a series of controlled trials by swapping out pre-trained GloVe vectors (Pennington et al., 2014) for contextualized word vectors from each biLM computed by applying the learned weighted average ELMo pooling from Peters et al. (2018).[6] Each task model only includes one type of pre-trained word representation, either GloVe or contextual, this is a direct test of the transferability of the word representations. In addition, to isolate the general purpose LM representations from any task specific supervision, we did not fine tune the LM weights.

Table 2 shows the results. Across all tasks, the LSTM architectures perform the best. All architectures improve significantly over the GloVe only baseline, with relative improvements of 13% – 25% for most tasks and architectures. The gains for MultiNLI are more modest, with relative improvements over GloVe ranging from 6% for the Gated CNN to 13% for the 4-layer LSTM. The remainder of this section provides a description of the individual tasks and models with details in the Appendix.

4.1 MultiNLI

The MultiNLI dataset (Williams et al., 2018) contains crowd sourced textual entailment annotations across five diverse domains for training and an additional five domains for testing. Our model is a re-implementation of the ESIM sequence model (Chen et al., 2017). It first uses a biLSTM to encode the premise and hypothesis, then computes an attention matrix followed by a local inference layer, another biLSTM inference composition layer, and finally a pooling operation before

[5] While the CNN and Transformer implementations are reasonably well optimized, the LSTM biLM is not as it does not use an optimized CUDA kernel due to the use of the projection cell.

[6] Generally speaking, we found adding pre-trained GloVe vectors in addition to the biLM representations provided a small improvement across the tasks.

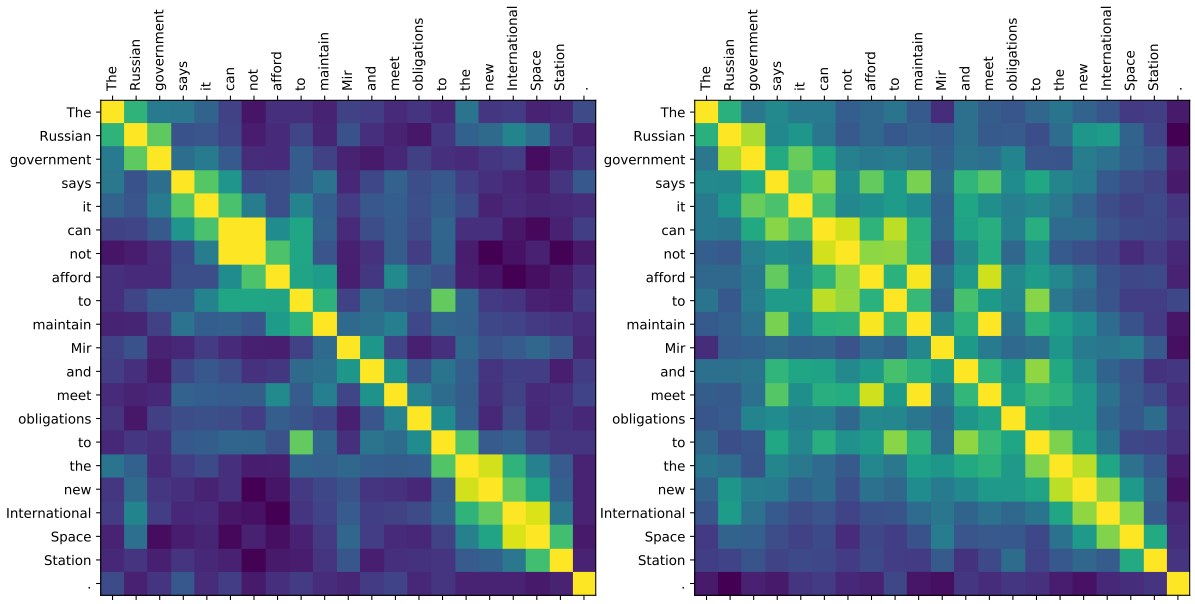


Figure 1: Visualization of contextual similarity between all word pairs in a single sentence using the 4-layer LSTM. The left panel uses context vectors from the bottom LSTM layer while the right panel uses the top LSTM layer. Lighter yellow-colored areas have higher contextual similarity.

the output layer. With the 2-layer LSTM ELMo representations, it is state-of-the-art for SNLI (Peters et al., 2018). As shown in Table 2, the LSTMs perform the best, with the Transformer accuracies 0.2% / 0.6% (matched/mismatched) less then the 2-layer LSTM. In addition, the contextual representations reduce the matched/mismatched performance differences showing that the biLMs can help mitigate domain effects. The ESIM model with the 4-layer LSTM ELMo-like embeddings sets a new state-of-the-art for this task, exceeding the highest previously published result by 1.3% matched and 1.9% mismatched from Gong et al. (2018).

4.2 Semantic Role Labeling

The Ontonotes 5.0 Dataset (Pradhan et al., 2013) contains predicate argument annotations for a variety of types of text, including conversation logs, web data, and biblical extracts. For our model, we use the deep biLSTM from He et al. (2017) who modeled SRL as a BIO tagging task. With ELMo representations, it is state-of-the-art for this task (Peters et al., 2018). For this task, the LSTM based word representations perform the best, with absolute improvements of 0.6% of the 4-layer LSTM over the Transformer and CNN.

4.3 Constituency parsing

The Penn Treebank (Marcus et al., 1993) contains phrase structure annotation for approximately 40k sentences sourced from the Wall Street Journal. Our model is the Reconciled Span Parser (RSP; Joshi et al., 2018), which, using ELMo representations, achieved state of the art performance for this task. As shown in Table 2, the LSTM based models demonstrate the best performance with a 0.2% and 1.0% improvement over the Transformer and CNN models, respectively. Whether the explicit recurrence structure modeled with the biLSTM in the RSP is important for parsing is explored in Sec. 5.3.

4.4 Named entity recognition

The CoNLL 2003 NER task (Sang and Meulder, 2003) provides entity annotations for approximately 20K sentences from the Reuters RCV1 news corpus. Our model is a re-implementation of the state-of-the-art system in Peters et al. (2018) with a character based CNN word representation, two biLSTM layers and a conditional random field (CRF) loss (Lafferty et al., 2001). For this task, the 2-layer LSTM performs the best, with averaged F_1 0.4% - 0.8% higher then the other biLMs averaged across five random seeds.

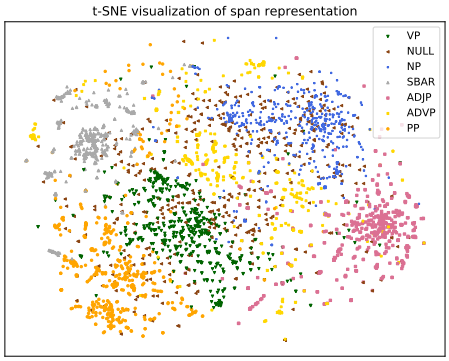


Figure 2: t-SNE visualization of 3K random chunks and 500 unlabeled spans ("NULL") from the CoNLL 2000 chunking dataset.

5 Properties of contextual vectors

In this section, we examine the intrinsic properties of contextual vectors learned with biLMs, focusing on those that are independent of the architecture details. In particular, we seek to understand how familiar types of linguistic information such as syntactic or coreferent relationships are represented throughout the depth of the network. Our experiments show that deep biLMs learn representations that vary with network depth, from morphology in the word embedding layer, to local syntax in the lowest contextual layers, to semantic relationships such as coreference in the upper layers.

We gain intuition and motivate our analysis by first considering the inter-sentence contextual similarity of words and phrases (Sec. 5.1). Then, we show that, in contrast to traditional word vectors, the biLM word embeddings capture little semantic information (Sec. 5.2) that is instead represented in the contextual layers (Sec. 5.3). Our analysis moves beyond single tokens by showing that a simple span representation based on the context vectors captures elements of phrasal syntax.

5.1 Contextual similarity

Nearest neighbors using cosine similarity are a popular way to visualize the relationships encoded in word vectors and we can apply a similar method to context vectors. As the biLMs use context vectors to pass information between layers in the network, this allows us to visualize how information is represented throughout the network.

Intra-sentence similarity Fig. 1 shows the intra-sentence contextual similarity between all pairs of words in single sentence using the 4-layer LSTM.[7] From the figure, we make several observations. First, the lower layer (left) captures mostly local information, while the top layer (right) represents longer range relationships. Second, at the lowest layer the biLM tends to place words from the same syntactic constituents in similar parts of the vector space. For example, the words in the noun phrase "the new international space station" are clustered together, similar to "can not" and "The Russian government".

In addition, we can see how the biLM is implicitly learning other linguistic information in the upper layer. For example, all of the verbs ("says", "can", "afford", "maintain", "meet") have high similarity suggesting the biLM is capturing part-of-speech information. We can also see some hints that the model is implicitly learning to perform coreference resolution by considering the high contextual similarity of "it" to "government", the head of "it"'s antecedent span. Section 5.3 provides empirical support for these observations.

Span representations The observation that the biLM's context vectors abruptly change at syntactic boundaries suggests we can also use them to form representations of spans, or consecutive token sequences. To do so, given a span of S tokens from indices s_0 to s_1, we compute a span representation $\mathbf{s}_{(s_0,s_1),i}$ at layer i by concatenating the first and last context vectors with the element wise product and difference of the first and last vectors:

$$\mathbf{s}_{(s_0,s_1),i} = [\mathbf{h}_{s_0,i}; \mathbf{h}_{s_1,i}; \mathbf{h}_{s_0,i} \odot \mathbf{h}_{s_1,i}; \mathbf{h}_{s_0,i} - \mathbf{h}_{s_1,i}].$$

Figure 2 shows a t-SNE (Maaten and Hinton, 2008) visualization of span representations of 3,000 labeled chunks and 500 spans not labeled as chunks from the CoNLL 2000 chunking dataset (Sang and Buchholz, 2000), from the first layer of the 4-layer LSTM. As we can see, the spans are clustered by chunk type confirming our intuition that the span representations capture elements of syntax. Sec. 5.3 evaluates whether we can use these span representations for constituency parsing.

Unsupervised pronominal coref We hypothesize that the contextual similarity of coreferential mentions should be similar, as in many cases it is possible to replace them with their referent. If true, we should be able to use contextual similarity to perform unsupervised coreference reso-

[7] See appendix for visualizations of the other models.

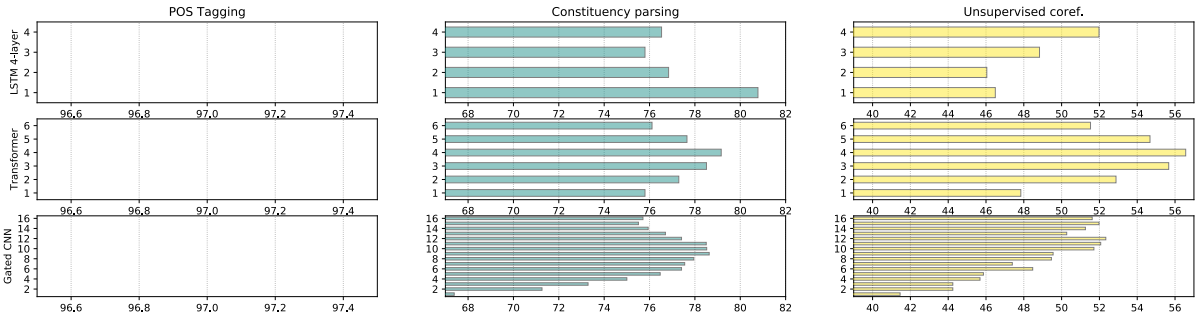


Figure 3: Various methods of probing the information stored in context vectors of deep biLMs. Each panel shows the results for all layers from a single biLM, with the first layer of contextual representations at the bottom and last layer at the top. From top to bottom, the figure shows results from the 4-layer LSTM, the Transformer and Gated CNN models. From left to right, the figure shows linear POS tagging accuracy (%; Sec. 5.3), linear constituency parsing (F_1; Sec. 5.3), and unsupervised pronominal coreference accuracy (%; Sec. 5.1).

Representation	Syntactic	Semantic
GloVe	77.9	**79.2**
n-gram hash	72.3	0.5
LSTM 4-layer	74.2	11.5
Transformer	**87.1**	48.8
Gated CNN	83.6	26.3

Table 3: Accuracy (%) for word vector analogies. In addition to the 300 dimension 840B GloVe vectors, the table contains results from a character n-gram hash and the context insensitive word embedding layer ($\mathbf{x}_k$) from the biLMs.

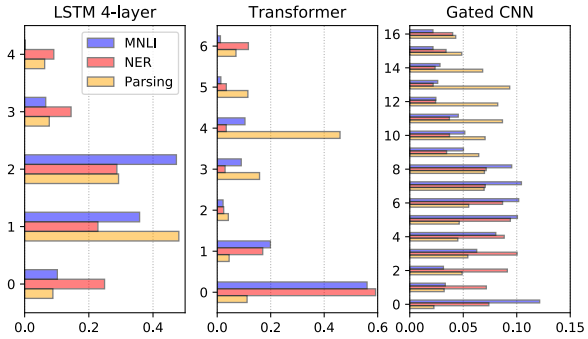


Figure 4: Normalized layer weights $\mathbf{s}$ for the tasks in Sec. 4. The vertical axis indexes the layer in the biLM, with layer 0 the word embedding $\mathbf{x}_k$.

lution. To test this, we designed an experiment as follows. To rule out trivially high mention-mention similarities due to lexical overlap, we restricted to pronominal coreference resolution. We took all sentences from the development set of the OntoNotes annotations in the CoNLL 2012 shared task (Pradhan et al., 2012) that had a third-person personal pronoun[8] and antecedent in the same sentence (904 sentences), and tested whether a system could identify the head word of the antecedent span given the pronoun location. In addition, by restricting to pronouns, systems are forced to rely on context to form their representation of the pronoun, as the surface form of the pronoun is uninformative. As an upper bound on performance, the state-of-the-art coreference model from Lee et al. (2017)[9] finds an antecedent span with the head word 64% of the time. As a lower bound on performance, a simple baseline that chooses the closest noun occurring before the pronoun has an accuracy of 27%, and one that chooses the first noun in the sentence has an accuracy of 35%. If we add an additional rule and further restrict to antecedent nouns matching the pronoun in number, the accuracies increase to 41% and 47% respectively.

To use contextual representations to solve this task, we first compute the mean context vector of the smallest constituent with more then one word containing the pronoun and subtract it from the pronoun's context vector. This step is motivated by the above observation that local syntax is the dominant signal in the contextualized word vectors, and removing it improves the accuracies of our method. Then, we choose the noun with the highest contextual similarity to the adjusted context vector that occurs before the pronoun and matches it in number.

The right hand column of Fig. 3 shows the results for all layers of the biLMs. Accuracies for the

[8] he, him, she, her, it, them, they

[9] http://allennlp.org/models

models peak between 52% and 57%, well above the baseline, with the Transformer overall having the highest accuracy. Interestingly, accuracies only drop 2-3% compared to 12-14% in the baseline if we remove the assumption of number agreement and simply consider all nouns, highlighting that the biLMs are to a large extent capturing number agreement across coreferent clusters. Finally, accuracies are highest at layers near the top of each model, showing that the upper layer representations are better at capturing longer range coreferent relationships then lower layers.

5.2 Context independent word representation

The word analogy task introduced in Mikolov et al. (2013) are commonly used as intrinsic evaluations of word vectors. Here, we use them to compare the word embedding layer from the biLMs to word vectors. The task has two types of analogies: syntactic with examples such as "bird:birds :: goat:goats", and semantic with examples such as "Athens:Greece :: Oslo:Norway". Traditional word vectors score highly on both sections. However, as shown in Table 3, the word embedding layer $\mathbf{x}_k$ from the biLMs is markedly different with syntactic accuracies on par or better then GloVe, but with very low semantic accuracies. To further highlight this distinction, we also computed a purely orthographically based word vector by hashing all character 1, 2, and 3-grams in a word into a sparse 300 dimensional vector. As expected, vectors from this method had near zero accuracy in the semantic portion, but scored well on the syntactic portion, showing that most of these analogies can be answered with morphology alone. As a result, we conclude that the word representation layer in deep biLMs is only faithfully encoding morphology with little semantics.

5.3 Probing contextual information

In this section, we quantify some of the anecdotal observations made in Sec. 5.1. To do so, we adopt a series of linear probes (Belinkov et al., 2017) with two NLP tasks to test the contextual representations in each model layer for each biLM architecture. In addition to examining single tokens, we also depart from previous work by examining to what extent the span representations capture phrasal syntax.

Our results show that all biLM architectures learn syntax, including span-based syntax; and part-of-speech information is captured at lower layers then constituent structure. When combined with the coreference accuracies in Sec. 5.1 that peak at even higher layers, this supports our claim that deep biLMs learn a hierarchy of contextual information.

POS tagging Peters et al. (2018) showed that the contextual vectors from the first layer of the 2-layer LSTM plus a linear classifier was near state-of-the-art for part-of-speech tagging. Here, we test whether this result holds for the other architectures. The second row of Fig. 3 shows tagging accuracies for all layers of the biLMs evaluated with the Wall Street Journal portion of Penn Treebank (Marcus et al., 1993). Accuracies for all of the models are high, ranging from 97.2 to 97.4, and follow a similar trend with maximum values at lower layers (bottom layer for LSTM, second layer for Transformer, and third layer for CNN).

Constituency parsing Here, we test whether the span representations introduced in Sec. 5.1 capture enough information to model constituent structure. Our linear model is a very simple and independently predicts the constituent type for all possible spans in a sentence using a linear classifier and the span representation. Then, a valid tree is built with a greedy decoding step that reconciles overlapping spans with an ILP, similar to Joshi et al. (2018).

The third row in Fig. 3 shows the results. Remarkably, predicting spans independently using the biLM representations alone has F_1 of near 80% for the best layers from each model. For comparison, a linear model using GloVe vectors performs very poorly, with F_1 of 18.1%. Across all architectures, the layers best suited for constituency parsing are at or above the layers with maximum POS accuracies as modeling phrasal syntactic structure requires a wider context then token level syntax. Similarity, the layers most transferable to parsing are at or below the layers with maximum pronominal coreference accuracy in all models, as constituent structure tends to be more local than coreference (Kuncoro et al., 2017).

5.4 Learned layer weights

Fig. 4 plots the softmax-normalized layer weights $\mathbf{s}$ from each biLM, learned as part of the tasks in Sec. 4. The SRL model weights are omitted as they close to constant since we had to regularize them to stabilize training. For constituency parsing, $\mathbf{s}$ mirrors the layer wise linear parsing results,

with the largest weights near or at the same layers as maximum linear parsing. For both NER and MultiNLI, the Transformer focuses heavily on the word embedding layer, $\mathbf{x}_k$, and the first contextual layer. In all cases, the maximum layer weights occur below the top layers as the most transferable contextual representations tend to occur in the middle layers, while the top layers specialize for language modeling.

6 Related work

In addition to biLM-based representations, McCann et al. (2017) learned contextualized vectors with a neural machine translation system (CoVe). However, as Peters et al. (2018) showed the biLM based representations outperformed CoVe in all considered tasks, we focus exclusively on biLMs.

Liu et al. (2018) proposed using densely connected RNNs and layer pruning to speed up the use of context vectors for prediction. As their method is applicable to other architectures, it could also be combined with our approach.

Several prior studies have examined the learned representations in RNNs. Karpathy et al. (2015) trained a character LSTM language model on source code and showed that individual neurons in the hidden state track the beginning and end of code blocks. Linzen et al. (2016) assessed whether RNNs can learn number agreement in subject-verb dependencies. Our analysis in Sec. 5.1 showed that biLMs also learn number agreement for coreference. Kádár et al. (2017) attributed the activation patters of RNNs to input tokens and showed that a RNN language model is strongly sensitive to tokens with syntactic functions. Belinkov et al. (2017) used linear classifiers to determine whether neural machine translation systems learned morphology and POS tags. Concurrent with our work, Khandelwal et al. (2018) studied the role of context in influencing language model predictions, Gaddy et al. (2018) analyzed neural constituency parsers, Blevins et al. (2018) explored whether RNNs trained with several different objectives can learn hierarchical syntax, and Conneau et al. (2018) examined to what extent sentence representations capture linguistic features. Our intrinsic analysis is most similar to Belinkov et al. (2017); however, we probe span representations in addition to word representations, evaluate the transferability of the biLM representations to semantic tasks in addition to syntax tasks, and consider a wider variety of neural architectures in addition to RNNs.

Other work has focused on attributing network predictions. Li et al. (2016) examined the impact of erasing portions of a network's representations on the output, Sundararajan et al. (2017) used a gradient based method to attribute predictions to inputs, and Murdoch et al. (2018) decomposed LSTMs to interpret classification predictions. In contrast to these approaches, we explore the types of contextual information encoded in the biLM internal states instead of focusing on attributing this information to words in the input sentence.

7 Conclusions and future work

We have shown that deep biLMs learn a rich hierarchy of contextual information, both at the word and span level, and that this is captured in three disparate types of network architectures. Across all architecture types, the lower biLM layers specialize in local syntactic relationships, allowing the higher layers to model longer range relationships such as coreference, and to specialize for the language modeling task at the top most layers. These results highlight the rich nature of the linguistic information captured in the biLM's representations and show that biLMs act as a general purpose feature extractor for natural language, opening the way for computer vision style feature re-use and transfer methods.

Our results also suggest avenues for future work. One open question is to what extent can the quality of biLM representations be improved by simply scaling up model size or data size? As our results have show that computationally efficient architectures also learn high quality representations, one natural direction would be exploring the very large model and data regime.

Despite their successes biLM representations are far from perfect; during training, they have access to only surface forms of words and their order, meaning deeper linguistic phenomena must be learned "tabula rasa". Infusing models with explicit syntactic structure or other linguistically motivated inductive biases may overcome some of the limitations of sequential biLMs. An alternate direction for future work combines the purely unsupervised biLM training objective with existing annotated resources in a multitask or semi-supervised manner.

References

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James R. Glass. 2017. What do neural machine translation models learn about morphology? In *ACL*.

T. Blevins, O. Levy, and L. Zettlemoyer. 2018. Deep RNNs Encode Soft Hierarchical Syntax. In *ACL*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH*.

Qian Chen, Xiao-Dan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *ACL*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *ACL*.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *NIPS*.

Yann Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *ICML*.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. What's going on in neural constituency parsers? An analysis. In *NAACL*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *ACL workshop for NLP Open Source Software*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML*.

Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural language inference over interaction space. In *ICLR*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Luheng He, Kenton Lee, Mike Lewis, and Luke S. Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *ACL*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Vidur Joshi, Matthew Peters, and Mark Hopkins. 2018. Extending a parser to distant domains using a few dozen partially annotated examples. In *ACL*.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv*, abs/1602.02410.

Ákos Kádár, Grzegorz Chrupala, and Afra Alishahi. 2017. Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics*, 43:761–780.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. In *ICLR workshop*.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In *ACL*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. In *AAAI 2016*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *ACL*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *ACL*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *EACL*.

John D. Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Kenton Lee, Luheng He, Mike Lewis, and Luke S. Zettlemoyer. 2017. End-to-end neural coreference resolution. In *EMNLP*.

Jiwei Li, Will Monroe, and Daniel Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv*, abs/1612.08220.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. In *TACL*.

Liyuan Liu, Xiang Ren, Jingbo Shang, Jian Peng, and Jiawei Han. 2018. Efficient contextualized representation: Language model pruning for sequence labeling. In *EMNLP*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS 2017*.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *ICLR*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing lstm language models. In *ICLR*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

W. James Murdoch, Peter J. Liu, and Bin Yu. 2018. Beyond word importance: Contextual decomposition to extract interactions from lstms. In *ICLR*.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. In *arXiv*, volume abs/1609.03499.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *CoNLL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *EMNLP-CoNLL Shared Task*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Prajit Ramachandran, Peter J. Liu, and Quoc V. Le. 2017. Unsupervised pretraining for sequence to sequence learning. In *EMNLP*.

Hasim Sak, Andrew W. Senior, and Franoise Beaufays. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH*.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task chunking. In *CoNLL/LLL*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *NIPS*.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *ICML*.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *AAAI*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*.

Matthew D. Zeiler. 2012. ADADELTA: An adaptive learning rate method. *arXiv*, abs/1212.5701.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833.

Kelly Zhang and Samuel R. Bowman. 2018. Language modeling teaches you more syntax than translation does: Lessons learned through auxiliary task analysis. *arXiv preprint*.

Appendix to accompany "Dissecting contextual word embeddings: Architecture and Representation"

A biLM hyperparameters

For consistency and a fair comparison, all biLMs use a 512 dimensional representation in each direction at each layer, providing a 1024 dimensional contextual representation at each layer. All models use the same character based word embedding layer $\mathbf{x}_k$, with the exception of the 4-layer LSTM as described below. All systems use residual connections between the contextual layers (He et al., 2016).

LSTM Hyperparameters for the 4-layer LSTM closely follow those from the 2-layer ELMo model in Peters et al. (2018). It has four layers, with each direction in each layer having a 4096 dimension hidden state and a 512 dimensional projection. To reduce training time, the character based word representation is simplified from the other models. It uses the same 2048 character n-gram CNN filters as the other models, but moves the projection layer from 2048 to 512 dimensions after the convolutions and before the two highway layers.

