

Learning and Evaluating General Linguistic Intelligence

Dani Yogatama, Cyprien de Masson d’Autume, Jerome Connor,
Tomas Kocisky, Mike Chrzanowski, Lingpeng Kong,
Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, Phil Blunsom
DeepMind, London, United Kingdom

{dyogatama,cyprien,jeromeconnor,tkocisky,chrzanowskim}@google.com
{lingpenk,angeliki,lingwang,leiyu,cdyer,pblunsom}@google.com

Abstract

We define general linguistic intelligence as the ability to reuse previously acquired knowledge about a language’s lexicon, syntax, semantics, and pragmatic conventions to adapt to new tasks quickly. Using this definition, we analyze state-of-the-art natural language understanding models and conduct an extensive empirical investigation to evaluate them against these criteria through a series of experiments that assess the task-independence of the knowledge being acquired by the learning process. **In addition to task performance, we propose a new evaluation metric based on an online encoding of the test data that quantifies how quickly an existing agent (model) learns a new task.** Our results show that while the field has made impressive progress in terms of model architectures that generalize to many tasks, these models still require a lot of in-domain training examples (e.g., for fine tuning, training task-specific modules), and are prone to catastrophic forgetting. **Moreover, we find that far from solving general tasks (e.g., document question answering), our models are overfitting to the quirks of particular datasets (e.g., SQuAD).** We discuss missing components and conjecture on how to make progress toward general linguistic intelligence.

1 Introduction

Advances in deep learning techniques (e.g., attention mechanisms, memory modules, and architecture search) have considerably improved natural language processing (NLP) models on many important tasks. For example, machine performance on both Chinese–English machine translation and document question answering on the Stanford question answering dataset (SQuAD; Rajpurkar et al., 2016) has been claimed to have surpassed human levels (Hassan et al., 2018; Devlin et al., 2018). While the tasks that initiated learning-based NLP models were motivated by external demands and are important applications in their own right (e.g., machine translation, question answering, automatic speech recognition, text to speech, etc.), there is a marked and troubling tendency for recent datasets to be set up to be easy to solve with little in the way of *generalization* or *abstraction*; for instance, ever larger datasets created by crowd-sourcing processes that may not well approximate the natural distributions they are intended to span, although there are some notable counter-examples (Kwiatkowski et al., 2019). When there exist multiple datasets that are representative of the exact same task from different domains (e.g., various question answering datasets), we rarely evaluate on all of them. This state of affairs promotes development of models that only work well for a specific purpose, overestimates our success at having solved the general task, fails to reward sample efficient generalization that requires the ability to discover and make use of rich linguistic structures, and ultimately limits progress.

Despite these methodological difficulties, recent breakthroughs demonstrate that models that are pretrained on a large unlabeled corpus perform well on many language understanding tasks with minimal modifications (Radford et al., 2018; Peters et al., 2018; Devlin et al., 2018). These models are trained on a large corpus in an unsupervised fashion on tasks such as language modeling (predicting the next word given a context), masked language modeling (predicting a missing word in a sentence from the context), and next sentence predictions (predicting whether two sentences are consecutive sentences). The models can then be used on various NLP tasks by adding an extra task-specific final layer and fine tuning on supervised data for the task of interest. Variants of these models achieve impressive results on the GLUE benchmark (sentence and sentence pair classification tasks; Wang et al., 2018) and the SQuAD question answering dataset, showing the clear benefit of, and significant progress on, transfer learning in NLP.

Another promising avenue is to train a model on multiple tasks simultaneously. McCann et al. (2018) train a question-answering language model on many different NLP tasks (e.g., machine translation, summarization, sentiment analysis, etc.) by formulating every task as question answering in order to make the input and output of the model be consistent. This model can then be used to perform multiple tasks by asking different questions at test time. While their overall multitask results are still below task-specific models, it represents a prototype for models of general linguistic intelligence.

In order to make progress toward models that manifest general linguistic intelligence, we argue for an evaluation paradigm that rewards abilities to (i) deal with the full complexity of natural language across a variety of tasks, (ii) effectively store and reuse representations, combinatorial modules (e.g. which compose words into representations of phrases, sentences, and documents), and previously acquired linguistic knowledge to avoid *catastrophic forgetting* (McCloskey & Cohen, 1989; French, 1999), and (iii) adapt to new linguistic tasks in new environments with little experience (i.e., robustness to domain shifts). This is inspired by the evolving artificial general intelligence task suites that are used to develop generic exploration, planning, and reasoning abilities in agents that learn to interact with simulated visual environments (Beattie et al., 2016; Brockman et al., 2016). Here, we focus on language processing modules that learn using primarily supervised and unsupervised methods from static corpora. We leave extensions to learning through interactions with simulated linguistic environments to future work.

In this paper we analyze the capabilities of previously proposed algorithms as models of general linguistic intelligence. To directly quantify how quickly an existing model adapts to a new task we propose a new evaluation metric (§4) based on online (prequential) coding (Blier & Ollivier, 2018). This evaluation considers the number of task-specific training examples needed to reach high performance, thus rewarding sample efficiency. Our experiments show that existing models still require considerable fine tuning on each task before they perform well (§5.1), and although they significantly benefit from transfer learning on other datasets from related