Transformer The Transformer biLM uses six layers, each with eight attention heads and a 2048 hidden dimension for the feed forward layers. 10% dropout was used after the word embedding layer $\mathbf{x}_k$, multi-headed attention, the hidden layers in the feed forward layers and before the residual connections. Optimization used batches of 12,000 tokens split across 4 GPUs with, using the learning rate schedule from Vaswani et al. (2017) with 2,000 warm up steps. The final model weights were averaged over 10 consecutive checkpoints.

Gated CNN The Gated CNN has 16 layers of $[4, 512]$ residual blocks with 5% dropout between each block. Optimization used Adagrad with linearly increasing learning rate from 0 to 0.4 over the first 10,000 batches. The batch size was 7,500 split across 4 GPUs. Gradients were clipped if their norm exceeded 5.0. The final model weights were averaged over 10 consecutive checkpoints.

B Task model hyperparameters

MultiNLI Our implementation of the ESIM model uses 300 dimensions for all LSTMs and all feed forward layers. For regularization we used 50% dropout at the input to each LSTM and after each feed forward layer. Optimization used Adam with learning rate 0.0004 and batch size of 32 sentence pairs.

Semantic Role Labeling The SRL model uses the reimplementation of He et al. (2017) from Gardner et al. (2018). Word representations are concatenated with a 100 dimensional binary predicate representation, specifying the location of the predicate for the given frame. This is passed through an 8 layer bidirectional LSTM, where the layers alternate between forward and backward directions. Highway connections and variational dropout are used between every LSTM layer. Models are with a batch size of 80 sentences using Adadelta (Zeiler, 2012) with an initial learning rate of 1.0 and rho 0.95.

Constituency Parsing The constituency Parser is a reimplementation of Joshi et al. (2018), available in AllenNLP (Gardner et al., 2018). Word representations from the various biLMs models are passed through a two layer bidirectional LSTM with hidden size 250 and 0.2 dropout. Then, the span representations are passed through a feedforward layer (dropout 0.1, hidden size 250) with a relu non-linearity before classification. We use a batch size of 64 and gradients are normalized to have a global norm ≤ 5.0. Optmization uses Adadelta with initial learning rate of 1.0 and rho 0.95.

NER The NER model concatenates 128 character CNN filters of width 3 characters to the pretrained word representations. It uses two LSTM layers with hidden size 200 with 50% dropout at input and output. The final layer is a CRF, with constrained decoding to enforce valid tag sequences. We employ early stopping on the development set and report average F_1 across five random seeds.

C Contextual similarities

Figures 5, 6, and 7 show contextual similarities similar to Figure 1 for all layers from the 4-layer LSTM, the Transformer and gated CNN biLMs.

D Layer diagnostics

Tables 4, 5 and 6 list full results corresponding to the top three rows in Fig. 3.

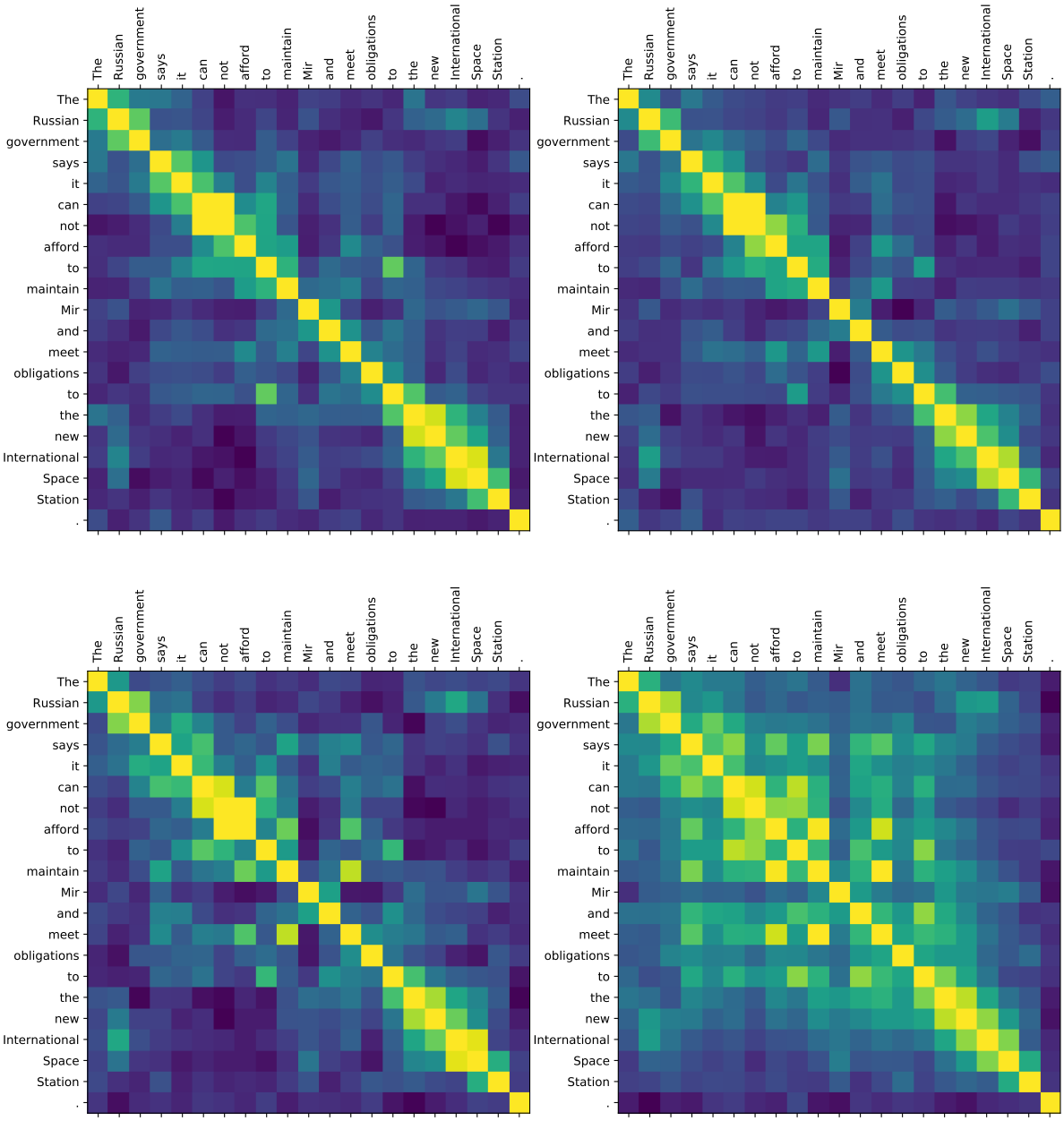


Figure 5: Visualization of contextual similarities from the 4-layer LSTM biLM. The first layer is at top left and last layer at bottom right, with the layer indices increasing from left to right and top to bottom in the image.

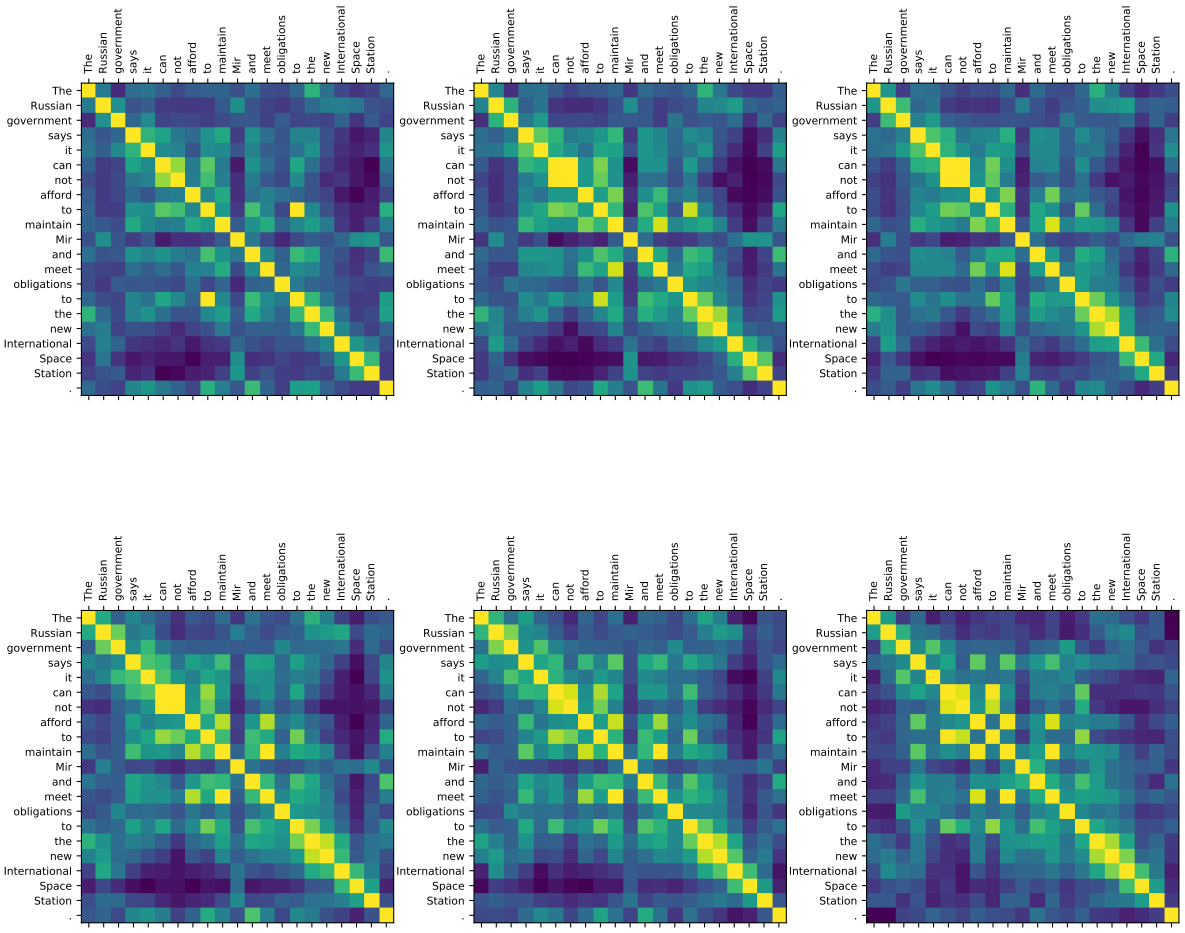


Figure 6: Visualization of contextual similarities from the Transformer biLM. The first layer is at top left and last layer at bottom right, with the layer indices increasing from left to right and top to bottom in the image.

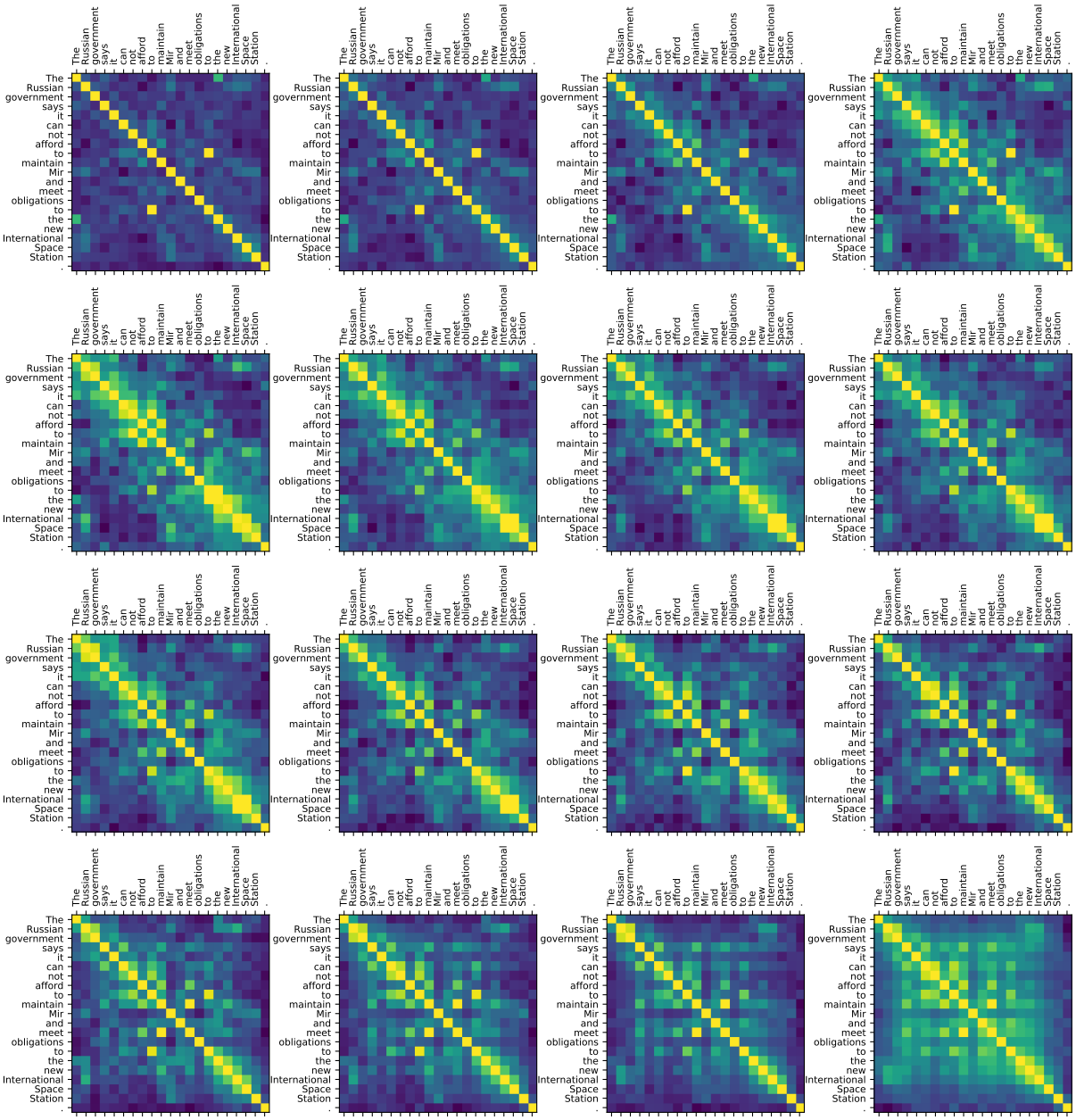


Figure 7: Visualization of contextual similarities from the gated CNN biLM. The first layer is at top left and last layer at bottom right, with the layer indices increasing from left to right and top to bottom in the image.

Layer	Accuracy
Elmo - 4 Layer	
Layer 1	46.5
Layer 2	46.0
Layer 3	48.8
Layer 4	**52.0**
Transformer	
Layer 1	47.8
Layer 2	52.9
Layer 3	55.7
Layer 4	**56.7**
Layer 5	54.7
Layer 6	51.5
Gated CNN	
Layer 1	41.5
Layer 2	44.2
Layer 3	44.2
Layer 4	45.7
Layer 5	45.9
Layer 6	48.5
Layer 7	47.4
Layer 8	49.6
Layer 9	51.7
Layer 10	47.8
Layer 11	52.1
Layer 12	**52.4**
Layer 13	50.3
Layer 14	51.3
Layer 15	52.0
Layer 16	51.6

Table 4: Unsupervised pronominal accuracies using the CoNLL 2012 development set.

Layer	Accuracy
GloVe Only	88.61
Elmo - 4 Layer	
Layer 1	**97.36**
Layer 2	97.16
Layer 3	96.90
Layer 4	96.58
Weighted Layers	97.22
Transformer	
Layer 1	97.30
Layer 2	97.35
Layer 3	97.25
Layer 4	97.15
Layer 5	96.90
Layer 6	96.82
Weighted Layers	**97.48**
Gated CNN	
Layer 1	97.09
Layer 2	97.16
Layer 3	97.19
Layer 4	97.16
Layer 5	97.11
Layer 6	97.09
Layer 7	97.08
Layer 8	97.01
Layer 9	97.00
Layer 10	96.97
Layer 11	96.96
Layer 12	96.97
Layer 13	96.85
Layer 14	96.80
Layer 15	96.64
Layer 16	96.43
Weighted Layers	**97.26**

Table 5: POS tagging accuracies for the linear models on the PTB dev set.

Layer	F_1	Precision	Recall
GloVe Only	18.1	11.2	45.9
LSTM - 4 Layer			
Layer 1	80.8	87.0	74.1
Layer 2	76.8	83.7	71.4
Layer 3	75.8	84.6	68.7
Layer 4	76.5	81.6	72.1
Weighted Layers	**80.9**	**87.9**	**75.4**
Transformer			
Layer 1	75.8	80.4	71.7
Layer 2	77.3	82.6	72.6
Layer 3	78.5	82.5	75.0
Layer 4	79.2	80.5	77.9
Layer 5	77.7	78.3	77.0
Layer 6	76.1	77.3	75.0
Weighted Layers	**82.8**	**87.5**	**78.6**
Gated CNN			
Layer 1	67.4	79.5	58.5
Layer 2	71.3	81.8	63.1
Layer 3	73.3	83.8	65.1
Layer 4	75.0	84.6	67.3
Layer 5	76.5	85.3	69.3
Layer 6	77.4	85.6	70.7
Layer 7	77.6	85.9	70.7
Layer 8	78.0	86.9	71.1
Layer 9	78.6	85.0	**73.3**
Layer 10	78.5	85.9	72.3
Layer 11	78.5	84.5	**73.3**
Layer 12	77.4	85.4	70.8
Layer 13	76.7	84.7	70.1
Layer 14	75.9	83.3	69.9
Layer 15	75.5	82.8	69.4
Layer 16	75.7	83.0	69.6
Weighted Layers	**78.6**	**86.4**	72.0

Table 6: Labeled Bracketing F_1, Precision and Recall for the linear parsing models on the PTB dev set.

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).

1 Introduction

Language model pre-training has been shown to be effective for improving many natural language processing tasks (Dai and Le, 2015; Peters et al., 2018a; Radford et al., 2018; Howard and Ruder, 2018). These include sentence-level tasks such as natural language inference (Bowman et al., 2015; Williams et al., 2018) and paraphrasing (Dolan and Brockett, 2005), which aim to predict the relationships between sentences by analyzing them holistically, as well as token-level tasks such as named entity recognition and question answering, where models are required to produce fine-grained output at the token level (Tjong Kim Sang and De Meulder, 2003; Rajpurkar et al., 2016).

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pre-trained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations.

We argue that current techniques restrict the power of the pre-trained representations, especially for the fine-tuning approaches. The major limitation is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training. For example, in OpenAI GPT, the authors use a left-to-right architecture, where every token can only attend to previous tokens in the self-attention layers of the Transformer (Vaswani et al., 2017). Such restrictions are sub-optimal for sentence-level tasks, and could be very harmful when applying fine-tuning based approaches to token-level tasks such as question answering, where it is crucial to incorporate context from both directions.

In this paper, we improve the fine-tuning based approaches by proposing BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. BERT alleviates the previously mentioned unidirectionality constraint by using a "masked language model" (MLM) pre-training objective, inspired by the Cloze task (Taylor, 1953). The masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked

word based only on its context. Unlike left-to-right language model pre-training, the MLM objective enables the representation to fuse the left and the right context, which allows us to pre-train a deep bidirectional Transformer. In addition to the masked language model, we also use a "next sentence prediction" task that jointly pre-trains text-pair representations. The contributions of our paper are as follows:

- We demonstrate the importance of bidirectional pre-training for language representations. Unlike Radford et al. (2018), which uses unidirectional language models for pre-training, BERT uses masked language models to enable pre-trained deep bidirectional representations. This is also in contrast to Peters et al. (2018a), which uses a shallow concatenation of independently trained left-to-right and right-to-left LMs.
- We show that pre-trained representations reduce the need for many heavily-engineered task-specific architectures. BERT is the first fine-tuning based representation model that achieves state-of-the-art performance on a large suite of sentence-level *and* token-level tasks, outperforming many task-specific architectures.
- BERT advances the state of the art for eleven NLP tasks. The code and pre-trained models are available at `https://github.com/google-research/bert`.

2 Related Work

There is a long history of pre-training general language representations, and we briefly review the most widely-used approaches in this section.

2.1 Unsupervised Feature-based Approaches

Learning widely applicable representations of words has been an active area of research for decades, including non-neural (Brown et al., 1992; Ando and Zhang, 2005; Blitzer et al., 2006) and neural (Mikolov et al., 2013; Pennington et al., 2014) methods. Pre-trained word embeddings are an integral part of modern NLP systems, offering significant improvements over embeddings learned from scratch (Turian et al., 2010). To pre-train word embedding vectors, left-to-right language modeling objectives have been used (Mnih and Hinton, 2009), as well as objectives to discriminate correct from incorrect words in left and right context (Mikolov et al., 2013).

These approaches have been generalized to coarser granularities, such as sentence embeddings (Kiros et al., 2015; Logeswaran and Lee, 2018) or paragraph embeddings (Le and Mikolov, 2014). To train sentence representations, prior work has used objectives to rank candidate next sentences (Jernite et al., 2017; Logeswaran and Lee, 2018), left-to-right generation of next sentence words given a representation of the previous sentence (Kiros et al., 2015), or denoising auto-encoder derived objectives (Hill et al., 2016).

ELMo and its predecessor (Peters et al., 2017, 2018a) generalize traditional word embedding research along a different dimension. They extract *context-sensitive* features from a left-to-right and a right-to-left language model. The contextual representation of each token is the concatenation of the left-to-right and right-to-left representations. When integrating contextual word embeddings with existing task-specific architectures, ELMo advances the state of the art for several major NLP benchmarks (Peters et al., 2018a) including question answering (Rajpurkar et al., 2016), sentiment analysis (Socher et al., 2013), and named entity recognition (Tjong Kim Sang and De Meulder, 2003). Melamud et al. (2016) proposed learning contextual representations through a task to predict a single word from both left and right context using LSTMs. Similar to ELMo, their model is feature-based and not deeply bidirectional. Fedus et al. (2018) shows that the cloze task can be used to improve the robustness of text generation models.

2.2 Unsupervised Fine-tuning Approaches

As with the feature-based approaches, the first works in this direction only pre-trained word embedding parameters from unlabeled text (Collobert and Weston, 2008).

More recently, sentence or document encoders which produce contextual token representations have been pre-trained from unlabeled text and fine-tuned for a supervised downstream task (Dai and Le, 2015; Howard and Ruder, 2018; Radford et al., 2018). The advantage of these approaches is that few parameters need to be learned from scratch. At least partly due to this advantage, OpenAI GPT (Radford et al., 2018) achieved previously state-of-the-art results on many sentence-level tasks from the GLUE benchmark (Wang et al., 2018a). Left-to-right language model-

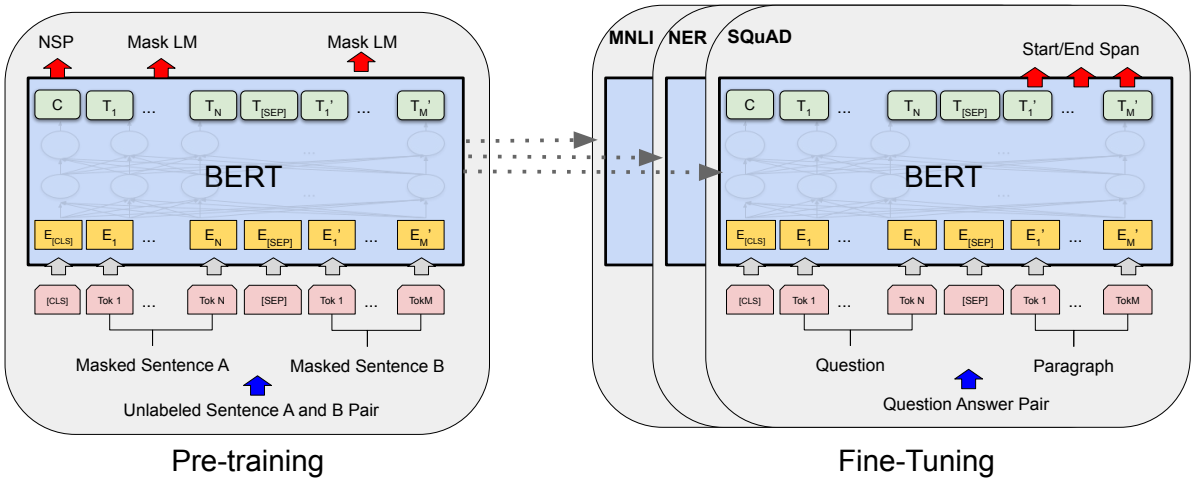


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. `[CLS]` is a special symbol added in front of every input example, and `[SEP]` is a special separator token (e.g. separating questions/answers).

ing and auto-encoder objectives have been used for pre-training such models (Howard and Ruder, 2018; Radford et al., 2018; Dai and Le, 2015).

2.3 Transfer Learning from Supervised Data

There has also been work showing effective transfer from supervised tasks with large datasets, such as natural language inference (Conneau et al., 2017) and machine translation (McCann et al., 2017). Computer vision research has also demonstrated the importance of transfer learning from large pre-trained models, where an effective recipe is to fine-tune models pre-trained with ImageNet (Deng et al., 2009; Yosinski et al., 2014).

3 BERT

We introduce BERT and its detailed implementation in this section. There are two steps in our framework: *pre-training* and *fine-tuning*. During pre-training, the model is trained on unlabeled data over different pre-training tasks. For fine-tuning, the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters. The question-answering example in Figure 1 will serve as a running example for this section.

A distinctive feature of BERT is its unified architecture across different tasks. There is minimal difference between the pre-trained architecture and the final downstream architecture.

Model Architecture BERT's model architecture is a multi-layer bidirectional Transformer encoder based on the original implementation described in Vaswani et al. (2017) and released in the `tensor2tensor` library.[1] Because the use of Transformers has become common and our implementation is almost identical to the original, we will omit an exhaustive background description of the model architecture and refer readers to Vaswani et al. (2017) as well as excellent guides such as "The Annotated Transformer."[2]

In this work, we denote the number of layers (i.e., Transformer blocks) as L, the hidden size as H, and the number of self-attention heads as A.[3] We primarily report results on two model sizes: $\textbf{BERT}_{\textbf{BASE}}$ (L=12, H=768, A=12, Total Parameters=110M) and $\textbf{BERT}_{\textbf{LARGE}}$ (L=24, H=1024, A=16, Total Parameters=340M).

$\text{BERT}_{\text{BASE}}$ was chosen to have the same model size as OpenAI GPT for comparison purposes. Critically, however, the BERT Transformer uses bidirectional self-attention, while the GPT Transformer uses constrained self-attention where every token can only attend to context to its left.[4]

[1] https://github.com/tensorflow/tensor2tensor

[2] http://nlp.seas.harvard.edu/2018/04/03/attention.html

[3] In all cases we set the feed-forward/filter size to be $4H$, i.e., 3072 for the $H = 768$ and 4096 for the $H = 1024$.

[4] We note that in the literature the bidirectional Trans-

Input/Output Representations To make BERT handle a variety of down-stream tasks, our input representation is able to unambiguously represent both a single sentence and a pair of sentences (e.g., ⟨ Question, Answer ⟩) in one token sequence. Throughout this work, a "sentence" can be an arbitrary span of contiguous text, rather than an actual linguistic sentence. A "sequence" refers to the input token sequence to BERT, which may be a single sentence or two sentences packed together.

We use WordPiece embeddings (Wu et al., 2016) with a 30,000 token vocabulary. The first token of every sequence is always a special classification token (`[CLS]`). The final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks. Sentence pairs are packed together into a single sequence. We differentiate the sentences in two ways. First, we separate them with a special token (`[SEP]`). Second, we add a learned embedding to every token indicating whether it belongs to sentence `A` or sentence `B`. As shown in Figure 1, we denote input embedding as E, the final hidden vector of the special `[CLS]` token as $C \in \mathbb{R}^H$, and the final hidden vector for the i^{th} input token as $T_i \in \mathbb{R}^H$.

For a given token, its input representation is constructed by summing the corresponding token, segment, and position embeddings. A visualization of this construction can be seen in Figure 2.

3.1 Pre-training BERT

Unlike Peters et al. (2018a) and Radford et al. (2018), we do not use traditional left-to-right or right-to-left language models to pre-train BERT. Instead, we pre-train BERT using two unsupervised tasks, described in this section. This step is presented in the left part of Figure 1.

Task #1: Masked LM Intuitively, it is reasonable to believe that a deep bidirectional model is strictly more powerful than either a left-to-right model or the shallow concatenation of a left-to-right and a right-to-left model. Unfortunately, standard conditional language models can only be trained left-to-right *or* right-to-left, since bidirectional conditioning would allow each word to indirectly "see itself", and the model could trivially predict the target word in a multi-layered context.

In order to train a deep bidirectional representation, we simply mask some percentage of the input tokens at random, and then predict those masked tokens. We refer to this procedure as a "masked LM" (MLM), although it is often referred to as a *Cloze* task in the literature (Taylor, 1953). In this case, the final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary, as in a standard LM. In all of our experiments, we mask 15% of all WordPiece tokens in each sequence at random. In contrast to denoising auto-encoders (Vincent et al., 2008), we only predict the masked words rather than reconstructing the entire input.

Although this allows us to obtain a bidirectional pre-trained model, a downside is that we are creating a mismatch between pre-training and fine-tuning, since the `[MASK]` token does not appear during fine-tuning. To mitigate this, we do not always replace "masked" words with the actual `[MASK]` token. The training data generator chooses 15% of the token positions at random for prediction. If the i-th token is chosen, we replace the i-th token with (1) the `[MASK]` token 80% of the time (2) a random token 10% of the time (3) the unchanged i-th token 10% of the time. Then, T_i will be used to predict the original token with cross entropy loss. We compare variations of this procedure in Appendix C.2.

Task #2: Next Sentence Prediction (NSP) Many important downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI) are based on understanding the *relationship* between two sentences, which is not directly captured by language modeling. In order to train a model that understands sentence relationships, we pre-train for a binarized *next sentence prediction* task that can be trivially generated from any monolingual corpus. Specifically, when choosing the sentences `A` and `B` for each pre-training example, 50% of the time `B` is the actual next sentence that follows `A` (labeled as `IsNext`), and 50% of the time it is a random sentence from the corpus (labeled as `NotNext`). As we show in Figure 1, C is used for next sentence prediction (NSP).[5] Despite its simplicity, we demonstrate in Section 5.1 that pre-training towards this task is very beneficial to both QA and NLI. [6]

former is often referred to as a "Transformer encoder" while the left-context-only version is referred to as a "Transformer decoder" since it can be used for text generation.

[5] The final model achieves 97%-98% accuracy on NSP.

[6] The vector C is not a meaningful sentence representation without fine-tuning, since it was trained with NSP.

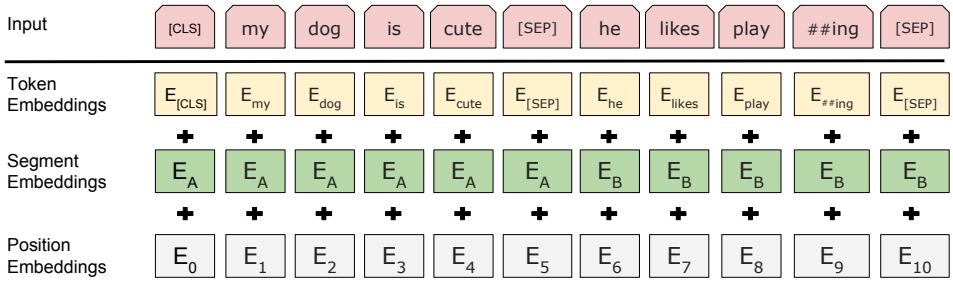


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

The NSP task is closely related to representation-learning objectives used in Jernite et al. (2017) and Logeswaran and Lee (2018). However, in prior work, only sentence embeddings are transferred to down-stream tasks, where BERT transfers all parameters to initialize end-task model parameters.

Pre-training data The pre-training procedure largely follows the existing literature on language model pre-training. For the pre-training corpus we use the BooksCorpus (800M words) (Zhu et al., 2015) and English Wikipedia (2,500M words). For Wikipedia we extract only the text passages and ignore lists, tables, and headers. It is critical to use a document-level corpus rather than a shuffled sentence-level corpus such as the Billion Word Benchmark (Chelba et al., 2013) in order to extract long contiguous sequences.

3.2 Fine-tuning BERT

Fine-tuning is straightforward since the self-attention mechanism in the Transformer allows BERT to model many downstream tasks—whether they involve single text or text pairs—by swapping out the appropriate inputs and outputs. For applications involving text pairs, a common pattern is to independently encode text pairs before applying bidirectional cross attention, such as Parikh et al. (2016); Seo et al. (2017). BERT instead uses the self-attention mechanism to unify these two stages, as encoding a concatenated text pair with self-attention effectively includes *bidirectional* cross attention between two sentences.

For each task, we simply plug in the task-specific inputs and outputs into BERT and fine-tune all the parameters end-to-end. At the input, sentence `A` and sentence `B` from pre-training are analogous to (1) sentence pairs in paraphrasing, (2) hypothesis-premise pairs in entailment, (3) question-passage pairs in question answering, and (4) a degenerate text-$\varnothing$ pair in text classification or sequence tagging. At the output, the token representations are fed into an output layer for token-level tasks, such as sequence tagging or question answering, and the `[CLS]` representation is fed into an output layer for classification, such as entailment or sentiment analysis.

Compared to pre-training, fine-tuning is relatively inexpensive. All of the results in the paper can be replicated in at most 1 hour on a single Cloud TPU, or a few hours on a GPU, starting from the exact same pre-trained model.[7] We describe the task-specific details in the corresponding subsections of Section 4. More details can be found in Appendix A.5.

4 Experiments

In this section, we present BERT fine-tuning results on 11 NLP tasks.

4.1 GLUE

The General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018a) is a collection of diverse natural language understanding tasks. Detailed descriptions of GLUE datasets are included in Appendix B.1.

To fine-tune on GLUE, we represent the input sequence (for single sentence or sentence pairs) as described in Section 3, and use the final hidden vector $C \in \mathbb{R}^H$ corresponding to the first input token (`[CLS]`) as the aggregate representation. The only new parameters introduced during fine-tuning are classification layer weights $W \in \mathbb{R}^{K \times H}$, where K is the number of labels. We compute a standard classification loss with C and W, i.e., $\log(\text{softmax}(CW^T))$.

[7] For example, the BERT SQuAD model can be trained in around 30 minutes on a single Cloud TPU to achieve a Dev F1 score of 91.0%.

[8] See (10) in `https://gluebenchmark.com/faq`.

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

We use a batch size of 32 and fine-tune for 3 epochs over the data for all GLUE tasks. For each task, we selected the best fine-tuning learning rate (among 5e-5, 4e-5, 3e-5, and 2e-5) on the Dev set. Additionally, for $\text{BERT}_{\text{LARGE}}$ we found that fine-tuning was sometimes unstable on small datasets, so we ran several random restarts and selected the best model on the Dev set. With random restarts, we use the same pre-trained checkpoint but perform different fine-tuning data shuffling and classifier layer initialization.[9]

Results are presented in Table 1. Both $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$ outperform all systems on all tasks by a substantial margin, obtaining 4.5% and 7.0% respective average accuracy improvement over the prior state of the art. Note that $\text{BERT}_{\text{BASE}}$ and OpenAI GPT are nearly identical in terms of model architecture apart from the attention masking. For the largest and most widely reported GLUE task, MNLI, BERT obtains a 4.6% absolute accuracy improvement. On the official GLUE leaderboard[10], $\text{BERT}_{\text{LARGE}}$ obtains a score of 80.5, compared to OpenAI GPT, which obtains 72.8 as of the date of writing.

We find that $\text{BERT}_{\text{LARGE}}$ significantly outperforms $\text{BERT}_{\text{BASE}}$ across all tasks, especially those with very little training data. The effect of model size is explored more thoroughly in Section 5.2.

4.2 SQuAD v1.1

The Stanford Question Answering Dataset (SQuAD v1.1) is a collection of 100k crowd-sourced question/answer pairs (Rajpurkar et al., 2016). Given a question and a passage from Wikipedia containing the answer, the task is to predict the answer text span in the passage.

As shown in Figure 1, in the question answering task, we represent the input question and passage as a single packed sequence, with the question using the A embedding and the passage using the B embedding. We only introduce a start vector $S \in \mathbb{R}^H$ and an end vector $E \in \mathbb{R}^H$ during fine-tuning. The probability of word i being the start of the answer span is computed as a dot product between T_i and S followed by a softmax over all of the words in the paragraph: $P_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}}$. The analogous formula is used for the end of the answer span. The score of a candidate span from position i to position j is defined as $S \cdot T_i + E \cdot T_j$, and the maximum scoring span where $j \geq i$ is used as a prediction. The training objective is the sum of the log-likelihoods of the correct start and end positions. We fine-tune for 3 epochs with a learning rate of 5e-5 and a batch size of 32.

Table 2 shows top leaderboard entries as well as results from top published systems (Seo et al., 2017; Clark and Gardner, 2018; Peters et al., 2018a; Hu et al., 2018). The top results from the SQuAD leaderboard do not have up-to-date public system descriptions available,[11] and are allowed to use any public data when training their systems. We therefore use modest data augmentation in our system by first fine-tuning on TriviaQA (Joshi et al., 2017) befor fine-tuning on SQuAD.

Our best performing system outperforms the top leaderboard system by +1.5 F1 in ensembling and +1.3 F1 as a single system. In fact, our single BERT model outperforms the top ensemble system in terms of F1 score. Without TriviaQA fine-

[9] The GLUE data set distribution does not include the Test labels, and we only made a single GLUE evaluation server submission for each of $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$.

[10] https://gluebenchmark.com/leaderboard

[11] QANet is described in Yu et al. (2018), but the system has improved substantially after publication.

System	Dev		Test	
	EM	F1	EM	F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

Table 2: SQuAD 1.1 results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

System	Dev		Test	
	EM	F1	EM	F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	86.3	89.0	86.9	89.5
#1 Single - MIR-MRC (F-Net)	-	-	74.8	78.0
#2 Single - nlnet	-	-	74.2	77.1
Published				
unet (Ensemble)	-	-	71.4	74.9
SLQA+ (Single)	-		71.4	74.4
Ours				
$\text{BERT}_{\text{LARGE}}$ (Single)	78.7	81.9	80.0	83.1

Table 3: SQuAD 2.0 results. We exclude entries that use BERT as one of their components.

tuning data, we only lose 0.1-0.4 F1, still outperforming all existing systems by a wide margin.[12]

4.3 SQuAD v2.0

The SQuAD 2.0 task extends the SQuAD 1.1 problem definition by allowing for the possibility that no short answer exists in the provided paragraph, making the problem more realistic.

We use a simple approach to extend the SQuAD v1.1 BERT model for this task. We treat questions that do not have an answer as having an answer span with start and end at the `[CLS]` token. The probability space for the start and end answer span positions is extended to include the position of the `[CLS]` token. For prediction, we compare the score of the no-answer span: $s_{\texttt{null}} = S{\cdot}C + E{\cdot}C$ to the score of the best non-null span $\hat{s_{i,j}} = \max_{j \geq i} S{\cdot}T_i + E{\cdot}T_j$. We predict a non-null answer when $\hat{s_{i,j}} > s_{\texttt{null}} + \tau$, where the threshold τ is selected on the dev set to maximize F1. We did not use TriviaQA data for this model. We fine-tuned for 2 epochs with a learning rate of 5e-5 and a batch size of 48.

The results compared to prior leaderboard entries and top published work (Sun et al., 2018; Wang et al., 2018b) are shown in Table 3, excluding systems that use BERT as one of their components. We observe a +5.1 F1 improvement over the previous best system.

[12] The TriviaQA data we used consists of paragraphs from TriviaQA-Wiki formed of the first 400 tokens in documents, that contain at least one of the provided possible answers.

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
OpenAI GPT	-	78.0
$\text{BERT}_{\text{BASE}}$	81.6	-
$\text{BERT}_{\text{LARGE}}$	**86.6**	**86.3**
Human (expert)†	-	85.0
Human (5 annotations)†	-	88.0

Table 4: SWAG Dev and Test accuracies. †Human performance is measured with 100 samples, as reported in the SWAG paper.

4.4 SWAG

The Situations With Adversarial Generations (SWAG) dataset contains 113k sentence-pair completion examples that evaluate grounded commonsense inference (Zellers et al., 2018). Given a sentence, the task is to choose the most plausible continuation among four choices.

When fine-tuning on the SWAG dataset, we construct four input sequences, each containing the concatenation of the given sentence (sentence `A`) and a possible continuation (sentence `B`). The only task-specific parameters introduced is a vector whose dot product with the `[CLS]` token representation C denotes a score for each choice which is normalized with a softmax layer.

We fine-tune the model for 3 epochs with a learning rate of 2e-5 and a batch size of 16. Results are presented in Table 4. $\text{BERT}_{\text{LARGE}}$ outperforms the authors' baseline ESIM+ELMo system by +27.1% and OpenAI GPT by 8.3%.

5 Ablation Studies

In this section, we perform ablation experiments over a number of facets of BERT in order to better understand their relative importance. Additional

Tasks	Dev Set				
	MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
$\text{BERT}_{\text{BASE}}$	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

Table 5: Ablation over the pre-training tasks using the $\text{BERT}_{\text{BASE}}$ architecture. "No NSP" is trained without the next sentence prediction task. "LTR & No NSP" is trained as a left-to-right LM without the next sentence prediction, like OpenAI GPT. "+ BiLSTM" adds a randomly initialized BiLSTM on top of the "LTR + No NSP" model during fine-tuning.

ablation studies can be found in Appendix C.

5.1 Effect of Pre-training Tasks

We demonstrate the importance of the deep bidirectionality of BERT by evaluating two pre-training objectives using exactly the same pre-training data, fine-tuning scheme, and hyperparameters as $\text{BERT}_{\text{BASE}}$:

No NSP: A bidirectional model which is trained using the "masked LM" (MLM) but without the "next sentence prediction" (NSP) task.

LTR & No NSP: A left-context-only model which is trained using a standard Left-to-Right (LTR) LM, rather than an MLM. The left-only constraint was also applied at fine-tuning, because removing it introduced a pre-train/fine-tune mismatch that degraded downstream performance. Additionally, this model was pre-trained without the NSP task. This is directly comparable to OpenAI GPT, but using our larger training dataset, our input representation, and our fine-tuning scheme.

We first examine the impact brought by the NSP task. In Table 5, we show that removing NSP hurts performance significantly on QNLI, MNLI, and SQuAD 1.1. Next, we evaluate the impact of training bidirectional representations by comparing "No NSP" to "LTR & No NSP". The LTR model performs worse than the MLM model on all tasks, with large drops on MRPC and SQuAD.

For SQuAD it is intuitively clear that a LTR model will perform poorly at token predictions, since the token-level hidden states have no right-side context. In order to make a good faith attempt at strengthening the LTR system, we added a randomly initialized BiLSTM on top. This does significantly improve results on SQuAD, but the results are still far worse than those of the pre-trained bidirectional models. The BiLSTM hurts performance on the GLUE tasks.

We recognize that it would also be possible to train separate LTR and RTL models and represent each token as the concatenation of the two models, as ELMo does. However: (a) this is twice as expensive as a single bidirectional model; (b) this is non-intuitive for tasks like QA, since the RTL model would not be able to condition the answer on the question; (c) this it is strictly less powerful than a deep bidirectional model, since it can use both left and right context at every layer.

5.2 Effect of Model Size

In this section, we explore the effect of model size on fine-tuning task accuracy. We trained a number of BERT models with a differing number of layers, hidden units, and attention heads, while otherwise using the same hyperparameters and training procedure as described previously.

Results on selected GLUE tasks are shown in Table 6. In this table, we report the average Dev Set accuracy from 5 random restarts of fine-tuning. We can see that larger models lead to a strict accuracy improvement across all four datasets, even for MRPC which only has 3,600 labeled training examples, and is substantially different from the pre-training tasks. It is also perhaps surprising that we are able to achieve such significant improvements on top of models which are already quite large relative to the existing literature. For example, the largest Transformer explored in Vaswani et al. (2017) is (L=6, H=1024, A=16) with 100M parameters for the encoder, and the largest Transformer we have found in the literature is (L=64, H=512, A=2) with 235M parameters (Al-Rfou et al., 2018). By contrast, $\text{BERT}_{\text{BASE}}$ contains 110M parameters and $\text{BERT}_{\text{LARGE}}$ contains 340M parameters.

It has long been known that increasing the model size will lead to continual improvements on large-scale tasks such as machine translation and language modeling, which is demonstrated by the LM perplexity of held-out training data shown in Table 6. However, we believe that this is the first work to demonstrate convincingly that scaling to extreme model sizes also leads to large improvements on very small scale tasks, provided that the model has been sufficiently pre-trained. Peters et al. (2018b) presented

mixed results on the downstream task impact of increasing the pre-trained bi-LM size from two to four layers and Melamud et al. (2016) mentioned in passing that increasing hidden dimension size from 200 to 600 helped, but increasing further to 1,000 did not bring further improvements. Both of these prior works used a feature-based approach — we hypothesize that when the model is fine-tuned directly on the downstream tasks and uses only a very small number of randomly initialized additional parameters, the task-specific models can benefit from the larger, more expressive pre-trained representations even when downstream task data is very small.

5.3 Feature-based Approach with BERT

All of the BERT results presented so far have used the fine-tuning approach, where a simple classification layer is added to the pre-trained model, and all parameters are jointly fine-tuned on a downstream task. However, the feature-based approach, where fixed features are extracted from the pre-trained model, has certain advantages. First, not all tasks can be easily represented by a Transformer encoder architecture, and therefore require a task-specific model architecture to be added. Second, there are major computational benefits to pre-compute an expensive representation of the training data once and then run many experiments with cheaper models on top of this representation.

In this section, we compare the two approaches by applying BERT to the CoNLL-2003 Named Entity Recognition (NER) task (Tjong Kim Sang and De Meulder, 2003). In the input to BERT, we use a case-preserving WordPiece model, and we include the maximal document context provided by the data. Following standard practice, we formulate this as a tagging task but do not use a CRF layer in the output. We use the representation of the first sub-token as the input to the token-level classifier over the NER label set.

Hyperparams				Dev Set Accuracy		
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

Table 6: Ablation over BERT model size. #L = the number of layers; #H = hidden size; #A = number of attention heads. "LM (ppl)" is the masked LM perplexity of held-out training data.

System	Dev F1	Test F1
ELMo (Peters et al., 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbik et al., 2018)	-	**93.1**
Fine-tuning approach		
$\text{BERT}_{\text{LARGE}}$	96.6	92.8
$\text{BERT}_{\text{BASE}}$	96.4	92.4
Feature-based approach ($\text{BERT}_{\text{BASE}}$)		
Embeddings	91.0	-
Second-to-Last Hidden	95.6	-
Last Hidden	94.9	-
Weighted Sum Last Four Hidden	95.9	-
Concat Last Four Hidden	96.1	-
Weighted Sum All 12 Layers	95.5	-

Table 7: CoNLL-2003 Named Entity Recognition results. Hyperparameters were selected using the Dev set. The reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.

To ablate the fine-tuning approach, we apply the feature-based approach by extracting the activations from one or more layers *without* fine-tuning any parameters of BERT. These contextual embeddings are used as input to a randomly initialized two-layer 768-dimensional BiLSTM before the classification layer.

Results are presented in Table 7. $\text{BERT}_{\text{LARGE}}$ performs competitively with state-of-the-art methods. The best performing method concatenates the token representations from the top four hidden layers of the pre-trained Transformer, which is only 0.3 F1 behind fine-tuning the entire model. This demonstrates that BERT is effective for both fine-tuning and feature-based approaches.

6 Conclusion

Recent empirical improvements due to transfer learning with language models have demonstrated that rich, unsupervised pre-training is an integral part of many language understanding systems. In particular, these results enable even low-resource tasks to benefit from deep unidirectional architectures. Our major contribution is further generalizing these findings to deep *bidirectional* architectures, allowing the same pre-trained model to successfully tackle a broad set of NLP tasks.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. 2018. Character-level language modeling with deeper self-attention. *arXiv preprint arXiv:1808.04444*.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853.

Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. 2009. The fifth PASCAL recognizing textual entailment challenge. In *TAC*. NIST.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*. Association for Computational Linguistics.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Z. Chen, H. Zhang, X. Zhang, and L. Zhao. 2018. Quora question pairs.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *ACL*.

Kevin Clark, Minh-Thang Luong, Christopher D Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. Maskgan: Better text generation via filling in the_. *arXiv preprint arXiv:1801.07736*.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*. Association for Computational Linguistics.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018. Reinforced mnemonic reader for machine reading comprehension. In *IJCAI*.

Yacine Jernite, Samuel R. Bowman, and David Sontag. 2017. Discourse-based objectives for fast unsupervised sentence representation learning. *CoRR*, abs/1705.00557.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The winograd schema challenge. In *Aaai spring symposium: Logical formalizations of commonsense reasoning*, volume 46, page 47.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *International Conference on Learning Representations*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS*.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *CoNLL*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1081–1088. Curran Associates, Inc.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *NAACL*.

Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, OpenAI.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Fu Sun, Linyang Li, Xipeng Qiu, and Yang Liu. 2018. U-net: Machine reading comprehension with unanswerable questions. *arXiv preprint arXiv:1810.06638*.

Wilson L Taylor. 1953. Cloze procedure: A new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018a. Glue: A multi-task benchmark and analysis platform

for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Wei Wang, Ming Yan, and Chen Wu. 2018b. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. QANet: Combining local convolution with global self-attention for reading comprehension. In *ICLR*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Appendix for "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"

We organize the appendix into three sections:

- Additional implementation details for BERT are presented in Appendix A;
- Additional details for our experiments are presented in Appendix B; and
- Additional ablation studies are presented in Appendix C.

  We present additional ablation studies for BERT including:

  – Effect of Number of Training Steps; and
  – Ablation for Different Masking Procedures.

A Additional Details for BERT

A.1 Illustration of the Pre-training Tasks

We provide examples of the pre-training tasks in the following.

Masked LM and the Masking Procedure Assuming the unlabeled sentence is `my dog is hairy`, and during the random masking procedure we chose the 4-th token (which corresponding to `hairy`), our masking procedure can be further illustrated by

- 80% of the time: Replace the word with the `[MASK]` token, e.g., `my dog is hairy` → `my dog is [MASK]`
- 10% of the time: Replace the word with a random word, e.g., `my dog is hairy` → `my dog is apple`
- 10% of the time: Keep the word unchanged, e.g., `my dog is hairy` → `my dog is hairy`. The purpose of this is to bias the representation towards the actual observed word.

The advantage of this procedure is that the Transformer encoder does not know which words it will be asked to predict or which have been replaced by random words, so it is forced to keep a distributional contextual representation of *every* input token. Additionally, because random replacement only occurs for 1.5% of all tokens (i.e., 10% of 15%), this does not seem to harm the model's language understanding capability. In Section C.2, we evaluate the impact this procedure.

Compared to standard langauge model training, the masked LM only make predictions on 15% of tokens in each batch, which suggests that more pre-training steps may be required for the model

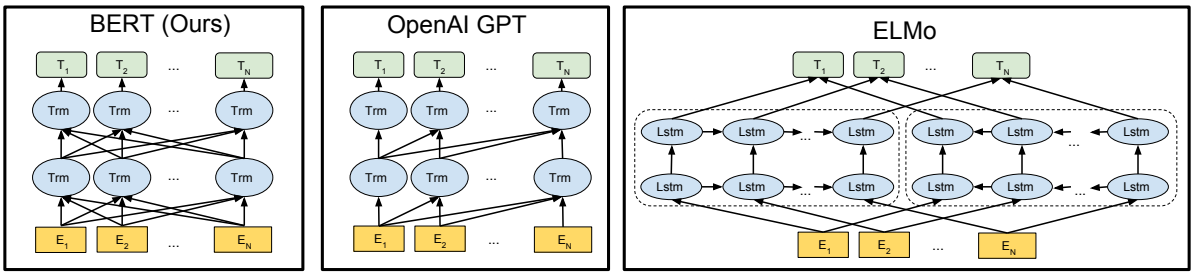


Figure 3: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTMs to generate features for downstream tasks. Among the three, only BERT representations are jointly conditioned on both left and right context in all layers. In addition to the architecture differences, BERT and OpenAI GPT are fine-tuning approaches, while ELMo is a feature-based approach.

to converge. In Section C.1 we demonstrate that MLM does converge marginally slower than a left-to-right model (which predicts every token), but the empirical improvements of the MLM model far outweigh the increased training cost.

Next Sentence Prediction The next sentence prediction task can be illustrated in the following examples.

Input = `[CLS] the man went to [MASK] store [SEP]`
`he bought a gallon [MASK] milk [SEP]`
Label = `IsNext`

Input = `[CLS] the man [MASK] to the store [SEP]`
`penguin [MASK] are flight ##less birds [SEP]`
Label = `NotNext`

A.2 Pre-training Procedure

To generate each training input sequence, we sample two spans of text from the corpus, which we refer to as "sentences" even though they are typically much longer than single sentences (but can be shorter also). The first sentence receives the `A` embedding and the second receives the `B` embedding. 50% of the time `B` is the actual next sentence that follows `A` and 50% of the time it is a random sentence, which is done for the "next sentence prediction" task. They are sampled such that the combined length is $\leq$ 512 tokens. The LM masking is applied after WordPiece tokenization with a uniform masking rate of 15%, and no special consideration given to partial word pieces.

We train with batch size of 256 sequences (256 sequences * 512 tokens = 128,000 tokens/batch) for 1,000,000 steps, which is approximately 40 epochs over the 3.3 billion word corpus. We use Adam with learning rate of 1e-4, $\beta_1 = 0.9$, $\beta_2 = 0.999$, L2 weight decay of 0.01, learning rate warmup over the first 10,000 steps, and linear decay of the learning rate. We use a dropout probability of 0.1 on all layers. We use a `gelu` activation (Hendrycks and Gimpel, 2016) rather than the standard `relu`, following OpenAI GPT. The training loss is the sum of the mean masked LM likelihood and the mean next sentence prediction likelihood.

Training of $\text{BERT}_{\text{BASE}}$ was performed on 4 Cloud TPUs in Pod configuration (16 TPU chips total).[13] Training of $\text{BERT}_{\text{LARGE}}$ was performed on 16 Cloud TPUs (64 TPU chips total). Each pre-training took 4 days to complete.

Longer sequences are disproportionately expensive because attention is quadratic to the sequence length. To speed up pretraing in our experiments, we pre-train the model with sequence length of 128 for 90% of the steps. Then, we train the rest 10% of the steps of sequence of 512 to learn the positional embeddings.

A.3 Fine-tuning Procedure

For fine-tuning, most model hyperparameters are the same as in pre-training, with the exception of the batch size, learning rate, and number of training epochs. The dropout probability was always kept at 0.1. The optimal hyperparameter values are task-specific, but we found the following range of possible values to work well across all tasks:

- **Batch size**: 16, 32

[13] https://cloudplatform.googleblog.com/2018/06/Cloud-TPU-now-offers-preemptible-pricing-and-global-availability.html

- **Learning rate (Adam)**: 5e-5, 3e-5, 2e-5
- **Number of epochs**: 2, 3, 4

We also observed that large data sets (e.g., 100k+ labeled training examples) were far less sensitive to hyperparameter choice than small data sets. Fine-tuning is typically very fast, so it is reasonable to simply run an exhaustive search over the above parameters and choose the model that performs best on the development set.

A.4 Comparison of BERT, ELMo ,and OpenAI GPT

Here we studies the differences in recent popular representation learning models including ELMo, OpenAI GPT and BERT. The comparisons between the model architectures are shown visually in Figure 3. Note that in addition to the architecture differences, BERT and OpenAI GPT are fine-tuning approaches, while ELMo is a feature-based approach.

The most comparable existing pre-training method to BERT is OpenAI GPT, which trains a left-to-right Transformer LM on a large text corpus. In fact, many of the design decisions in BERT were intentionally made to make it as close to GPT as possible so that the two methods could be minimally compared. The core argument of this work is that the bi-directionality and the two pre-training tasks presented in Section 3.1 account for the majority of the empirical improvements, but we do note that there are several other differences between how BERT and GPT were trained:

- GPT is trained on the BooksCorpus (800M words); BERT is trained on the BooksCorpus (800M words) and Wikipedia (2,500M words).
- GPT uses a sentence separator (`[SEP]`) and classifier token (`[CLS]`) which are only introduced at fine-tuning time; BERT learns `[SEP]`, `[CLS]` and sentence `A`/`B` embeddings during pre-training.
- GPT was trained for 1M steps with a batch size of 32,000 words; BERT was trained for 1M steps with a batch size of 128,000 words.
- GPT used the same learning rate of 5e-5 for all fine-tuning experiments; BERT chooses a task-specific fine-tuning learning rate which performs the best on the development set.

To isolate the effect of these differences, we perform ablation experiments in Section 5.1 which demonstrate that the majority of the improvements are in fact coming from the two pre-training tasks and the bidirectionality they enable.

A.5 Illustrations of Fine-tuning on Different Tasks

The illustration of fine-tuning BERT on different tasks can be seen in Figure 4. Our task-specific models are formed by incorporating BERT with one additional output layer, so a minimal number of parameters need to be learned from scratch. Among the tasks, (a) and (b) are sequence-level tasks while (c) and (d) are token-level tasks. In the figure, E represents the input embedding, T_i represents the contextual representation of token i, [CLS] is the special symbol for classification output, and [SEP] is the special symbol to separate non-consecutive token sequences.

B Detailed Experimental Setup

B.1 Detailed Descriptions for the GLUE Benchmark Experiments.

Our GLUE results in Table1 are obtained from `https://gluebenchmark.com/leaderboard` and `https://blog.openai.com/language-unsupervised`. The GLUE benchmark includes the following datasets, the descriptions of which were originally summarized in Wang et al. (2018a):

MNLI Multi-Genre Natural Language Inference is a large-scale, crowdsourced entailment classification task (Williams et al., 2018). Given a pair of sentences, the goal is to predict whether the second sentence is an *entailment*, *contradiction*, or *neutral* with respect to the first one.

QQP Quora Question Pairs is a binary classification task where the goal is to determine if two questions asked on Quora are semantically equivalent (Chen et al., 2018).

QNLI Question Natural Language Inference is a version of the Stanford Question Answering Dataset (Rajpurkar et al., 2016) which has been converted to a binary classification task (Wang et al., 2018a). The positive examples are (question, sentence) pairs which do contain the correct answer, and the negative examples are (question, sentence) from the same paragraph which do not contain the answer.

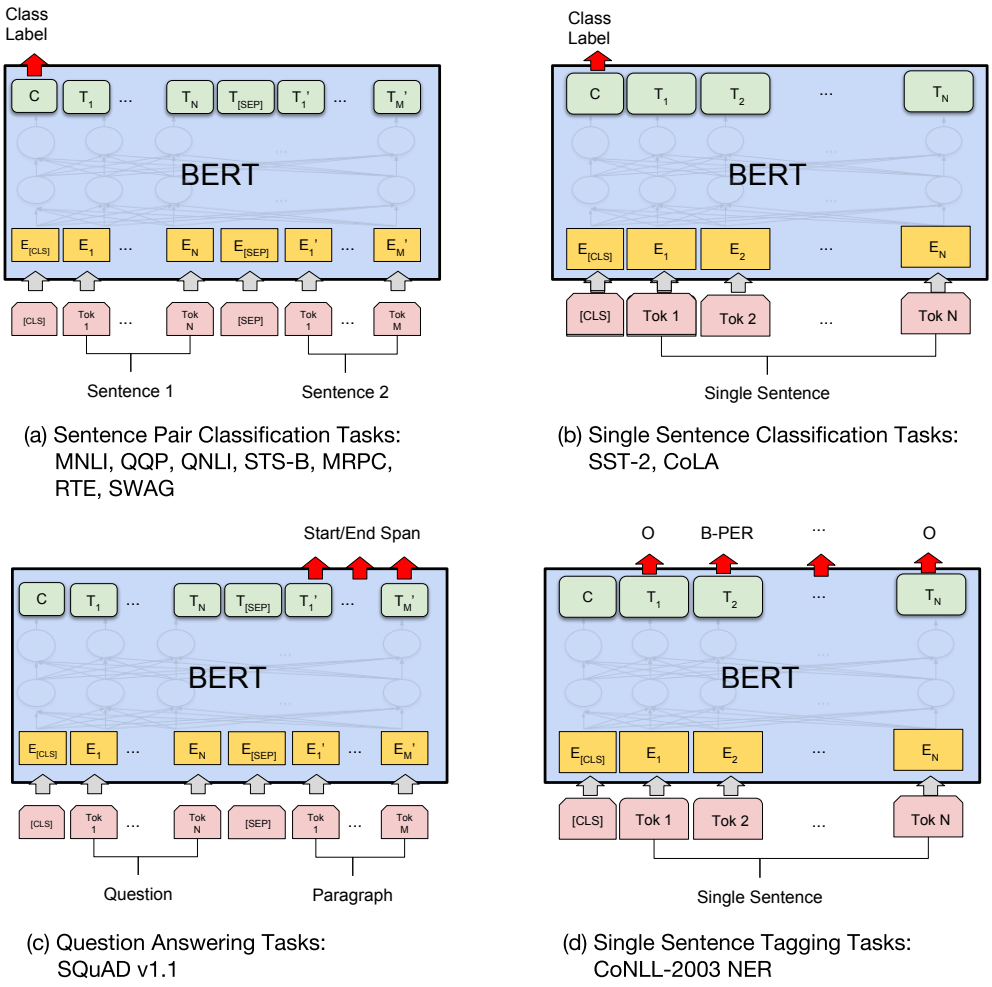


Figure 4: Illustrations of Fine-tuning BERT on Different Tasks.

SST-2 The Stanford Sentiment Treebank is a binary single-sentence classification task consisting of sentences extracted from movie reviews with human annotations of their sentiment (Socher et al., 2013).

CoLA The Corpus of Linguistic Acceptability is a binary single-sentence classification task, where the goal is to predict whether an English sentence is linguistically "acceptable" or not (Warstadt et al., 2018).

STS-B The Semantic Textual Similarity Benchmark is a collection of sentence pairs drawn from news headlines and other sources (Cer et al., 2017). They were annotated with a score from 1 to 5 denoting how similar the two sentences are in terms of semantic meaning.

MRPC Microsoft Research Paraphrase Corpus consists of sentence pairs automatically extracted from online news sources, with human annotations for whether the sentences in the pair are semantically equivalent (Dolan and Brockett, 2005).

RTE Recognizing Textual Entailment is a binary entailment task similar to MNLI, but with much less training data (Bentivogli et al., 2009).[14]

WNLI Winograd NLI is a small natural language inference dataset (Levesque et al., 2011). The GLUE webpage notes that there are issues with the construction of this dataset, [15] and every trained system that's been submitted to GLUE has performed worse than the 65.1 baseline accuracy of predicting the majority class. We therefore exclude this set to be fair to OpenAI GPT. For our GLUE submission, we always predicted the ma-

[14]Note that we only report single-task fine-tuning results in this paper. A multitask fine-tuning approach could potentially push the performance even further. For example, we did observe substantial improvements on RTE from multi-task training with MNLI.

[15]https://gluebenchmark.com/faq

jority class.

C Additional Ablation Studies

C.1 Effect of Number of Training Steps

Figure 5 presents MNLI Dev accuracy after fine-tuning from a checkpoint that has been pre-trained for k steps. This allows us to answer the following questions:

1. Question: Does BERT really need such a large amount of pre-training (128,000 words/batch * 1,000,000 steps) to achieve high fine-tuning accuracy?
   Answer: Yes, $\text{BERT}_{\text{BASE}}$ achieves almost 1.0% additional accuracy on MNLI when trained on 1M steps compared to 500k steps.

2. Question: Does MLM pre-training converge slower than LTR pre-training, since only 15% of words are predicted in each batch rather than every word?
   Answer: The MLM model does converge slightly slower than the LTR model. However, in terms of absolute accuracy the MLM model begins to outperform the LTR model almost immediately.

C.2 Ablation for Different Masking Procedures

In Section 3.1, we mention that BERT uses a mixed strategy for masking the target tokens when pre-training with the masked language model (MLM) objective. The following is an ablation study to evaluate the effect of different masking strategies.

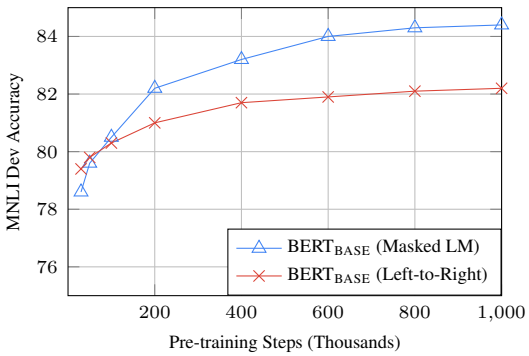


Figure 5: Ablation over number of training steps. This shows the MNLI accuracy after fine-tuning, starting from model parameters that have been pre-trained for k steps. The x-axis is the value of k.

Note that the purpose of the masking strategies is to reduce the mismatch between pre-training and fine-tuning, as the `[MASK]` symbol never appears during the fine-tuning stage. We report the Dev results for both MNLI and NER. For NER, we report both fine-tuning and feature-based approaches, as we expect the mismatch will be amplified for the feature-based approach as the model will not have the chance to adjust the representations.

Masking Rates			Dev Set Results		
MASK	SAME	RND	MNLI	NER	
			Fine-tune	Fine-tune	Feature-based
80%	10%	10%	84.2	95.4	94.9
100%	0%	0%	84.3	94.9	94.0
80%	0%	20%	84.1	95.2	94.6
80%	20%	0%	84.4	95.2	94.7
0%	20%	80%	83.7	94.8	94.6
0%	0%	100%	83.6	94.9	94.6

Table 8: Ablation over different masking strategies.

The results are presented in Table 8. In the table, MASK means that we replace the target token with the `[MASK]` symbol for MLM; SAME means that we keep the target token as is; RND means that we replace the target token with another random token.

The numbers in the left part of the table represent the probabilities of the specific strategies used during MLM pre-training (BERT uses 80%, 10%, 10%). The right part of the paper represents the Dev set results. For the feature-based approach, we concatenate the last 4 layers of BERT as the features, which was shown to be the best approach in Section 5.3.

From the table it can be seen that fine-tuning is surprisingly robust to different masking strategies. However, as expected, using only the MASK strategy was problematic when applying the feature-based approach to NER. Interestingly, using only the RND strategy performs much worse than our strategy as well.

A Hierarchical Multi-task Approach for Learning Embeddings from Semantic Tasks

Victor Sanh[1], Thomas Wolf[1], Sebastian Ruder[2,3]
[1]Hugging Face, 20 Jay Street, Brooklyn, New York, United States
[2]Insight Research Centre, National University of Ireland, Galway, Ireland
[3]Aylien Ltd., 2 Harmony Court, Harmony Row, Dublin, Ireland
{victor, thomas}@huggingface.co, sebastian@ruder.io

Abstract

Much effort has been devoted to evaluate whether multi-task learning can be leveraged to learn rich representations that can be used in various Natural Language Processing (NLP) down-stream applications. However, there is still a lack of understanding of the settings in which multi-task learning has a significant effect. In this work, we introduce a hierarchical model trained in a multi-task learning setup on a set of carefully selected semantic tasks. The model is trained in a hierarchical fashion to introduce an inductive bias by supervising a set of low level tasks at the bottom layers of the model and more complex tasks at the top layers of the model. This model achieves state-of-the-art results on a number of tasks, namely Named Entity Recognition, Entity Mention Detection and Relation Extraction without hand-engineered features or external NLP tools like syntactic parsers. The hierarchical training supervision induces a set of shared semantic representations at lower layers of the model. We show that as we move from the bottom to the top layers of the model, the hidden states of the layers tend to represent more complex semantic information.

Introduction

Recent Natural Language Processing (NLP) models heavily rely on rich distributed representations (typically word or sentence embeddings) to achieve good performance. One example are so-called "universal representations" (Conneau et al. 2017) which are expected to encode a varied set of linguistic features, transferable to many NLP tasks. This kind of rich word or sentence embeddings can be learned by leveraging the training signal from different tasks in a multi-task setting. It is known that a model trained in a multi-task framework can take advantage of inductive transfer between the tasks, achieving a better generalization performance (Caruana 1993). Recent works in sentence embeddings (Subramanian et al. 2018; Jernite, Bowman, and Sontag 2017) indicate that complementary aspects of the sentence (e.g. syntax, sentence length, word order) should be encoded in order for the model to produce sentence embeddings that are able to generalize over a wide range of tasks. Complementary aspects in representations can be naturally encoded by training a model on a set of diverse tasks, such as, machine translation, sentiment classification or natural language inference. Although, (i) the selection of this diverse set of tasks, as well as, (ii) the control of the interactions between them are of great importance, a deeper understanding of (i) and (ii) is missing as highlighted in the literature (Caruana 1997; Mitchell 1980; Ruder 2017). This work explores this line of research by combining, in a single model, four fundamental semantic NLP tasks: Named Entity Recognition, Entity Mention Detection (also sometimes referred as mention detection), Coreference Resolution and Relation Extraction. This selection of tasks is motivated by the inter-dependencies these tasks share. In Table 1, we give three simple examples to exemplify the reasons why these tasks should benefit from each other. For instance, in the last example knowing that *the company* and *Dell* are referring to the same real world entity, *Dell* is more likely to be an organization than a person.

Table 1: A few examples motivating our selection of tasks. Abbreviations: CR: Coreference Resolution, RE: Relation Extraction, EMD: Entity Mention Detection, NER: Named Entity Recognition, X $\rightsquigarrow$ Y: X is more likely to be Y.

Example	Predictions on one task...	...can help disambiguate other tasks
X works for Y	RE: {*work, X, Y*}	*X* $\rightsquigarrow$ Person (EMD) *Y* $\rightsquigarrow$ Organization or Person (NER)
I love Melbourne. I've lived three years in this city.	CR: (*Melbourne, this city*) RE: {*live, I, this city*}	*Melbourne* $\rightsquigarrow$ Location (EMD/NER)
Dell announced a $500M net loss. The company is near bankruptcy.	CR: (*Dell, The company*)	*Dell* $\rightsquigarrow$ Organization (EMD/NER)

Several prior works (Yang, Salakhutdinov, and Cohen 2016; Bingel and Søgaard 2017) avoid the question of the linguistic hierarchies between NLP tasks. We argue that some tasks (so-called "low level" tasks) are simple and require a limited amount of modification to the input of the model while other tasks (so-called "higher level" tasks) require a deeper processing of the inputs and likely a more complex architecture. Following (Hashimoto et al. 2017; Søgaard and Goldberg 2016), we therefore introduce a hierarchy between the tasks so that low level tasks are supervised at lower levels of the architecture while keeping more com-

plex interactions at deeper layers. Unlike previous works (Li and Ji 2014; Miwa and Bansal 2016), our whole model can be trained end-to-end without any external linguistic tools or hand-engineered features while giving stronger results on both Relation Extraction and Entity Mention Detection.

Our main contributions are the following: (1) we propose a multi-task architecture combining four different tasks that have not been explored together to the best of our knowledge. This architecture uses neural networks and does not involve external linguistic tools or hand-engineered features. We also propose a new sampling strategy for multi-task learning, *proportional sampling*. (2) We show that this architecture can lead to state-of-the art results on several tasks e.g. Named Entity Recognition, Relation Extraction and Entity Mention Detection while using simple models for each of these tasks. This suggests that the information encoded in the embeddings is rich and covers a variety of linguistic phenomena. (3) We study and give insights on the influence of multi-task learning on (i) the speed of training and (ii) the type of biases learned in the hierarchical model.

Model

In this section, we describe our model beginning at the lower levels of the architecture and ascending to the top layers. Our model introduces a hierarchical inductive bias between the tasks by supervising low-level tasks (that are assumed to require less knowledge and language understanding) at the bottom layers of the model architecture and supervising higher-level tasks at higher layers. The architecture of the model is shown in Figure 1. Following (Hashimoto et al. 2017), we use shortcut connections so that top layers can have access to bottom layer representations.

Words embeddings

Our model encodes words w_t of an input sentence $s = (w_1, w_2, ..., w_n)$ as a combination of three different types of embeddings. We denote the concatenation of the these three embeddings g_e.

Pre-trained word embeddings: We use GloVe (Pennington, Socher, and Manning 2014) pre-trained word level embeddings. These embeddings are fine-tuned during training.

Pre-trained contextual word embeddings: We also use contextualized ELMo embeddings (Peters et al. 2018). These word embeddings differ from GloVe word embeddings in that each token is represented by a vector that is a function of the whole sentence (a word can thus have different representations depending on the sentence it is extracted from). These representations are given by the hidden states of a bi-directional language model. ELMo embeddings have been shown to give state-of-the-art results in multiple NLP tasks (Peters et al. 2018).

Character-level word embeddings: Following (Chiu and Nichols 2015; Lample et al. 2016), we use character-level word embeddings to extract character-level features. Specifically, we use a convolutional neural network (CNN) (followed by a max pooling layer) for the ease of training since Recurrent Neural Network-based encodings do not significantly outperform CNNs while being computationally more expensive to train (Ling et al. 2015).

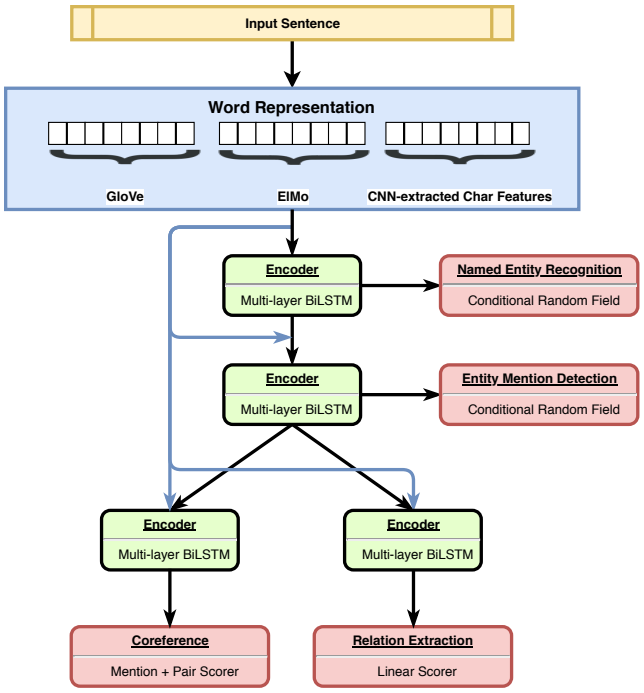


Figure 1: Diagram of the model architecture

Named Entity Recognition (NER)

The first layers of our model are supervised by Named Entity Recognition labels. NER aims to identify mentions of named entities in a sequence and classify them into predefined categories. In accordance with previous work (Chiu and Nichols 2015; Lample et al. 2016) the tagging module contains an RNN-based encoding layer followed by a sequence tagging module based on a conditional random field (Lafferty, McCallum, and Pereira 2001). We use multi-layer bi-directional LSTMs (Long Short-Term Memory) as encoders. The encoders take as input the concatenated word embeddings g_e and produce (sequence) embeddings g_{ner}. Specifically, g_{ner} are the concatentation of the backward and forward hidden states of the top layer of the biLSTMs, which are then fed to the sequence tagging layer.

We adopt the BILOU (Beginning, Inside, Last, Outside, Unit) tagging scheme. The tagging decisions are modeled using a CRF, which explicitly reasons about interactions between neighbour tokens tags.

Entity Mention Detection (EMD)

A second group of layers of our model are supervised using Entity Mention Detection labels. EMD is similar in spirit to NER but more general as it aims to identify all the mentions related to a real life entity, whereas NER only focuses on the named entities. Let us consider an example: $[\textit{The men}]_{PERS}$ *held on* $[\textit{the sinking vessel}]_{VEH}$ *until* $[\textit{the passenger ship}]_{VEH}$ *was able to reach them from* $[\textit{Corsica}]_{GPE}$. Here, NER annotations would only tag *Corsica*, while EMD requires a deeper understanding of the entities in the sentence.

We formulate Mention Detection as a sequence tagging task using a BILOU scheme. We use a multi-layer biLSTM followed by a CRF tagging layer. We adopt shortcut connections so that each layer can build on top of the representations extracted by the lower layers in a hierarchical fashion. The encoder thus takes as input the concatenation of the lower layer representations $[g_e, g_{ner}]$ and outputs sequence embeddings denoted by g_{emd}.

To be able to compare our results with previous works (Bekoulis et al. 2018; Miwa and Bansal 2016; Katiyar and Cardie 2017) on EMD, we identify the head of the entity mention rather than the whole mention.

Coreference Resolution (CR)

Ascending one layer higher in our model, CR is the task of identifying mentions that are referring to the same real life entity and cluster them together (typically at the level of a few sentences). For instance, in the example *My mom tasted the cake. She said it was delicious*, there are two clusters: (My mom, She) and (the cake, it). CR is thus a task which requires a form of semantic representation to cluster the mentions pointing to the same entity.

We use the model proposed in (Lee et al. 2017). This model considers all the spans in a document as potential mentions and learns to distinguish the candidate coreferent mentions from the other spans using a mention scorer to prune the number of mentions. The output of the mention scorer is fed to a mention pair scorer, which decides whether identified candidates mentions are coreferent. The main elements introduced in (Lee et al. 2017) are the use of span embeddings to combine context-dependent boundary representation and an attention mechanism over the span to point to the mention's head. This model is trained fully end-to-end without relying on external parser pre-processing.

The encoder takes as input the concatenation of $[g_e, g_{emd}]$ and outputs representations denoted as g_{cr}, which are then fed to the mention pair scorer.

Relation Extraction (RE)

The last supervision of our model is given by Relation Extraction (RE). RE aims at identifying semantic relational structures between entity mentions in unstructured text. Traditional systems treat this task as two pipelined tasks: (i) identifying mentions and (ii) classifying the relations between identified mentions. We use the Joint Resolution Model proposed by Bekoulis et al. (2018) in which the selection of the mentions and classification of the relation between these mentions are performed jointly. Following previous work (Li and Ji 2014; Katiyar and Cardie 2017; Bekoulis et al. 2018), we only consider relations between the last token of the head mentions involved in the relation. Redundant relations are therefore not classified.

The RE encoder is a multi-layer BiLSTM which takes as input $[g_e, g_{emd}]$ and outputs a representation denoted g_{re}. These contextualized representations are fed to a feedforward neural network. More specifically, considering two token's contextualized representations, g_i and g_j, both of size $\mathbb{R}^l$, we compute a vector score:

$$t(w_i, w_j) = V\phi(Ug_j + Wg_i + b) \quad (1)$$

where $U \in \mathbb{R}^{d\times l}$, $W \in \mathbb{R}^{d\times l}$, $b \in \mathbb{R}^d$, and $V \in \mathbb{R}^{r\times d}$ are learned transformation weights, l is the size of the embeddings output by the encoder, d is the size of the hidden layer of the feedforward network, r is the number of possible relations, and ϕ is a non-linear activation function. The relation probabilities are then estimated as $p = \sigma(t(w_i, w_j)) \in \mathbb{R}^r$ where p_k $(1 \leq k \leq r)$ is the probability that token w_i and token w_j are respectively labeled as *ARG1* and *ARG2* in a relation of type k. The model predictions are computed by thresholding estimated probabilities. The parameters of the model V, U, W, and b are trained by minimizing a cross-entropy loss.

In this formulation, a mention may be involved in several relations at the same time (for instance being the *ARG1* and the *ARG2* in two respective relations), which can occur in real life examples. If we replaced the *sigmoid* function by a *softmax* function, this is not possible.

In the model, the CR and RE modules are both on the same level. We did not find it helpful to introduce a hierarchical relation between these two tasks as they both rely on deeper semantic modeling, i.e. both trying to link mentions.

Experiment setting

Datasets and evaluation metrics

We use labeled data from different sources to train and evaluate our model. For NER, we use the English portion of OntoNotes 5.0 (Pradhan et al. 2013). Following Strubell et al. (2017), we use the same data split as used for coreference resolution in the CoNLL-2012 shared task (Pradhan et al. 2012). We report the performance on NER using span level F_1 score on the test set. The dataset covers a large set of document types (including telephone conversations, web text, broadcast news and translated documents), and a diverse set of 18 entity types (including PERSON, NORP, FACILITY, ORGANIZATION, GPE). Statistics of the corpus are detailed in Table 2. We also report performance on more commonly used CoNLL2003 NER dataset.

For CR, EMD and RE, we use the Automatic Content Extraction (ACE) program ACE05 corpus (Doddington et al. 2004). The ACE05 corpus is one of the largest corpus annotated with CR, EMD and RE making it a compelling dataset for multi-task learning. Mention tags in ACE05 cover 7 types of entities such as Person, Organization, or Geographical Entities. For each entity, both the mention boundaries and the head spans are annotated. ACE05 also introduces 6 relation types (including Organization-Affiliation (ORG-AFF), GEN-Affiliation (GEN-AFF), and Part-Whole (PART-WHOLE)). We use the same data splits as previous work (Li and Ji 2014; Miwa and Bansal 2016; Katiyar and Cardie 2017) for both RE and EMD and report F_1-scores, Precision, and Recall. We consider *an entity mention* correct if the model correctly predicted both the mention's head and its type. We consider *a relation* correct if the model correctly predicted the heads of the two arguments and the relation type.

Table 2: Data statistics

OntoNotes	Train	Dev	Test
Documents	2,483	319	322
Sentences	59,924	8,529	8,262
Named Entities	81,828	11,066	11,257
Tokens	1,088,503	147,724	152,728
ACE05	Train	Dev	Test
Documents	351	80	80
Sentences	7,273	1,765	1,535
Mentions	26,470	6,421	1,535
Relations	4,779	1,179	1,147

For CR, we use different splits to be able to compare to previous work (Bansal and Klein 2012; Durrett and Klein 2014). These splits (introduced in (Rahman and Ng 2009)) use the whole ACE05 dataset leaving 117 documents for test while having 482 documents for training (as in (Bansal and Klein 2012), we randomly split the training into a 70/30 ratio to form a validation set). We evaluate coreference on both splits. We compare all coreference systems using the commonly used metrics: MUC, B3, CEAFe (CEAFϕ_4) as well as the average F_1 of the three metrics as computed by the official CoNLL-2012 scorer. Note that Durrett and Klein make use of external NLP tools including an automatic parser (Durrett and Klein 2013).

We compare our model to several previous systems that have driven substantial improvements over the past few years both using graphical models or neural-net-based models. These are the strongest baselines to the best of our knowledge.

Training Details

Subramanian et al. (2018) observe that there is no clear consensus on how to correctly train a multi-task model. Specifically, there remain many open questions such as "when should the training schedule switch from one task to another task?" or "should each task be weighted equally?" One of the main issues that arises when training a multi-task model is *catastrophic forgetting* (French 1999) where the model abruptly forgets part of the knowledge related to a previously learned task as a new task is learned. This phenomenon is especially present when multiple tasks are trained sequentially.

We selected the simple yet effective training method described in (Søgaard and Goldberg 2016; Ruder et al. 2017): after each parameter update, a task is randomly selected and a batch of the dataset attached to this task is also sampled at random to train the model. This process is repeated until convergence (the validation metrics do not improve anymore). We tested both uniform and proportional sampling and found that proportional sampling performs better both in terms of performance and speed of convergence. In proportional sampling, the probability of sampling a task is proportional to the relative size of each dataset compared to the cumulative size of all the datasets. Note that unlike (Subramanian et al. 2018), the updates for a particular task affect the layers associated with this task and all the layers below but not the layers above.

Results and Discussion

Overall Performance

In this section, we present our main results on each task and dataset. The hierarchical model and multi-task learning framework presented in this work achieved state-of-the-art results on three tasks, namely NER (+0.52), EMD (+3.8) and RE (+6.8). Table 3 summarizes the results and introduces each setups' abbreviation (alphabetical letters). In the following subsections, we highlight a few useful observations.

To be able to compare our work on CR with the various baselines, we report results using different settings and splits. More precisely, *GM* indicates that gold mentions were used for evaluation and that coreference was trained using the ACE05 splits introduced in (Rahman and Ng 2009).

Using gold mentions is impossible in real settings so we also relax this condition leading to a more challenging task in which we make no use of external tools or metadata (such as speaker ID used by some systems (Clark and Manning 2015)). Comparing setups A and A-GM shows how the supervision from one module (e.g. CR) can flow through the entire architecture and impact other tasks' performance: RE's F_1 score drops by $\sim$1 point on A. Note that the GM setup impacts the training exit condition (the validation metrics stop improving) and the evaluation metrics (it is well known that using gold mentions at evaluation time improves CR's performance). Similarly, the A-GM setup leads to the state-of-the-art on EMD and RE. It increases the F_1 by $\sim$1.5 points for EMD and $\sim$1 point for RE (A vs. A-GM). This suggests that having different type of information on different sentences brings richer information than having multiple types of information on the same sentences (setup A-CoNLL2012 -see Table 4- supports this claim as CR trained on another dataset leads to comparable performance on the three other tasks).

To analyze which components of the model are driving the improvements and understand the interactions between the different tasks and modules, we performed an ablation study summarized in the following paragraphs and on Tables 3 and 5.

Single Task vs. Full Models The largest difference between a single-task and a multi-task setup is observed on the RE task (A vs. D on Table 3), while the results on NER are similar in multi-task and single-task setups (B vs. A & A-GM). This further highlights how the RE module can be sensitive to information learned from other tasks. Results on EMD are in the middle, with the single task setup giving higher score than a multi-task-setup except for A-GM and I. More surprisingly, CR can give slightly better results when being single-task-trained (A vs. E).

Progressively adding tasks To better understand the contribution of each module, we vary the number of tasks in our training setup. The experiments show that training using RE helps both for NER and for EMD. Adding RE supervision leads to an increase of $\sim$1 point on NER while boosting both precision and recall on EMD (F vs. I). CR and RE can help NER as shown by comparing setups A and F: recall and F_1

Table 3: Results: Baselines and ablation study on the tasks. *GM* means that the same coreference module uses gold mentions at evaluation time and that we used the splits introduced in (Rahman and Ng 2009). Otherwise, we use for coreference the same splits as for EMD and RE (351/80/80). For coreference, figures that are comparable with (Durrett and Klein 2014) are tagged with an *.

		NER			EMD			RE			CR			
Setup	Model	P	R	F_1	P	R	F_1	P	R	F_1	MUC	B3	Ceafe	Avg. F_1
	(Strubell et al. 2017)	-	-	86.99	-	-	-	-	-	-	-	-	-	-
	(Katiyar and Cardie 2017)	-	-	-	84.0	81.3	82.6	57.9	54.0	55.9	-	-	-	-
	(Miwa and Bansal 2016)	-	-	-	82.9	83.9	83.4	57.2	54.0	55.6	-	-	-	-
	(Li and Ji 2014)	-	-	-	85.2	76.9	80.8	68.9	41.9	52.1	-	-	-	-
	(Durrett and Klein 2014)	-	-	-	-	-	-	-	-	-	81.03*	74.89*	72.56*	76.16*
	(Bansal and Klein 2012)	-	-	-	-	-	-	-	-	-	70.2*	72.5*	-	-
(A)	Full Model	87.52	87.21	87.36	85.68	85.69	85.69	68.53	54.48	61.30	73.89	61.34	59.11	64.78
(A-GM)	Full Model - GM	87.12	87.09	87.10	87.15	87.33	**87.24**	70.40	56.40	**62.69**	82.49*	67.64*	60.75*	**70.29***
(B)	NER	87.24	87.00	87.12	-	-	-	-	-	-	-	-	-	-
(C)	EMD	-	-	-	87.03	85.27	86.14	-	-	-	-	-	-	-
(D)	RE	-	-	-	-	-	-	60.47	52.14	55.99	-	-	-	-
(E)	CR	-	-	-	-	-	-	-	-	-	74.80	62.63	59.59	65.67
(E-GM)	CR - GM	-	-	-	-	-	-	-	-	-	81.78*	66.42*	59.93*	69.38*
(F)	NER + EMD	87.50	86.32	86.91	86.54	85.49	86.02	-	-	-	-	-	-	-
(G)	EMD + RE	-	-	-	85.58	85.38	85.50	68.66	54.05	60.49	-	-	-	-
(H)	EMD + CR	-	-	-	85.84	85.46	85.65	-	-	-	72.67	59.05	57.33	63.02
(I)	NER + EMD + RE	87.37	87.65	**87.51**	86.63	85.90	86.26	65.57	55.62	60.18	-	-	-	-
(J)	NER + EMD + CR	87.67	87.34	87.50	85.89	85.86	85.87	-	-	-	75.73	62.92	61.24	**66.64**
(K)	EMD + NER	85.67	87.19	86.43	85.62	84.76	85.19	-	-	-	-	-	-	-
(L)	EMD + NER + RE + CR	85.78	86.66	86.21	85.24	85.05	85.14	63.32	55.54	59.17	73.29	60.37	58.86	64.17

for NER are $\sim$1 point stronger, while the impact on EMD is negative. Finally, training using CR supervision boosts NER (F vs. J) by increasing NER's recall while lowering EMD's precision and F_1. In other words the information flowing along the hierarchical model (e.g. stacking of encoders and shortcut connections) enables higher levels' supervision to train lower levels to learn better representations. More generally, whenever the task RE is combined with another task, it always increases the F_1 score (most of the improvement coming from the precision) by 2–6 F_1 points.

Experimenting with the hierarchy order Comparing setups F vs. K and A vs. L in which we switched NER and EMD, provides evidence for the hierarchical relation between NER and EMD: supervising EMD at a lower level than NER is detrimental to the overall performance. This supports our intuition that the hierarchy should follow the intrinsic difficulty of the tasks.

Comparison to other canonical datasets We also compare our model on two other canonical datasets for NER (CoNLL-2003) and CR (CoNLL-2012). Details are reported in Table 4. We did not tune hyperparameters, keeping the same hyperparameters as used in the previous experiments. We reach performance comparable to previous work and the other tasks, demonstrating that our improvements are not dataset-dependent.

Effect of the embeddings We perform an ablation study on the words and character embeddings g_e. Results are reported in Table 5. As expected, contextualized ELMo embeddings have a noticeable effect on each metrics. Removing ELMo leads to a $\sim$4 F_1 points drop on each task. Furthermore, character-level embeddings, which are sensitive to morphological features such as prefixes and suffixes and capitalization, also have a strong impact, in particular on NER, RE and CR. Removing character-level embeddings does not affect EMD suggesting that the EMD module can compensate for this information. The main improvements on the CR task stem from the increase in B3 and Ceafe metrics. Note that the strong effect of removing a type of embedding is also a consequence of using shortcut connections: removing an embedding has a direct impact on the input to each task's module.

Table 4: Comparison to other canonical datasets on NER (CoNLL-2003) and coreference (CoNLL-2012). A-CoNLL: train A-RS-GM using CoNLL-2003 for NER; A-CoNLL-2012: train A using CoNLL-2012 for coreference.

Model	NER (F_1)	CR (F_1)
Lample et al. (2016)	90.94	-
Strubell et al. (2017)	90.54	-
Peters et al. (2018)	92.22	-
(A-CoNLL-2003)	91.63	70.14
Durrett and Klein (2014)	-	61.71
Lee et al. (2017) (single)	-	67.2
Lee et al. (2017) (ensemble)	-	68.8
(A-CoNLL-2012)	86.90	62.48

Table 5: Ablation study on the embeddings. We remove one by one the embeddings on the first layer of the best performing model (A-RS-GM).

Model	NER (F_1)	EMD (F_1)	RE (F_1)	CR (F_1)
Glove + Char. embds + ELMo	87.10	87.24	62.69	70.29
Glove + Char. embds	84.33	83.13	57.47	66.44
Glove	79.81	83.00	53.77	64.26

What did the embeddings and encoders learn?

High scores on a specific task suggest that the representations learned by the encoders (and the embeddings) have somehow managed to capture relevant linguistic or statistical features for the task. However, using complex archi-

tectures makes it difficult to understand what is actually encoded in the embeddings and hidden states and what type of linguistic information, if any, is being used by the model to make a particular prediction. To further understand our architecture, we analyze the inductive biases encoded in the embeddings and hidden states of the various layers. We follow Conneau et al. (2018) who introduced 10 different probing tasks[1] to analyze the quality of sentence embeddings. These tasks aim at evaluating a wide set of linguistic properties from surface information, to syntactic information through semantic information.

We use a simple logistic regression classifier, which takes the sentence embeddings as inputs and predicts the linguistic property. We study both the word embeddings (g_e) and the hidden state representations (biLSTM encoders) specific to each module in our model. The sentence embedding of an input sequence of length L is computed from the L hidden states of an encoder by taking the maximum value over each dimension of the last layer activations as done in (Conneau et al. 2017). Sentence embeddings are obtained from word and character-level embeddings by max-pooling over a sentence's words. Averaging word embeddings is known to be a strong baseline for sentence embeddings (Arora, Liang, and Ma 2017) and we also report the results of this simpler procedure in Table 6.

Results We compare our results with two baselines from (Conneau et al. 2018): bag-of-words computed from FastText embeddings and SkipThought sentence embeddings (Kiros et al. 2015). We compare the base word embeddings g_e of our model with the first baseline and the output of the task-specific encoders to the second baseline. A first observation is that the word embeddings already encode rich representations, having an accuracy higher than 70% on seven of the ten probing tasks. We suspect that shortcut connections are key to this good performances by allowing high level tasks to encode rich representations. The good performance on *Bigram Shift* (compared to BoV-FastText: +38.8) likely comes from the use of ELMo embeddings which are sensitive to word order. The same argument may also explain the strong performance on *Sentence Length*.

There are significant discrepancies between the results of the word embeddings g_e and the encoder representations, indicating that the learned linguistic features are different between these two types of embeddings. Averaging the base embeddings surpasses encoder embeddings on almost all the probing tasks (except *Coordination Inversion*). The difference is particularly high on the *Word Content* task in which the results of the encoders embeddings barely rise above 11.0, indicating that the ability to recover a specific word is not a useful feature for our four semantic tasks.

The performance of the encoder representation is stronger on semantic probing tasks, compared to the low signal for surface and syntatic tasks. The only exception is the *Sentence Length* which suggest that this linguistic aspect is naturally encoded. The performances of the NER and EMD encoders are generally in the same range supporting the fact that these two tasks are similar in nature. Finally, we observe that the highest scores for encoder representations always stem from the coreference encoder suggesting that CR is both the highest level task and that the CR module requires rich and diverse representations to make a decision.

[1]A probing task is a classification task that focuses on a well identified linguistic property.

Multi-task learning accelerating training

It is also interesting to understand the influence of a multi-task learning framework on the training time of the model. In the following section, we compare the speed of training in terms of number of parameter updates (a parameter update being equal to a back-propagation pass) for each of the tasks in the multi-task framework against a single-task framework. The speed of training is defined as the number of updates necessary to reach convergence based on the validation metric.

Results are presented in Table 7 for the best performing multi-task model (A-GM). The multi-task framework needs less updates to reach comparable (or higher) F_1 score in most cases except on the RE task. This supports the intuition that knowledge gathered from one task is beneficial to the other tasks in the hierarchical architecture of our model.

Related work

Our work is most related to Hashimoto et al. (2017) who develop a joint hierarchical model trained on syntactic and semantic tasks. The top layers of this model are supervised by semantic relatedness and textual entailment between two input sentences, implying that the lower layers need to be run two times on different input sentences. Our choice of tasks avoids such a setup. Our work also adopts a different approach to multi-task training (proportional sampling) therefore avoiding the use of complex regularization schemes to prevent catastrophic forgetting. Our results show that strong performances can be reached without these ingredients. In addition, the tasks examined in this work are more focused on learning semantic representations, thereby reducing the need to learn surface and syntactic information, as evidenced by the linguistic probing tasks.

Unlike (Hashimoto et al. 2017) and other previous work (Katiyar and Cardie 2017; Bekoulis et al. 2018; Augenstein, Ruder, and Søgaard 2018), we do not learn label embeddings, meaning that the (supervised) output/prediction of a layer is not directly fed to the following layer through an embedding learned during training. Nonetheless, sharing embeddings and stacking hierarchical encoders allows us to share the supervision from each task along the full structure of our model and achieve state-of-the-art performance.

Unlike some studies on multi-task learning such as (Subramanian et al. 2018), each task has its own contextualized encoder (multi-layer BiLSTM) rather than a shared one, which we found to improve the performance.

Our work is also related to Søgaard and Goldberg (2016) who propose to cast a cascade architecture into a multi-task learning framework. However, this work was focused on syntactic tasks and concluded that adding a semantic task like NER to a set of syntactic tasks does not bring any improvement. This confirms the intuition that multi-task learn-

Table 6: SentEval Probing task accuracies. Classification is performed using a simple logistic regression enabling fair evaluation of the richness of a sentence embedding. We report two baselines from Conneau et al..

Tasks	Surface Information		Syntatic Information			Semantic Information				
	SentLen	WC	TreeDepth	TopConst	BShift	Tense	SubjNum	ObjNum	SOMO	CoordInv
Word Embeddings										
Bov-fastText ((Conneau et al. 2018))	54.8	91.6	32.3	63.1	50.8	87.8	81.9	79.3	50.3	52.7
Our model (g_{emb}) - Max	62.4	43.0	32.5	76.3	74.5	88.1	85.7	82.7	54.7	56.9
Our model (g_{emb}) - Average	72.1	70.0	38.5	79.9	81.4	89.7	88.5	86.5	57.4	63.0
BiLSTM-max encoders										
SkipThought (Conneau et al.)	59.6	35.7	42.7	70.5	73.4	90.1	83.3	79.0	70.3	70.1
Our model (Encoder NER g_{ner})	50.7	3.24	19.5	34.2	57.2	66.6	63.5	61.6	50.7	52.0
Our model (Encoder EMD g_{emd})	43.3	1.8	19.3	30.0	56.3	64.0	60.1	57.9	51.3	50.4
Our model (Encoder RE g_{re})	56.8	1.2	19.3	24.5	53.9	62.3	60.8	57.1	50.4	52.2
Our model (Encoder CR g_{cr})	61.9	11.0	29.5	55.9	70.0	82.8	83.0	76.5	53.3	58.7

Table 7: Speed of training: Difference in number of updates necessary before convergence: Multi-task (Full Model: A-GM) compared to single task. We report the differences in F_1 performance. Lower time is better, higher performance is better.

Setup	Model	Time Δ	Performance Δ
(B)	NER	-16%	-0.02
(C)	EMD	-44%	+1.14
(D)	RE	+78%	+6.76
(E-GM)	Coref-GM	-28%	+0.91

ing is mostly effective when tasks are related and that syntactic and semantic tasks may be too distant to take advantage of shared representations. The authors used a linear classifier with a softmax activation, relying on the richness on the embeddings. As a consequence the sequence tagging decisions are made independently for each token, in contrast to our work.

One central question in multi-task learning is the training procedure. Several schemes have been proposed in the literature. Hashimoto et al. (2017) train their hierarchical model following the model's architecture from bottom to top: the trainer successively goes through the whole dataset for each task before moving to the task of the following level. The underlying hypothesis is that the model should perform well on low-level tasks before being trained in more complicated tasks. Hashimoto et al. avoid catastrophic forgetting by introducing *successive regularization* using slack constraints on the parameters. Subramanian et al. (2018) adopt a simpler strategy for each parameter update: a task is randomly selected and a batch of the associated dataset is sampled for the current update. More recently, Mccann et al. (2018) explored various batch-level sampling strategies and showed that an anti-curriculum learning strategy (Bengio et al. 2009) is most effective. In contrast, we propose a novel proportional sampling strategy, which we find to be more effective.

Regarding the selection of the set of tasks, our work is closest to (Durrett and Klein 2014; Singh et al. 2013). Durrett and Klein (2014) combine coreference resolution, entity linking (sometimes referred to as *Wikification*) and mention detection. Singh et al. (2013) combine entity tagging, coreference resolution and relation extraction. These two works are based on graphical models with hand-engineered factors. We are using a neural-net-based approach fully trainable in an end-to-end fashion, with no need for external NLP tools (such as in (Durrett and Klein 2014)) or hand-engineered features. Coreference resolution is rarely used in combination with other tasks. The main work we are aware of is (Dhingra et al. 2018), which uses coreference clusters to improve reading comprehension and the works on language modeling by Ji et al. (2017) and Yang et al. (2016).

Regarding the combination of entity mention detection and relation, we refer to our baselines detailed above. Here again, our predictors do not require additional features like dependency trees (Miwa and Bansal 2016) or hand-engineered heuristics (Li and Ji 2014).

Conclusion

We proposed a hierarchically supervised multi-task learning model focused on a set of semantic task. This model achieved state-of-the-art results on the tasks of Named Entity Recognition, Entity Mention Detection and Relation Extraction and competitive results on Coreference Resolution while using simpler training and regularization procedures than previous works. The tasks share common embeddings and encoders allowing an easy information flow from the lowest level to the top of the architecture. We analyzed several aspects of the representations learned by this model as well as the effect of each tasks on the overall performances of the model.

Acknowledgments

We thank Gianis Bekoulis and Victor Quach for helpful feedbacks and the anonymous reviewers for constructive comments on the paper.

References

Arora, S.; Liang, Y.; and Ma, T. 2017. A simple but tough-to-beat baseline for sentence embeddings.

Augenstein, I.; Ruder, S.; and Søgaard, A. 2018. Multi-task Learning of Pairwise Sequence Classification Tasks Over Disparate Label Spaces. In *Proceedings of NAACL-HLT 2018*.

Bansal, M., and Klein, D. 2012. Coreference semantics from web features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, 389–398. Stroudsburg, PA, USA: Association for Computational Linguistics.

Bekoulis, G.; Deleu, J.; Demeester, T.; and Develder, C. 2018. Joint entity recognition and relation extraction as a multi-head selection problem.

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, 41–48. New York, NY, USA: ACM.

Bingel, J., and Søgaard, A. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. *CoRR* abs/1702.08303.

Caruana, R. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*.

Caruana, R. 1997. Multitask learning. *Mach. Learn.* 28(1):41–75.

Chiu, J. P. C., and Nichols, E. 2015. Named Entity Recognition with Bidirectional LSTM-CNNs.

Clark, K., and Manning, C. D. 2015. Entity-centric coreference resolution with model stacking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 1405–1415. Association for Computational Linguistics.

Conneau, A.; Kiela, D.; Schwenk, H.; Barrault, L.; and Bordes, A. 2017. Supervised learning of universal sentence representations from natural language inference data. *CoRR* abs/1705.02364.

Conneau, A.; Kruszewski, G.; Lample, G.; Barrault, L.; and Baroni, M. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *CoRR* abs/1805.01070.

Dhingra, B.; Jin, Q.; Yang, Z.; Cohen, W. W.; and Salakhutdinov, R. 2018. Neural models for reasoning over multiple mentions using coreference. *CoRR* abs/1804.05922.

Doddington, G.; Mitchell, A.; Przybocki, M.; Ramshaw, L.; Strassel, S.; and Weischedel, R. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. 2.

Durrett, G., and Klein, D. 2013. Easy victories and uphill battles in coreference resolution. In *EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 1971–1982.

Durrett, G., and Klein, D. 2014. A joint model for entity analysis: Coreference, typing, and linking. In *Transactions of the Association for Computational Linguistics*, volume 2, 477–490.

French, R. M. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences* 3(4):128 – 135.

Hashimoto, K.; Xiong, C.; Tsuruoka, Y.; and Socher, R. 2017. A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Jernite, Y.; Bowman, S. R.; and Sontag, D. 2017. Discourse-based objectives for fast unsupervised sentence representation learning. *CoRR* abs/1705.00557.

Ji, Y.; Tan, C.; Martschat, S.; Choi, Y.; and Smith, N. A. 2017. Dynamic Entity Representations in Neural Language Models.

Katiyar, A., and Cardie, C. 2017. Going out on a limb: Joint Extraction of Entity Mentions and Relations without Dependency Trees. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 917–928.

Kiros, R.; Zhu, Y.; Salakhutdinov, R.; Zemel, R. S.; Torralba, A.; Urtasun, R.; and Fidler, S. 2015. Skip-thought vectors. *arXiv preprint arXiv:1506.06726*.

Lafferty, J. D.; McCallum, A.; and Pereira, F. C. N. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 282–289.

Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; and Dyer, C. 2016. Neural Architectures for Named Entity Recognition.

Lee, K.; He, L.; Lewis, M.; and Zettlemoyer, L. 2017. End-to-end Neural Coreference Resolution.

Li, Q., and Ji, H. 2014. Incremental Joint Extraction of Entity Mentions and Relations. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)* 402–412.

Ling, W.; Luís, T.; Marujo, L.; Astudillo, R. F.; Amir, S.; Dyer, C.; Black, A. W.; and Trancoso, I. 2015. Finding function in form: Compositional character models for open vocabulary word representation. *CoRR* abs/1508.02096.

Mccann, B.; Keskar, N. S.; Xiong, C.; and Socher, R. 2018. The Natural Language Decathlon : Multitask Learning as Question Answering. (Nips).

Mitchell, T. M. 1980. The need for biases in learning generalizations. Technical report.

Miwa, M., and Bansal, M. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1105–1116. Association for Computational Linguistics.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations.

Pradhan, S.; Moschitti, A.; Xue, N.; Uryupina, O.; and Zhang, Y. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *EMNLP-CoNLL Shared Task*.

Pradhan, S.; Moschitti, A.; Xue, N.; Tou Ng, H.; Bjrkelund, A.; Uryupina, O.; Zhang, Y.; and Zhong, Z. 2013. Towards robust linguistic analysis using ontonotes. 143–152.

Rahman, A., and Ng, V. 2009. Supervised models for coreference resolution.

Ruder, S.; Bingel, J.; Augenstein, I.; and Søgaard, A. 2017. Learning what to share between loosely related tasks. *arXiv preprint arXiv:1705.08142*.

Ruder, S. 2017. An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv* (June).

Singh, S.; Riedel, S.; Martin, B.; Zheng, J.; and McCallum, A. 2013. Joint Inference of Entities, Relations, and Coreference. *Automated Knowledge Base Construction (AKBC CIKM 2013)* 1–6.

Søgaard, A., and Goldberg, Y. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL*.

Strubell, E.; Verga, P.; Belanger, D.; and McCallum, A. 2017. Fast and Accurate Entity Recognition with Iterated Dilated Convolutions.

Subramanian, S.; Trischler, A.; Bengio, Y.; and Pal, C. J. 2018. Learning General Purpose Distributed Sentence Representations Via Large Scale Multi-Task Learning.

Yang, Z.; Blunsom, P.; Dyer, C.; and Ling, W. 2016. Reference-aware language models. *CoRR* abs/1611.01628.

Yang, Z.; Salakhutdinov, R.; and Cohen, W. 2016. Multi-Task Cross-Lingual Sequence Tagging from Scratch.

Sequence classification with human attention

Maria Barrett[1] **Joachim Bingel**[2]
Nora Hollenstein[3] **Marek Rei**[4] **Anders Søgaard**[2]
[1]Department of Nordic Studies and Linguistics, University of Copenhagen, Denmark
[2]Department of Computer Science, University of Copenhagen, Denmark
[3]Department of Computer Science, ETH Zurich, Switzerland
[4]Department of Computer Science and Technology, University of Cambridge, United Kingdom
barrett@hum.ku.dk {bingel, soegaard}@di.ku.dk
noraho@ethz.ch marek.rei@cl.cam.ac.uk

Abstract

Learning attention functions requires large volumes of data, but many NLP tasks simulate human behavior, and in this paper, we show that human attention really does provide a good inductive bias on many attention functions in NLP. Specifically, we use estimated human attention derived from eye-tracking corpora to regularize attention functions in recurrent neural networks. We show substantial improvements across a range of tasks, including sentiment analysis, grammatical error detection, and detection of abusive language.

1 Introduction

When humans read a text, they do not attend to *all* its words (Carpenter and Just, 1983; Rayner and Duffy, 1988). For example, humans are likely to omit many function words and other words that are predictable in context and focus on less predictable content words. Moreover, when they fixate on a word, the duration of that fixation depends on a number of linguistic factors (Clifton et al., 2007; Demberg and Keller, 2008).

Since learning good attention functions for recurrent neural networks requires large volumes of data (Zoph et al., 2016; Britz et al., 2017), and errors in attention are known to propagate to classification decisions (Alkhouli et al., 2016), we explore the idea of using human attention, as estimated from eye-tracking corpora, as an inductive bias on such attention functions. Penalizing attention functions for departing from human attention may enable us to learn better attention functions when data is limited.

Eye-trackers provide millisecond-accurate records on where humans look when they are reading, and they are becoming cheaper and more easily available by the day (San Agustin et al., 2009). In this paper, we use publicly available eye-tracking corpora, i.e., texts augmented with eye-tracking measures such as fixation duration times, and large eye-tracking corpora have appeared increasingly over the past years. Some studies suggest that the relevance of text can be inferred from the gaze pattern of the reader (Salojärvi et al., 2003) – even on word-level (Loboda et al., 2011).

Contributions We present a recurrent neural architecture with attention for sequence classification tasks. The architecture jointly learns its parameters and an attention function, but can alternate between supervision signals from labeled sequences (with no explicit supervision of the attention function) and from attention trajectories. This enables us to use per-word fixation durations from eye-tracking corpora to regularize attention functions for sequence classification tasks. We show such regularization leads to significant improvements across a range of tasks, including sentiment analysis, detection of abusive language, and grammatical error detection. Our implementation is made available at https://github.com/coastalcph/Sequence_classification_with_human_attention.

2 Method

We present a recurrent neural architecture that jointly learns the recurrent parameters and the attention function, but can alternate between supervision signals from labeled sequences and from attention trajectories in eye-tracking corpora. The input will be a set of labeled sequences (sentences paired with discrete category labels) and a set of sequences, in which each token is associated with a scalar value representing the attention human readers devoted to this token on average.

The two input datasets, i.e., the target task train-

Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL 2018), pages 302–312
Brussels, Belgium, October 31 - November 1, 2018.

ing data of sentences paired with discrete categories, and the eye-tracking corpus, need not (and will not in our experiments) overlap in any way. Our experimental protocol, in other words, does not require in-task eye-tracking recordings, but simply leverages information from existing, available corpora.

Behind our approach lies the simple observation that we can correlate the token-level attention devoted by a recurrent neural network, even if trained on sentence-level signals, with any measure defined at the token level. In other words, we can compare the attention devoted by a recurrent neural network to various measures, including token-level annotation (Rei and Søgaard, 2018) and eye-tracking measures. The latter is particularly interesting as it is typically considered a measurement of *human* attention.

We go beyond this: Not only can we compare machine attention with human attention, we can also constrain or inform machine attention by human attention in various ways. In this paper, we explore this idea, proposing a particular architecture and training method that, in effect, uses human attention to *regularize* machine attention.

Our training method is similar to a standard approach to training multi-task architectures (Dong et al., 2015; Søgaard and Goldberg, 2016; Bingel and Søgaard, 2017), sometimes referred to as the *alternating* training approach (Luong et al., 2016): We randomly select a data point from our training data or the eye-tracking corpus with some (potentially equal) probability. If the data point is sampled from our training data, we predict a discrete category and use the computed loss to update our parameters. If the data point is sampled from the eye-tracking corpus, we still run the recurrent network to produce a category, but this time we only monitor the attention weights assigned to the input tokens. We then compute the minimum squared error between the normalized eye-tracking measure and the normalized attention score. In other words, in multi-task learning, we optimize each task for a fixed number of parameter updates (or mini-batches) before switching to the next task (Dong et al., 2015); in our case, we optimize for a target task (for a fixed number of updates), then improve our attention function based on human attention (for a fixed number of updates), then return to optimizing for the target task and continue iterating.

2.1 Model

Our architecture is a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) that encodes word representations x_i into forward and backward representations, and into combined hidden states h_i (of slightly lower dimensionality) at every timestep. In fact, our model is a hierarchical model whose word representations are concatenations of the output of character-level LSTMs and word embeddings, following Plank et al. (2016), but we ignore the character-level part of our architecture in the equations below:

$$\overrightarrow{h_i} = LSTM(x_i, \overrightarrow{h_{i-1}}) \tag{1}$$

$$\overleftarrow{h_i} = LSTM(x_i, \overleftarrow{h_{i+1}}) \tag{2}$$

$$\widetilde{h_i} = [\overrightarrow{h_i}; \overleftarrow{h_i}] \tag{3}$$

$$h_i = \tanh(W_h \widetilde{h_i} + b_h) \tag{4}$$

The final (reduced) hidden state is sometimes used as a sentence representation s, but we instead use attention to compute s by multiplying dynamically predicted attention weights with the hidden states for each time step. The final sentence predictions y are then computed by passing s through two more hidden layers:

$$s = \sum_i \widetilde{a_i} h_i \tag{5}$$

$$y = \sigma(W_y \tanh(W_{\tilde{y}} s + b_{\tilde{y}}) + b_y) \tag{6}$$

From the hidden states, we directly predict token-level raw attention scores a_i:

$$e_i = \tanh(W_e h_i + b_e) \tag{7}$$

$$a_i = W_a e_i + b_a \tag{8}$$

We normalize these predictions to attention weights $\widetilde{a_i}$:

$$\widetilde{a_i} = \frac{a_i}{\sum_k a_k} \tag{9}$$

Our model thus combines two distinct objectives: one at the sentence level and one at the token level. The sentence-level objective is to minimize the squared error between output activations and true sentence labels $\widehat{y}$.

$$L_{sent} = \sum_j (y^{(j)} - \widehat{y}^{(j)})^2 \tag{10}$$

The token-level objective, similarly, is to minimize the squared error for the attention not aligning with our human attention metric.

$$L_{tok} = \sum_j \sum_t (a^{(j)(t)} - \widehat{a}^{(j)(t)})^2 \quad (11)$$

These are finally combined to a weighted sum, using λ (between 0 and 1) to trade off loss functions at the sentence and token levels.

$$L = L_{sent} + \lambda L_{tok} \quad (12)$$

Note again that our architecture does not require the target task data to come with eye-tracking information. We instead learn jointly to predict sentence categories and to attend to the tokens humans tend to focus on for longer. This requires a training schedule that determines when to optimize for the sentence-level classification objective, and when to optimize the machine attention at the token level. We therefore define an epoch to comprise a fixed number of batches, and sample every batch of training examples either from the target task data or from the eye-tracking corpus, as determined by a coin flip, the bias of which is tuned as a hyperparameter. Specifically, we define an epoch to consist of n batches, where n is the number of training sentences in the target task data divided by the batch size. This coin is potentially weighted with data being drawn from the auxiliary task with some probability or a decreasing probability of $\frac{1}{E+1}$, where E is the current epoch; see Section 4 for hyper-parameters.

3 Data

As mentioned in the above, our architecture requires no overlap between the eye-tracking corpus and the training data for the target task. We therefore rely on publicly available eye-tracking corpora. For sentiment analysis, grammatical error detection, and hate speech detection, we use publicly available research datasets that have been used previously in the literature. All datasets were lower-cased.

3.1 Eye-tracking corpora

For our experiments, we concatenate two publicly available eye-tracking corpora, the Dundee Corpus (Kennedy et al., 2003) and the reading parts of the ZuCo Corpus (Hollenstein et al., 2018), described below. Both corpora contain eye-tracking measurements from several subjects reading the same text. For every token, we compute the mean duration of all fixations to this token as our measure of human attention, following previous work (Barrett et al., 2016a; Gonzalez-Garduno and Søgaard, 2018).

Dundee The English part of the Dundee corpus (Kennedy et al., 2003) comprises 2,368 sentences and more than 50,000 tokens. The texts were read by ten skilled, adult, native speakers. The texts are 20 newspaper articles from *The Independent*. The reading was self-paced and as close to natural, contextualized reading as possible for a laboratory data collection. The apparatus was a Dr Bouis Oculometer Eyetracker with a 1000 Hz monocular (right) sampling. At most five lines were shown per screen while subjects were reading.

ZuCo The ZuCo corpus (Hollenstein et al., 2018) is a combined eye-tracking and EEG dataset. It contains approximately 1,000 individual English sentences read by 12 adult, native speakers. Eye movements were recorded with the infrared video-based eye tracker *EyeLink 1000 Plus* at a sampling rate of 500 Hz. The sentences were presented at the same position on the screen, one at a time. Longer sentences spanned multiple lines. The subjects used a control pad to switch to the next sentence and to answer the control questions, which allowed for natural reading speed. The corpus contains both natural reading and reading in a task-solving context. For compatibility with the Dundee corpus, we only use the subset of the data, where humans were encouraged to read more naturally. This subset contains 700 sentences. This part of the Zuco corpus contains positive, negative or neutral sentences from the Stanford Sentiment Treebank (Socher et al., 2013) for passive reading, to analyze the elicitation of emotions and opinions during reading. As a control condition, the subjects sometimes had to rate the quality of the described movies; in approximately 10% of the cases. The Zuco corpus also contains instances where subjects were presented with Wikipedia sentences that contained semantic relations such as *employer*, *award* and *job_title* (Culotta et al., 2006). The control condition for this tasks consisted of multiple-choice questions about the content of the previous sentence; again, approximately 10% of all sentences were followed by a question.

Task	Trains set		Dev. set		Test set	
	domain	n sent	domain	n sent	domain	n sent
Sentiment	SemEval Twitter	7,177	SemEval Twitter	1,205	SemEval Twitter	2,870
Sentiment					SemEval SMS	2,094
Grammatical error	FCE	28,731	FCE	2,222	FCE	2,720
Abusive language	Waseem (2016)	5,529	Waseem (2016)	690	Waseem (2016)	690
Abusive language	Waseem and Hovy (2016)	11,225	Waseem and Hovy (2016)	1,403	Waseem and Hovy (2016)	1,403

Table 1: Overview of the tasks and datasets used.

Preprocessing of eye-tracking data Mean fixation duration (Mean fix dur) is extracted from the Dundee Corpus. For Zuco, we divide total reading time per word token with the number of fixations to obtain mean fixation duration. The mean fixation duration is selected empirically among gaze duration (sum of all fixations in the first pass reading of the a word) and total fixation duration, and n fixations. Then we average these numbers for all readers of the corpus to get a more robust average processing time. Eye-tracking is known to correlate with word frequency (Rayner and Duffy, 1988). We include a frequency baseline on the eye tracking text, BNC inv freq. The word frequencies comes from the British National Corpus (BNC) frequency lists (Kilgarriff, 1995). We use log-transformed frequency per million. Before normalizing, we take the additive inverse of the frequency, such that rare words get a high value, making it comparable to gaze.

Mean fix dur and BNC inv freq are min-max-normalized to a value in the range 0-1. Mean fix dur is normalized separately for the two eye tracking corpora. We expect the experimental bias – especially the fact that ZuCo contains reading of isolated sentences and Dundee contains longer texts – to influence the reading and therefore separate normalization should preserve the signal within each corpus better.

3.2 Sentiment classification

Table 1 presents an overview of all train, development and test sets used in this paper.

Our first task is sentence-level sentiment classification. We note that many sentiment analysis datasets contain document-level labels or include more fine-grained annotation of text spans, say phrases or words. For compatibility with our other tasks, we focus on sentence-level sentiment analysis. We use the SemEval-2013 Twitter dataset (Wilson et al., 2013; Rosenthal et al., 2015) for training and development. For test, we use a same-domain test set, the SemEval-2013 Twitter test set (SemEval Twitter pos | neg), and an out-of-domain test set, SemEval-2013 SMS test set (SemEval SMS pos | neg). The SemEval-2013 sentiment classification task was a three-way classification task with positive, negative and neutral classes. We reduce the task to binary tasks detecting negative sentences vs. non-negative and vice versa for the positive class. Therefore the dataset size is the same for POS and NEG experiments.

3.3 Grammatical error detection

Our second task is grammatical error detection. We use the First Certificate in English error detection dataset (FCE) (Yannakoudakis et al., 2011). This dataset contains essays written by English learners during language examinations, where any grammatical errors have been manually annotated by experts. Rei and Yannakoudakis (2016) converted the dataset for a sequence labeling task and we use their splits for training, development and testing. Similarly to Rei and Søgaard (2018), we perform sentence-level binary classification of sentences that need some editing vs. grammatically correct sentences. We do not use the token-level labels for training our model.

3.4 Hate speech detection

Our third and final task is detection of abusive language; or more specifically, hate speech detection. We use the datasets of Waseem (2016) and Waseem and Hovy (2016). The former contains 6,909 tweets; the latter 14,031 tweets. They are manually annotated for sexism and racism. In this study, sexism and racism are conflated into one category in both datasets. Both datasets are split in train, development and test splits consisting of 80%, 10% and 10% of the tweets respectively.

4 Experiments

Models In our experiments, we compare three models: (a) a baseline model with automatically learned attention, (b) our model with an attention

	BL			BNC INV FREQ			MEAN FIX DUR		
TASK	P	R	F_1	P	R	F_1	P	R	F_1
SEMEVAL SMS NEG	43.55	45.41	43.77	45.82	48.65	45.24	47.15	46.98	**45.77**
SEMEVAL SMS POS	65.79	50.81	57.08	65.92	51.04	57.45	65.46	52.95	**58.50**
SEMEVAL TWITTER NEG	57.39	26.87	35.70	62.50	28.66	37.78	60.52	30.67	**40.23**
SEMEVAL TWITTER POS	77.96	53.88	63.63	79.66	54.66	64.78	78.77	55.35	**64.96**
FCE	79.01	89.33	83.84	79.18	89.26	83.89	79.03	90.28	**84.28**
WASEEM (2016)	76.42	62.07	68.29	77.20	61.71	68.54	77.20	63.06	**69.30**
WASEEM AND HOVY (2016)	76.23	72.23	74.16	76.33	74.70	75.48	76.95	74.43	**75.61**
MEAN	68.05	57.23	60.92	69.52	58.38	61.88	69.30	59.10	**62.67**

Table 2: Sentence classification results. P(recision), R(ecall) and F_1. Averages over 10 random seeds. The best average F_1 score per task is shown in bold.

function regularized by information about human attention, and finally, (c) a second baseline using frequency information as a proxy for human attention and using the same regularization scheme as in our human attention model.

Hyperparameters Basic hyper-parameters such as number of hidden layers, layer size, and activation functions were following the settings of Rei and Søgaard (2018). The dimensionality of our word embedding layer was set to size 300, and we use publicly available pre-trained Glove word embeddings (Pennington et al., 2014) that we fine-tune during training. The dimensionality of the character embedding layer was set to 100. The recurrent layers in the character-level component have dimensionality 100; the word-level recurrent layers dimensionality 300. The dimensionality of our feed-forward layer, leading to reduced combined representations h_i, is 200, and the attention layer has dimensionality 100.

Three hyper-parameters, however, we tune for each architecture and for each task, by measuring sentence-level F_1-scores on the development sets. These are: (a) learning rate, (b) λ in Equation (12), i.e., controlling the relative importance of the attention regularization, and (c) the probability of sampling data from the eye-tracking corpus during training.

For all tasks and all conditions (baseline, frequency-informed baseline, and our human attention model), we perform a grid search over learning rates [.01 .1 1.], L_{att} weight λ values [.2 .4 .6 .8 1.], and probability of sampling from the eye-tracking corpus [.125 .25 .5 1., decreasing] – where *decreasing* means that the probability of sampling from the eye-tracking corpus initially is 0.5, but drops linearly for each epoch ($\frac{1}{E+1}$; see 2.1. We apply the models with the best average F_1 scores over three random seeds on the validation data, to our test sets.

Initialization Our models are randomly initialized. This leads to some variance in performance across different runs. We therefore report averages over 10 runs in our experiments below.

5 Results

Our performance metric across all our experiments is the sentence-level F_1 score. We report precision, recall and F_1 scores for all tasks in Table 2.

Our main finding is that our human attention model, based on regularization from mean fixation durations in publicly available eye-tracking corpora, consistently outperforms the recurrent architecture with learned attention functions. The improvements over both baseline and BNC frequency are significant ($p < 0.01$) using bootstrapping (Calmettes et al., 2012) over all tasks, with one seed. The mean error reduction over the baseline is 4.5%.

Unsurprisingly, knowing that human attention helps guide our recurrent architecture, the frequency-informed baseline is also better than the non-informed baseline across the board, but the human attention model is still significantly better across all tasks ($p < 0.01$). For all tasks except negative sentiment, we note that generally, most of the improvements over the learned attention baseline for the gaze-informed models, are due to improvements in recall. Precision is not worse, but we do not see any larger improvements on preci-

sion either. For the negative SEMEVAL tasks, we also see larger improvements for precision.

The observation that improvements are primarily due to increased recall, aligns well with the hypothesis that human attention serves as an efficient regularization, preventing overfitting to surface statistical regularities that can lead the network to rely on features that are not there at test time (Globerson and Roweis, 2006), at the expense of target class precision.

6 Analysis

We illustrate the differences between our baseline models and the model with gaze-informed attention by the attention weights of an example sentence. Though it is a single, cherry-picked example, it is representative of the general trends we observe in the data, when manually inspecting attention patterns. Table 3 presents a coarse visualization of the attention weights of six different models, namely our baseline architecture and the architecture with gaze-informed attention, trained on three different tasks: hate speech detection, negative sentiment classification, and error detection. The sentence is a positive hate speech example from the Waseem and Hovy (2016) development set. The words with more attention than the sentence average are bold-faced.

First note that the baseline models only attend to one or two coherent text parts. This pattern was very consistent across all the sentences we examined. This pattern was not observed with gaze-informed attention.

Our second observation is that the baseline models are more likely to attend to stop words than gaze-informed attention. This suggests that gaze-informed attention has learned to simulate human attention to some degree. We also see many differences between the jointly learned task-specific, gaze-informed attention functions.

The gaze-informed hate speech classifier, for example, places considerable attention on *BUT*, which in this case is a passive-aggressive hate speech indicator. It also gives weight to *double standards* and *certain rules*.

The gaze-informed sentiment classifier, on the other hand, focuses more on *sorry I am not sexist* which, in isolation, reads like an apologetic disclaimer. This model also gives weight to *double standards* and *certain rules*

The gaze-informed grammatical error detection model gives attention to *standards*, which is ungrammatical, because of the morphological number disagreement with its determiner *a*; it also gives attention to *certain rules*, which is disagreeing, again in number, with *there's*. It also gives attention to the non-word *fem*.

Overall, this, in combination with our results in Table 3, suggests that the regularization effect from human attention enables our architecture to learn to better attend to the most relevant aspects of sentences for the target tasks. In other words, human attention provides the inductive bias that makes learning possible.

7 Discussion and related work

Gaze in NLP It has previously been shown that several NLP tasks benefit from gaze information, including part-of-speech tagging (Barrett and Søgaard, 2015b; Barrett et al., 2016a), prediction of multiword expressions (Rohanian et al., 2017) and sentiment analysis (Mishra et al., 2017b).

Gaze information and other measures from psycholinguistics have been used in different ways in NLP. Some authors have used discretized, single features (Pate and Goldwater, 2011, 2013; Plank, 2016; Klerke et al., 2016), whereas others have used multidimensional, continuous values (Barrett et al., 2016a; Bingel et al., 2016). We follow Gonzalez-Garduno and Søgaard (2018) in using a single, continuous feature. We did not experiment with other representations, however. Specifically, we only considered the signal from token-level, normalized mean fixation durations.

Fixation duration is a feature that carries an enormous amount of information about the text and the language understanding process. Carpenter and Just (1983) show that readers are more likely to fixate on open-class words that are not predictable from context, and Kliegl et al. (2004) show that a higher cognitive load results in longer fixation durations. Fixations before skipped words are shorter before short or high-frequency words and longer before long or low-frequency words in comparison with control fixations (Kliegl and Engbert, 2005). Many of these findings suggest correlations with syntactic information, and many authors have confirmed that gaze information is useful to discriminate between syntactic phenomena (Demberg and Keller, 2008; Barrett and Søgaard, 2015a,b).

Gaze data has also been used in the context of

FCE		SemEval Twitter NEG		Waseem and Hovy (2016)	
BL	MFD	BL	MFD	BL	MFD
@CharlesClassiqk:	**@CharlesClassiqk:**	**@CharlesClassiqk:**	**@CharlesClaqqqqqqqssiqk:**	@CharlesClassiqk:	@CharlesClassiqk:
sorry	**sorry**	**sorry**	**sorry**	sorry	sorry
I'm	I'm	**I'm**	**I'm**	I'm	I'm
not	not	**not**	**not**	not	not
sexist	**sexist**	**sexist**	**sexist**	sexist	sexist
BUT	BUT	BUT	BUT	BUT	**BUT**
there	**there**	there	there	there	**there**
is	is	is	is	is	is
a	a	a	a	a	a
double	**double**	double	**double**	double	**double**
standards	**standards**	standards	**standards**	**standards**	**standards**
there's	**there's**	there's	there's	**there's**	**there's**
certain	**certain**	certain	**certain**	**certain**	**certain**
rules	rules	rules	**rules**	**rules**	**rules**
for	for	for	for	**for**	**for**
dudes	dudes	dudes	dudes	**dudes**	**dudes**
and	and	and	and	**and**	and
there's	**there's**	there's	there's	**there's**	**there's**
certain	**certain**	certain	**certain**	**certain**	**certain**
rules	**rules**	rules	**rules**	**rules**	**rules**
for	for	for	for	**for**	for
femâĂę	**femâĂę**	femâĂę	femâĂę	**femâĂę**	femâĂę

Table 3: One sentence marked as containing sexism from Waseem and Hovy (2016) development set. Using trained baseline (BL) and gaze model (MFD) for three tasks: error detection, sentiment classification, and hate speech detection. Words with more attention than sentence average are boldfaced.

sentiment analysis before (Mishra et al., 2017b,a). Mishra et al. (2017b) augmented a sentiment analysis system with eye-tracking features, including first fixation durations and fixation counts. They show that fixations not only have an impact in detecting sentiment, but also improve sarcasm detection. They train a convolutional neural network that learns features from both gaze and text and uses them to classify the input text (Mishra et al., 2017a). On a related note, Raudonis et al. (2013) developed a emotion recognition system from visual stimulus (not text) and showed that features such as pupil size and motion speed are relevant to accurately detect emotions from eye-tracking data. Wang et al. (2017) use variables shown to correlate with human attention, e.g. surprisal, to guide the attention for sentence representations.

Gaze has also been used in the context of grammaticality (Klerke et al., 2015a,b), as well as in readability assessment (Gonzalez-Garduno and Søgaard, 2018).

Gaze has either been used as features (Barrett and Søgaard, 2015a; Barrett et al., 2016b) or as a direct supervision signal in multi-task learning scenarios (Klerke et al., 2016; Gonzalez-Garduno and Søgaard, 2018). We are, to the best of our knowledge, the first to use gaze to inform attention functions in recurrent neural networks.

Human-inspired attention functions Ibraheem et al. (2017), however, uses optimal attention to simulate human attention in an interactive machine translation scenario, and Britz et al. (2017) limit attention to a local context, inspired by findings in studies of human reading. Rei and Søgaard (2018) use auxiliary data to regularize attention functions in recurrent neural networks; not from psycholinguistics data, but using small amounts of task-specific, token-level annotations. While their motivation is very different from ours, technically our models are very related. In a different context, Das et al. (2017) investigated whether humans attend to the same regions as neural networks solving visual question answering problems. Lindsey (2017) also used human-inspired, unsupervised attention in a computer vision context.

Other work on multi-purpose attention functions While our work is the first to use gaze data to guide attention in a recurrent architectures, there has recently been some work on sharing attention functions across tasks. Firat et al. (2016), for example, share attention functions between languages in the context of multi-way neural machine translation.

Sentiment analysis While sentiment analysis is most often considered a supervised learning problem, several authors have leveraged other signals

than annotated data to learn sentiment analysis models that generalize better. Felbo et al. (2017), for example, use emoji prediction to pretrain their sentiment analysis models. Mishra et al. (2018) use several auxiliary tasks, including gaze prediction, for document-level sentiment analysis. There is a lot of previous work, also, leveraging information across different sentiment analysis datasets, e.g., Liu et al. (2016).

Error detection In grammatical error detection, Rei (2017) used an unsupervised auxiliary language modeling task, which is similar in spirit to our second baseline, using frequency information as auxiliary data. Rei and Yannakoudakis (2017) go beyond this and evaluate the usefulness of many auxiliary tasks, primarily syntactic ones. They also use frequency information as an auxiliary task.

Hate speech detection In hate speech detection, many signals beyond the text are often leveraged (see Schmidt and Wiegand (2017) for an overview of the literature). Interestingly, many authors have used signals from sentiment analysis, e.g., Gitari et al. (2015), motivated by the correlation between hate speech and negative sentiment. This correlation may also explain why we see the biggest improvements with gaze-informed attention on those two tasks.

Human inductive bias Finally, our work relates to other work on providing better inductive biases for learning human-related tasks by observing humans (Tamuz et al., 2011; Wilson et al., 2015). We believe this is a truly exciting line of research that can help us push research horizons in many ways.

8 Conclusion

We have shown that human attention provides a useful inductive bias on machine attention in recurrent neural networks for sequence classification problems. We present an architecture that enables us to leverage human attention signals from general, publicly available eye-tracking corpora, to induce better, more robust task-specific NLP models. We evaluate our architecture and show improvements across three NLP tasks, namely sentiment analysis, grammatical error detection, and detection of abusive language. We observe that not only does human attention help models distribute their attention in a generally useful way; human attention also seems to act like a regularizer providing more robust performance across domains, and it enables better learning of task-specific attention functions through joint learning.

References

Tamer Alkhouli, Gabriel Bretschner, Jan-Thorsten Peter, Mohammed Hethnawi, Andreas Guta, and Hermann Ney. 2016. Alignment-based neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 54–65.

Maria Barrett, Joachim Bingel, Frank Keller, and Anders Søgaard. 2016a. Weakly supervised part-of-speech tagging using eye-tracking data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 2, pages 579–584.

Maria Barrett, Frank Keller, and Anders Søgaard. 2016b. Cross-lingual transfer of correlations between parts of speech and gaze features. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 1330–1339.

Maria Barrett and Anders Søgaard. 2015a. Reading behavior predicts syntactic categories. In *Proceedings of the nineteenth conference on computational natural language learning (CoNLL)*, pages 345–249.

Maria Barrett and Anders Søgaard. 2015b. Using reading behavior to predict grammatical functions. In *Workshop on Cognitive Aspects of Computational Language Learning (CogACLL)*, pages 1–5.

Joachim Bingel, Maria Barrett, and Anders Søgaard. 2016. Extracting token-level signals of syntactic processing from fMRI-with an application to POS induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 747–755.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, volume 2, pages 164–169.

Denny Britz, Melody Y. Guan, and Minh-Thang Luong. 2017. Efficient attention using a fixed-size memory representation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 392–400.

Guillaume Calmettes, Gordon B Drummond, and Sarah L Vowler. 2012. Making do with what we have: use your bootstraps. *The Journal of physiology*, 590(15):3403–3406.

Patricia A Carpenter and Marcel Adam Just. 1983. What your eyes do while your mind is reading. *Eye movements in reading: Perceptual and language processes*, pages 275–307.

Charles Clifton, Adrian Staub, and Keith Rayner. 2007. Eye movements in reading words and sentences. In *Eye Movements: A Window on Mind and Brain*, pages 341–371. Elsevier, Amsterdam, The Netherlands.

Aron Culotta, Andrew McCallum, and Jonathan Betz. 2006. Integrating probabilistic extraction models and data mining to discover relations and patterns in text. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 296–303. Association for Computational Linguistics.

Abhishek Das, Harsh Agrawal, Lawrence Zitnick, Devi Parikh, and Dhruv Batra. 2017. Human attention in visual question answering: Do humans and deep networks look at the same regions? *Computer Vision and Image Understanding*, 163:90–100.

Vera Demberg and Frank Keller. 2008. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition*, 109(2):193–210.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1723–1732.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyan Rahwan, and Sune. Lehmann. 2017. Using millions of emoji occurrences to pretrain any-domain models for detecting emotion, sentiment, and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of 14th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 866–875.

Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long. 2015. A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering*, 10(4):215–230.

Amir Globerson and Sam Roweis. 2006. Nightmare at test time: robust learning by feature deletion. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 353–360.

Ana Gonzalez-Garduno and Anders Søgaard. 2018. Learning to predict readability using eye-movement data from natives and learners. In *Proceedings of the Thirty-Second Association for the Advancement of Artificial Intelligence Conference (AAAI)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Nora Hollenstein, Jonathan Rotsztejn, Marius Troendle, Andreas Pedroni, Ce Zhang, and Nicolas Langer. 2018. ZuCo: A simultaneous EEG and eye-tracking resource for natural sentence reading. *Scientific data*, *Under Review*.

Samee Ibraheem, Nicholas Altieri, and John DeNero. 2017. Learning an interactive attention policy for neural machine translation. In *MTSummit*.

Alan Kennedy, Robin Hill, and Joël Pynte. 2003. The dundee corpus. In *Proceedings of the 12th European conference on eye movement*.

Adam Kilgarriff. 1995. BNC database and word frequency lists. *Retrieved Dec. 2017*.

Sigrid Klerke, Héctor Martínez Alonso, and Anders Søgaard. 2015a. Looking hard: Eye tracking for detecting grammaticality of automatically compressed sentences. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, pages 97–105.

Sigrid Klerke, Sheila Castilho, Maria Barrett, and Anders Søgaard. 2015b. Reading metrics for estimating task efficiency with MT output. In *Proceedings of the Sixth Workshop on Cognitive Aspects of Computational Language Learning (CogACLL)*, pages 6–13.

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. In *Proceedings of 14th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 1528–1533.

Reinhold Kliegl and Ralf Engbert. 2005. Fixation durations before word skipping in reading. *Psychonomic Bulletin & Review*, 12(1):132–138.

Reinhold Kliegl, Ellen Grabner, Martin Rolfs, and Ralf Engbert. 2004. Length, frequency, and predictability effects of words on eye movements in reading. *European Journal of Cognitive Psychology*, 16(1-2):262–284.

Jack Lindsey. 2017. Pre-training attention mechanisms. In *NIPS Workshop on Cognitive Informed Artificial Intelligence*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Deep multi-task learning with shared memory. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 118–127.

Tomasz D Loboda, Peter Brusilovsky, and Jöerg Brunstein. 2011. Inferring word relevance from eye-movements of readers. In *Proceedings of the 16th international conference on intelligent user interfaces*, pages 175–184. ACM.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence-to-sequence learning. In *International Conference on Learning Representations (ICLR)*.

Abhijit Mishra, Kuntal Dey, and Pushpak Bhattacharyya. 2017a. Learning cognitive features from gaze data for sentiment and sarcasm classification using convolutional neural network. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 377–387.

Abhijit Mishra, Diptesh Kanojia, Seema Nagar, Kuntal Dey, and Pushpak Bhattacharyya. 2017b. Leveraging cognitive features for sentiment analysis. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, pages 156–166.

Abhijit Mishra, Srikanth Tamilselvam, Riddhiman Dasgupta, Seema Nagar, and Kuntal Dey. 2018. Cognition-cognizant sentiment analysis with multitask subjectivity summarization based on annotators' gaze behavior. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*.

John K Pate and Sharon Goldwater. 2011. Unsupervised syntactic chunking with acoustic cues: computational models for prosodic bootstrapping. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*, pages 20–29.

John K Pate and Sharon Goldwater. 2013. Unsupervised dependency parsing with acoustic cues. *Transactions of the Association for Computational Linguistics (TACL)*, 1:63–74.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Barbara Plank. 2016. Keystroke dynamics as signal for shallow syntactic parsing. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING)*, pages 609–618.

Barbara Plank, Yoav Goldberg, and Anders Søgaard. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 412–418.

Vidas Raudonis, Gintaras Dervinis, Andrius Vilkauskas, Agne Paulauskaite-Taraseviciene, and Gintare Kersulyte-Raudone. 2013. Evaluation of human emotion from eye motions. *Evaluation*, 4(8).

Keith Rayner and Susan A. Duffy. 1988. On-line comprehension processes and eye movements in reading. In *Reading research: Advances in theory and practice*, pages 13–66, New York, NY, USA. Academic Press.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 2121–2130.

Marek Rei and Anders Søgaard. 2018. Zero-shot sequence labeling: Transferring knowledge from sentences to tokens. *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2018)*, pages 293–302.

Marek Rei and Helen Yannakoudakis. 2016. Compositional sequence labeling models for error detection in learner writing. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1181–1191.

Marek Rei and Helen Yannakoudakis. 2017. Auxiliary objectives for neural error detection models. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 33–43.

Omid Rohanian, Shiva Taslimipoor, Victoria Yaneva, and Le An Ha. 2017. Using gaze data to predict multiword expressions. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 601–609.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. Semeval-2015 Task 10: Sentiment analysis in Twitter. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 451–463.

Jarkko Salojärvi, Ilpo Kojo, Jaana Simola, and Samuel Kaski. 2003. Can relevance be inferred from eye movements in information retrieval. In *Proceedings of WSOM*, volume 3, pages 261–266.

Javier San Agustin, Henrik Skovsgaard, John Paulin Hansen, and Dan Witzner Hansen. 2009. Low-cost gaze interaction: ready to deliver the promises. In *CHI'09 Extended Abstracts on Human Factors in Computing Systems*, pages 4453–4458. ACM.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models

for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 231–235.

Omer Tamuz, Ce Liu, Serge Belongie, Ohad Shamir, and Adam Tauman Kalai. 2011. Adaptively learning the crowd kernel. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 673–680.

Shaonan Wang, Jiajun Zhang, and Chengqing Zong. 2017. Learning sentence representation with guidance of human attention. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 4137–4143.

Zeerak Waseem. 2016. Are you a racist or am i seeing things? Annotator influence on hate speech detection on Twitter. In *Proceedings of the first workshop on NLP and computational social science*, pages 138–142.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Andrew Wilson, Christoph Dann, Chris Lucas, and Eric Xing. 2015. The human kernel. In *Advances in neural information processing systems (NIPS)*, pages 2854–2862.

Theresa Wilson, Zornitsa Kozareva, Preslav Nakov, Sara Rosenthal, Veselin Stoyanov, and Alan Ritter. 2013. Sentiment analysis in Twitter. In *Proceedings of the Seventh International Workshop on Semantic*, pages 312–320.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 180–189. Association for Computational Linguistics.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1568–1575.

Improving Language Understanding by Generative Pre-Training

Alec Radford
OpenAI
alec@openai.com

Karthik Narasimhan
OpenAI
karthikn@openai.com

Tim Salimans
OpenAI
tim@openai.com

Ilya Sutskever
OpenAI
ilyasu@openai.com

Abstract

Natural language understanding comprises a wide range of diverse tasks such as textual entailment, question answering, semantic similarity assessment, and document classification. Although large unlabeled text corpora are abundant, labeled data for learning these specific tasks is scarce, making it challenging for discriminatively trained models to perform adequately. We demonstrate that large gains on these tasks can be realized by *generative pre-training* of a language model on a diverse corpus of unlabeled text, followed by *discriminative fine-tuning* on each specific task. In contrast to previous approaches, we make use of task-aware input transformations during fine-tuning to achieve effective transfer while requiring minimal changes to the model architecture. We demonstrate the effectiveness of our approach on a wide range of benchmarks for natural language understanding. Our general task-agnostic model outperforms discriminatively trained models that use architectures specifically crafted for each task, significantly improving upon the state of the art in 9 out of the 12 tasks studied. For instance, we achieve absolute improvements of 8.9% on commonsense reasoning (Stories Cloze Test), 5.7% on question answering (RACE), and 1.5% on textual entailment (MultiNLI).

1 Introduction

The ability to learn effectively from raw text is crucial to alleviating the dependence on supervised learning in natural language processing (NLP). Most deep learning methods require substantial amounts of manually labeled data, which restricts their applicability in many domains that suffer from a dearth of annotated resources [61]. In these situations, models that can leverage linguistic information from unlabeled data provide a valuable alternative to gathering more annotation, which can be time-consuming and expensive. Further, even in cases where considerable supervision is available, learning good representations in an unsupervised fashion can provide a significant performance boost. The most compelling evidence for this so far has been the extensive use of pre-trained word embeddings [10, 39, 42] to improve performance on a range of NLP tasks [8, 11, 26, 45].

Leveraging more than word-level information from unlabeled text, however, is challenging for two main reasons. First, it is unclear what type of optimization objectives are most effective at learning text representations that are useful for transfer. Recent research has looked at various objectives such as language modeling [44], machine translation [38], and discourse coherence [22], with each method outperforming the others on different tasks.[1] Second, there is no consensus on the most effective way to transfer these learned representations to the target task. Existing techniques involve a combination of making task-specific changes to the model architecture [43, 44], using intricate learning schemes [21] and adding auxiliary learning objectives [50]. These uncertainties have made it difficult to develop effective semi-supervised learning approaches for language processing.

[1] https://gluebenchmark.com/leaderboard

Preprint. Work in progress.

In this paper, we explore a semi-supervised approach for language understanding tasks using a combination of unsupervised pre-training and supervised fine-tuning. Our goal is to learn a universal representation that transfers with little adaptation to a wide range of tasks. We assume access to a large corpus of unlabeled text and several datasets with manually annotated training examples (target tasks). Our setup does not require these target tasks to be in the same domain as the unlabeled corpus. We employ a two-stage training procedure. First, we use a language modeling objective on the unlabeled data to learn the initial parameters of a neural network model. Subsequently, we adapt these parameters to a target task using the corresponding supervised objective.

For our model architecture, we use the *Transformer* [62], which has been shown to perform strongly on various tasks such as machine translation [62], document generation [34], and syntactic parsing [29]. This model choice provides us with a more structured memory for handling long-term dependencies in text, compared to alternatives like recurrent networks, resulting in robust transfer performance across diverse tasks. During transfer, we utilize task-specific input adaptations derived from traversal-style approaches [52], which process structured text input as a single contiguous sequence of tokens. As we demonstrate in our experiments, these adaptations enable us to fine-tune effectively with minimal changes to the architecture of the pre-trained model.

We evaluate our approach on four types of language understanding tasks – natural language inference, question answering, semantic similarity, and text classification. Our general task-agnostic model outperforms discriminatively trained models that employ architectures specifically crafted for each task, significantly improving upon the state of the art in 9 out of the 12 tasks studied. For instance, we achieve absolute improvements of 8.9% on commonsense reasoning (Stories Cloze Test) [40], 5.7% on question answering (RACE) [30], 1.5% on textual entailment (MultiNLI) [66] and 5.5% on the recently introduced GLUE multi-task benchmark [64]. We also analyzed zero-shot behaviors of the pre-trained model on four different settings and demonstrate that it acquires useful linguistic knowledge for downstream tasks.

2 Related Work

Semi-supervised learning for NLP Our work broadly falls under the category of semi-supervised learning for natural language. This paradigm has attracted significant interest, with applications to tasks like sequence labeling [24, 33, 57] or text classification [41, 70]. The earliest approaches used unlabeled data to compute word-level or phrase-level statistics, which were then used as features in a supervised model [33]. Over the last few years, researchers have demonstrated the benefits of using word embeddings [11, 39, 42], which are trained on unlabeled corpora, to improve performance on a variety of tasks [8, 11, 26, 45]. These approaches, however, mainly transfer word-level information, whereas we aim to capture higher-level semantics.

Recent approaches have investigated learning and utilizing more than word-level semantics from unlabeled data. Phrase-level or sentence-level embeddings, which can be trained using an unlabeled corpus, have been used to encode text into suitable vector representations for various target tasks [28, 32, 1, 36, 22, 12, 56, 31].

Unsupervised pre-training Unsupervised pre-training is a special case of semi-supervised learning where the goal is to find a good initialization point instead of modifying the supervised learning objective. Early works explored the use of the technique in image classification [20, 49, 63] and regression tasks [3]. Subsequent research [15] demonstrated that pre-training acts as a regularization scheme, enabling better generalization in deep neural networks. In recent work, the method has been used to help train deep neural networks on various tasks like image classification [69], speech recognition [68], entity disambiguation [17] and machine translation [48].

The closest line of work to ours involves pre-training a neural network using a language modeling objective and then fine-tuning it on a target task with supervision. Dai et al. [13] and Howard and Ruder [21] follow this method to improve text classification. However, although the pre-training phase helps capture some linguistic information, their usage of LSTM models restricts their prediction ability to a short range. In contrast, our choice of transformer networks allows us to capture longer-range linguistic structure, as demonstrated in our experiments. Further, we also demonstrate the effectiveness of our model on a wider range of tasks including natural language inference, paraphrase detection and story completion. Other approaches [43, 44, 38] use hidden representations from a

pre-trained language or machine translation model as auxiliary features while training a supervised model on the target task. This involves a substantial amount of new parameters for each separate target task, whereas we require minimal changes to our model architecture during transfer.

Auxiliary training objectives Adding auxiliary unsupervised training objectives is an alternative form of semi-supervised learning. Early work by Collobert and Weston [10] used a wide variety of auxiliary NLP tasks such as POS tagging, chunking, named entity recognition, and language modeling to improve semantic role labeling. More recently, Rei [50] added an auxiliary language modeling objective to their target task objective and demonstrated performance gains on sequence labeling tasks. Our experiments also use an auxiliary objective, but as we show, unsupervised pre-training already learns several linguistic aspects relevant to target tasks.

3 Framework

Our training procedure consists of two stages. The first stage is learning a high-capacity language model on a large corpus of text. This is followed by a fine-tuning stage, where we adapt the model to a discriminative task with labeled data.

3.1 Unsupervised pre-training

Given an unsupervised corpus of tokens $\mathcal{U} = \{u_1, \ldots, u_n\}$, we use a standard language modeling objective to maximize the following likelihood:

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta) \tag{1}$$

where k is the size of the context window, and the conditional probability P is modeled using a neural network with parameters Θ. These parameters are trained using stochastic gradient descent [51].

In our experiments, we use a multi-layer *Transformer decoder* [34] for the language model, which is a variant of the transformer [62]. This model applies a multi-headed self-attention operation over the input context tokens followed by position-wise feedforward layers to produce an output distribution over target tokens:

$$\begin{aligned} h_0 &= UW_e + W_p \\ h_l &= \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n] \\ P(u) &= \texttt{softmax}(h_n W_e^T) \end{aligned} \tag{2}$$

where $U = (u_{-k}, \ldots, u_{-1})$ is the context vector of tokens, n is the number of layers, W_e is the token embedding matrix, and W_p is the position embedding matrix.

3.2 Supervised fine-tuning

After training the model with the objective in Eq. 1, we adapt the parameters to the supervised target task. We assume a labeled dataset $\mathcal{C}$, where each instance consists of a sequence of input tokens, $x^1, \ldots, x^m$, along with a label y. The inputs are passed through our pre-trained model to obtain the final transformer block's activation h_l^m, which is then fed into an added linear output layer with parameters W_y to predict y:

$$P(y | x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y). \tag{3}$$

This gives us the following objective to maximize:

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y | x^1, \ldots, x^m). \tag{4}$$

We additionally found that including language modeling as an auxiliary objective to the fine-tuning helped learning by (a) improving generalization of the supervised model, and (b) accelerating convergence. This is in line with prior work [50, 43], who also observed improved performance with such an auxiliary objective. Specifically, we optimize the following objective (with weight λ):

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C}) \tag{5}$$

Overall, the only extra parameters we require during fine-tuning are W_y, and embeddings for delimiter tokens (described below in Section 3.3).

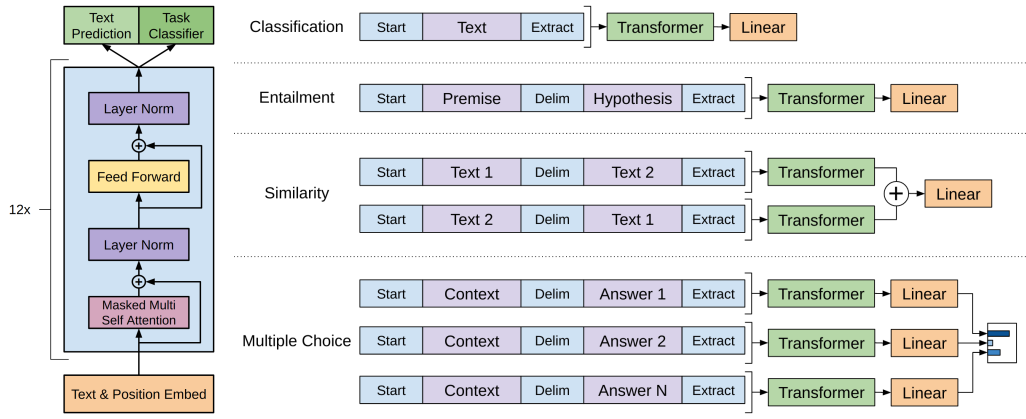


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

3.3 Task-specific input transformations

For some tasks, like text classification, we can directly fine-tune our model as described above. Certain other tasks, like question answering or textual entailment, have structured inputs such as ordered sentence pairs, or triplets of document, question, and answers. Since our pre-trained model was trained on contiguous sequences of text, we require some modifications to apply it to these tasks. Previous work proposed learning task specific architectures on top of transferred representations [44]. Such an approach re-introduces a significant amount of task-specific customization and does not use transfer learning for these additional architectural components. Instead, we use a traversal-style approach [52], where we convert structured inputs into an ordered sequence that our pre-trained model can process. These input transformations allow us to avoid making extensive changes to the architecture across tasks. We provide a brief description of these input transformations below and Figure 1 provides a visual illustration. All transformations include adding randomly initialized start and end tokens ($\langle s \rangle$, $\langle e \rangle$).

Textual entailment For entailment tasks, we concatenate the premise p and hypothesis h token sequences, with a delimiter token ($) in between.

Similarity For similarity tasks, there is no inherent ordering of the two sentences being compared. To reflect this, we modify the input sequence to contain both possible sentence orderings (with a delimiter in between) and process each independently to produce two sequence representations h_l^m which are added element-wise before being fed into the linear output layer.

Question Answering and Commonsense Reasoning For these tasks, we are given a context document z, a question q, and a set of possible answers $\{a_k\}$. We concatenate the document context and question with each possible answer, adding a delimiter token in between to get $[z; q; \$; a_k]$. Each of these sequences are processed independently with our model and then normalized via a softmax layer to produce an output distribution over possible answers.

4 Experiments

4.1 Setup

Unsupervised pre-training We use the BooksCorpus dataset [71] for training the language model. It contains over 7,000 unique unpublished books from a variety of genres including Adventure, Fantasy, and Romance. Crucially, it contains long stretches of contiguous text, which allows the generative model to learn to condition on long-range information. An alternative dataset, the 1B Word Benchmark, which is used by a similar approach, ELMo [44], is approximately the same size

Table 1: A list of the different tasks and datasets used in our experiments.

Task	Datasets
Natural language inference	SNLI [5], MultiNLI [66], Question NLI [64], RTE [4], SciTail [25]
Question Answering	RACE [30], Story Cloze [40]
Sentence similarity	MSR Paraphrase Corpus [14], Quora Question Pairs [9], STS Benchmark [6]
Classification	Stanford Sentiment Treebank-2 [54], CoLA [65]

but is shuffled at a sentence level - destroying long-range structure. Our language model achieves a very low token level perplexity of 18.4 on this corpus.

Model specifications Our model largely follows the original transformer work [62]. We trained a 12-layer decoder-only transformer with masked self-attention heads (768 dimensional states and 12 attention heads). For the position-wise feed-forward networks, we used 3072 dimensional inner states. We used the Adam optimization scheme [27] with a max learning rate of 2.5e-4. The learning rate was increased linearly from zero over the first 2000 updates and annealed to 0 using a cosine schedule. We train for 100 epochs on minibatches of 64 randomly sampled, contiguous sequences of 512 tokens. Since layernorm [2] is used extensively throughout the model, a simple weight initialization of $N(0, 0.02)$ was sufficient. We used a bytepair encoding (BPE) vocabulary with 40,000 merges [53] and residual, embedding, and attention dropouts with a rate of 0.1 for regularization. We also employed a modified version of L2 regularization proposed in [37], with $w = 0.01$ on all non bias or gain weights. For the activation function, we used the Gaussian Error Linear Unit (GELU) [18]. We used learned position embeddings instead of the sinusoidal version proposed in the original work. We use the *ftfy* library[2] to clean the raw text in BooksCorpus, standardize some punctuation and whitespace, and use the *spaCy* tokenizer.[3]

Fine-tuning details Unless specified, we reuse the hyperparameter settings from unsupervised pre-training. We add dropout to the classifier with a rate of 0.1. For most tasks, we use a learning rate of 6.25e-5 and a batchsize of 32. Our model finetunes quickly and 3 epochs of training was sufficient for most cases. We use a linear learning rate decay schedule with warmup over 0.2% of training. λ was set to 0.5.

4.2 Supervised fine-tuning

We perform experiments on a variety of supervised tasks including natural language inference, question answering, semantic similarity, and text classification. Some of these tasks are available as part of the recently released GLUE multi-task benchmark [64], which we make use of. Figure 1 provides an overview of all the tasks and datasets.

Natural Language Inference The task of natural language inference (NLI), also known as recognizing textual entailment, involves reading a pair of sentences and judging the relationship between them from one of `entailment`, `contradiction` or `neutral`. Although there has been a lot of recent interest [58, 35, 44], the task remains challenging due to the presence of a wide variety of phenomena like lexical entailment, coreference, and lexical and syntactic ambiguity. We evaluate on five datasets with diverse sources, including image captions (SNLI), transcribed speech, popular fiction, and government reports (MNLI), Wikipedia articles (QNLI), science exams (SciTail) or news articles (RTE).

Table 2 details various results on the different NLI tasks for our model and previous state-of-the-art approaches. Our method significantly outperforms the baselines on four of the five datasets, achieving absolute improvements of upto 1.5% on MNLI, 5% on SciTail, 5.8% on QNLI and 0.6% on SNLI over the previous best results. This demonstrates our model's ability to better reason over multiple sentences, and handle aspects of linguistic ambiguity. On RTE, one of the smaller datasets we evaluate on (2490 examples), we achieve an accuracy of 56%, which is below the 61.7% reported by a multi-task biLSTM model. Given the strong performance of our approach on larger NLI datasets, it is likely our model will benefit from multi-task training as well but we have not explored this currently.

[2] `https://ftfy.readthedocs.io/en/latest/`
[3] `https://spacy.io/`

Table 2: Experimental results on natural language inference tasks, comparing our model with current state-of-the-art methods. 5x indicates an ensemble of 5 models. All datasets use accuracy as the evaluation metric.

Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	89.3	-	-	-
CAFE [58] (5x)	80.2	79.0	89.3	-	-	-
Stochastic Answer Network [35] (3x)	80.6	80.1	-	-	-	-
CAFE [58]	78.7	77.9	88.5	83.3		
GenSen [64]	71.4	71.3	-	-	82.3	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	**61.7**
Finetuned Transformer LM (ours)	**82.1**	**81.4**	**89.9**	**88.3**	**88.1**	56.0

Table 3: Results on question answering and commonsense reasoning, comparing our model with current state-of-the-art methods.. 9x means an ensemble of 9 models.

Method	Story Cloze	RACE-m	RACE-h	RACE
val-LS-skip [55]	76.5	-	-	-
Hidden Coherence Model [7]	77.6	-	-	-
Dynamic Fusion Net [67] (9x)	-	55.6	49.4	51.2
BiAttention MRU [59] (9x)	-	60.2	50.3	53.3
Finetuned Transformer LM (ours)	**86.5**	**62.9**	**57.4**	**59.0**

Question answering and commonsense reasoning Another task that requires aspects of single and multi-sentence reasoning is question answering. We use the recently released RACE dataset [30], consisting of English passages with associated questions from middle and high school exams. This corpus has been shown to contain more reasoning type questions that other datasets like CNN [19] or SQuaD [47], providing the perfect evaluation for our model which is trained to handle long-range contexts. In addition, we evaluate on the Story Cloze Test [40], which involves selecting the correct ending to multi-sentence stories from two options. On these tasks, our model again outperforms the previous best results by significant margins - up to 8.9% on Story Cloze, and 5.7% overall on RACE. This demonstrates the ability of our model to handle long-range contexts effectively.

Semantic Similarity Semantic similarity (or paraphrase detection) tasks involve predicting whether two sentences are semantically equivalent or not. The challenges lie in recognizing rephrasing of concepts, understanding negation, and handling syntactic ambiguity. We use three datasets for this task – the Microsoft Paraphrase corpus (MRPC) [14] (collected from news sources), the Quora Question Pairs (QQP) dataset [9], and the Semantic Textual Similarity benchmark (STS-B) [6]. We obtain state-of-the-art results on two of the three semantic similarity tasks (Table 4) with a 1 point absolute gain on STS-B. The performance delta on QQP is significant, with a 4.2% absolute improvement over Single-task BiLSTM + ELMo + Attn.

Classification Finally, we also evaluate on two different text classification tasks. The Corpus of Linguistic Acceptability (CoLA) [65] contains expert judgements on whether a sentence is grammatical or not, and tests the innate linguistic bias of trained models. The Stanford Sentiment Treebank (SST-2) [54], on the other hand, is a standard binary classification task. Our model obtains an score of 45.4 on CoLA, which is an especially big jump over the previous best result of 35.0, showcasing the innate linguistic bias learned by our model. The model also achieves 91.3% accuracy on SST-2, which is competitive with the state-of-the-art results. We also achieve an overall score of 72.8 on the GLUE benchmark, which is significantly better than the previous best of 68.9.

Table 4: Semantic similarity and classification results, comparing our model with current state-of-the-art methods. All task evaluations in this table were done using the GLUE benchmark. (*mc*= Mathews correlation, *acc*=Accuracy, *pc*=Pearson correlation)

Method	**Classification**		**Semantic Similarity**			**GLUE**
	CoLA (mc)	SST2 (acc)	MRPC (F1)	STSB (pc)	QQP (F1)	
Sparse byte mLSTM [16]	-	**93.2**	-	-	-	-
TF-KLD [23]	-	-	**86.0**	-	-	-
ECNU (mixed ensemble) [60]	-	-	-	81.0	-	-
Single-task BiLSTM + ELMo + Attn [64]	35.0	90.2	80.2	55.5	66.1	64.8
Multi-task BiLSTM + ELMo + Attn [64]	18.9	91.6	83.5	72.8	63.3	68.9
Finetuned Transformer LM (ours)	**45.4**	91.3	82.3	**82.0**	**70.3**	**72.8**

Overall, our approach achieves new state-of-the-art results in 9 out of the 12 datasets we evaluate on, outperforming ensembles in many cases. Our results also indicate that our approach works well across datasets of different sizes, from smaller datasets such as STS-B (≈5.7k training examples) – to the largest one – SNLI (≈550k training examples).

5 Analysis

Impact of number of layers transferred We observed the impact of transferring a variable number of layers from unsupervised pre-training to the supervised target task. Figure 2(left) illustrates the performance of our approach on MultiNLI and RACE as a function of the number of layers transferred. We observe the standard result that transferring embeddings improves performance and that each transformer layer provides further benefits up to 9% for full transfer on MultiNLI. This indicates that each layer in the pre-trained model contains useful functionality for solving target tasks.

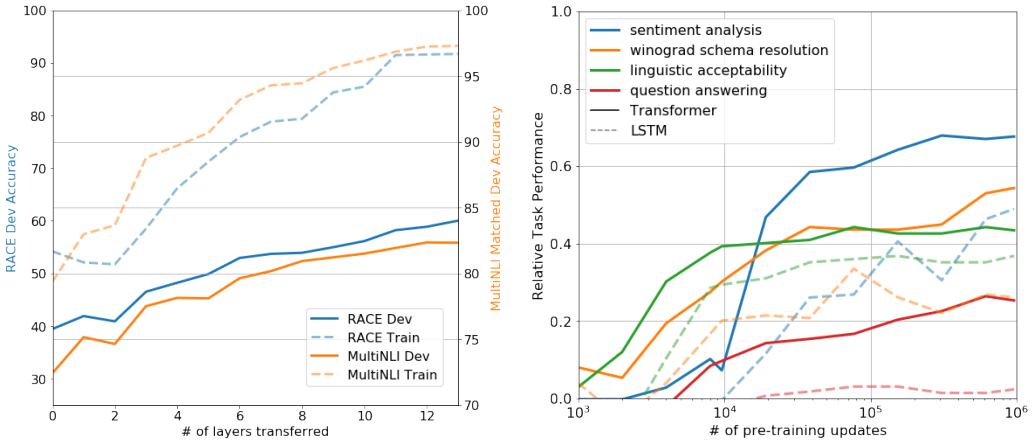


Figure 2: (**left**) Effect of transferring increasing number of layers from the pre-trained language model on RACE and MultiNLI. (**right**) Plot showing the evolution of zero-shot performance on different tasks as a function of LM pre-training updates. Performance per task is normalized between a random guess baseline and the current state-of-the-art with a single model.

Zero-shot Behaviors We'd like to better understand why language model pre-training of transformers is effective. A hypothesis is that the underlying generative model learns to perform many of the tasks we evaluate on in order to improve its language modeling capability and that the more structured

Table 5: Analysis of various model ablations on different tasks. Avg. score is a unweighted average of all the results. (*mc*= Mathews correlation, *acc*=Accuracy, *pc*=Pearson correlation)

Method	Avg. Score	CoLA (mc)	SST2 (acc)	MRPC (F1)	STSB (pc)	QQP (F1)	MNLI (acc)	QNLI (acc)	RTE (acc)
Transformer w/ aux LM (full)	74.7	45.4	91.3	82.3	82.0	**70.3**	**81.8**	**88.1**	**56.0**
Transformer w/o pre-training	59.9	18.9	84.0	79.4	30.9	65.5	75.7	71.2	53.8
Transformer w/o aux LM	**75.0**	**47.9**	**92.0**	**84.9**	**83.2**	69.8	81.1	86.9	54.4
LSTM w/ aux LM	69.1	30.3	90.5	83.2	71.8	68.1	73.7	81.1	54.6

attentional memory of the transformer assists in transfer compared to LSTMs. We designed a series of heuristic solutions that use the underlying generative model to perform tasks without supervised finetuning. We visualize the effectiveness of these heuristic solutions over the course of generative pre-training in Fig 2(right). We observe the performance of these heuristics is stable and steadily increases over training suggesting that generative pretraining supports the learning of a wide variety of task relevant functionality. We also observe the LSTM exhibits higher variance in its zero-shot performance suggesting that the inductive bias of the Transformer architecture assists in transfer.

For CoLA (linguistic acceptability), examples are scored as the average token log-probability the generative model assigns and predictions are made by thresholding. For SST-2 (sentiment analysis), we append the token *very* to each example and restrict the language model's output distribution to only the words *positive* and *negative* and guess the token it assigns higher probability to as the prediction. For RACE (question answering), we pick the answer the generative model assigns the highest average token log-probability when conditioned on the document and question. For DPRD [46] (winograd schemas), we replace the definite pronoun with the two possible referrents and predict the resolution that the generative model assigns higher average token log-probability to the rest of the sequence after the substitution.

Ablation studies We perform three different ablation studies (Table 5). First, we examine the performance of our method without the auxiliary LM objective during fine-tuning. We observe that the auxiliary objective helps on the NLI tasks and QQP. Overall, the trend suggests that larger datasets benefit from the auxiliary objective but smaller datasets do not. Second, we analyze the effect of the Transformer by comparing it with a single layer 2048 unit LSTM using the same framework. We observe a 5.6 average score drop when using the LSTM instead of the Transformer. The LSTM only outperforms the Transformer on one dataset – MRPC. Finally, we also compare with our transformer architecture directly trained on supervised target tasks, without pre-training. We observe that the lack of pre-training hurts performance across all the tasks, resulting in a 14.8% decrease compared to our full model.

6 Conclusion

We introduced a framework for achieving strong natural language understanding with a single task-agnostic model through generative pre-training and discriminative fine-tuning. By pre-training on a diverse corpus with long stretches of contiguous text our model acquires significant world knowledge and ability to process long-range dependencies which are then successfully transferred to solving discriminative tasks such as question answering, semantic similarity assessment, entailment determination, and text classification, improving the state of the art on 9 of the 12 datasets we study. Using unsupervised (pre-)training to boost performance on discriminative tasks has long been an important goal of Machine Learning research. Our work suggests that achieving significant performance gains is indeed possible, and offers hints as to what models (Transformers) and data sets (text with long range dependencies) work best with this approach. We hope that this will help enable new research into unsupervised learning, for both natural language understanding and other domains, further improving our understanding of how and when unsupervised learning works.

References

[1] S. Arora, Y. Liang, and T. Ma. A simple but tough-to-beat baseline for sentence embeddings. 2016.

[2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007.

[4] L. Bentivogli, P. Clark, I. Dagan, and D. Giampiccolo. The fifth pascal recognizing textual entailment challenge. In *TAC*, 2009.

[5] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. *EMNLP*, 2015.

[6] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

[7] S. Chaturvedi, H. Peng, and D. Roth. Story comprehension for predicting what happens next. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1603–1614, 2017.

[8] D. Chen and C. Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

[9] Z. Chen, H. Zhang, X. Zhang, and L. Zhao. Quora question pairs. https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs, 2018.

[10] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[11] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[12] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. *EMNLP*, 2017.

[13] A. M. Dai and Q. V. Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087, 2015.

[14] W. B. Dolan and C. Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[15] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

[16] S. Gray, A. Radford, and K. P. Diederik. Gpu kernels for block-sparse weights. 2017.

[17] Z. He, S. Liu, M. Li, M. Zhou, L. Zhang, and H. Wang. Learning entity representation for entity disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 30–34, 2013.

[18] D. Hendrycks and K. Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *arXiv preprint arXiv:1606.08415*, 2016.

[19] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015.

[20] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[21] J. Howard and S. Ruder. Universal language model fine-tuning for text classification. *Association for Computational Linguistics (ACL)*, 2018.

[22] Y. Jernite, S. R. Bowman, and D. Sontag. Discourse-based objectives for fast unsupervised sentence representation learning. *arXiv preprint arXiv:1705.00557*, 2017.

[23] Y. Ji and J. Eisenstein. Discriminative improvements to distributional sentence similarity. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 891–896, 2013.

[24] F. Jiao, S. Wang, C.-H. Lee, R. Greiner, and D. Schuurmans. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 209–216. Association for Computational Linguistics, 2006.

[25] T. Khot, A. Sabharwal, and P. Clark. Scitail: A textual entailment dataset from science question answering. In *Proceedings of AAAI*, 2018.

[26] Y. Kim. Convolutional neural networks for sentence classification. *EMNLP*, 2014.

[27] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[28] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

[29] N. Kitaev and D. Klein. Constituency parsing with a self-attentive encoder. *ACL*, 2018.

[30] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy. Race: Large-scale reading comprehension dataset from examinations. *EMNLP*, 2017.

[31] G. Lample, L. Denoyer, and M. Ranzato. Unsupervised machine translation using monolingual corpora only. *ICLR*, 2018.

[32] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

[33] P. Liang. *Semi-supervised learning for natural language*. PhD thesis, Massachusetts Institute of Technology, 2005.

[34] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer. Generating wikipedia by summarizing long sequences. *ICLR*, 2018.

[35] X. Liu, K. Duh, and J. Gao. Stochastic answer networks for natural language inference. *arXiv preprint arXiv:1804.07888*, 2018.

[36] L. Logeswaran and H. Lee. An efficient framework for learning sentence representations. *ICLR*, 2018.

[37] I. Loshchilov and F. Hutter. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 2017.

[38] B. McCann, J. Bradbury, C. Xiong, and R. Socher. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6297–6308, 2017.

[39] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[40] N. Mostafazadeh, M. Roth, A. Louis, N. Chambers, and J. Allen. Lsdsem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 46–51, 2017.

[41] K. Nigam, A. McCallum, and T. Mitchell. Semi-supervised text classification using em. *Semi-Supervised Learning*, pages 33–56, 2006.

[42] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[43] M. E. Peters, W. Ammar, C. Bhagavatula, and R. Power. Semi-supervised sequence tagging with bidirectional language models. *ACL*, 2017.

[44] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *NAACL*, 2018.

[45] Y. Qi, D. S. Sachan, M. Felix, S. J. Padmanabhan, and G. Neubig. When and why are pre-trained word embeddings useful for neural machine translation? *NAACL*, 2018.

[46] A. Rahman and V. Ng. Resolving complex cases of definite pronouns: the winograd schema challenge. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 777–789. Association for Computational Linguistics, 2012.

[47] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text. *EMNLP*, 2016.

[48] P. Ramachandran, P. J. Liu, and Q. V. Le. Unsupervised pretraining for sequence to sequence learning. *arXiv preprint arXiv:1611.02683*, 2016.

[49] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144, 2007.

[50] M. Rei. Semi-supervised multitask learning for sequence labeling. *ACL*, 2017.

[51] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[52] T. Rocktäschel, E. Grefenstette, K. M. Hermann, T. Kočiskỳ, and P. Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

[53] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[54] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[55] S. Srinivasan, R. Arora, and M. Riedl. A simple and effective approach to the story cloze test. *arXiv preprint arXiv:1803.05547*, 2018.

[56] S. Subramanian, A. Trischler, Y. Bengio, and C. J. Pal. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*, 2018.

[57] J. Suzuki and H. Isozaki. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. *Proceedings of ACL-08: HLT*, pages 665–673, 2008.

[58] Y. Tay, L. A. Tuan, and S. C. Hui. A compare-propagate architecture with alignment factorization for natural language inference. *arXiv preprint arXiv:1801.00102*, 2017.

[59] Y. Tay, L. A. Tuan, and S. C. Hui. Multi-range reasoning for machine comprehension. *arXiv preprint arXiv:1803.09074*, 2018.

[60] J. Tian, Z. Zhou, M. Lan, and Y. Wu. Ecnu at semeval-2017 task 1: Leverage kernel-based traditional nlp features and neural networks to build a universal model for multilingual and cross-lingual semantic textual similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 191–197, 2017.

[61] Y. Tsvetkov. Opportunities and challenges in working with low-resource languages. *CMU*, 2017.

[62] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010, 2017.

[63] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[64] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[65] A. Warstadt, A. Singh, and S. R. Bowman. Corpus of linguistic acceptability. http://nyu-mll.github.io/cola, 2018.

[66] A. Williams, N. Nangia, and S. R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *NAACL*, 2018.

[67] Y. Xu, J. Liu, J. Gao, Y. Shen, and X. Liu. Towards human-level machine reading comprehension: Reasoning and inference with multiple strategies. *arXiv preprint arXiv:1711.04964*, 2017.

[68] D. Yu, L. Deng, and G. Dahl. Roles of pre-training and fine-tuning in context-dependent dbn-hmms for real-world speech recognition. In *Proc. NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[69] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, volume 1, page 6, 2017.

[70] X. Zhu. Semi-supervised learning literature survey. 2005.

[71] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.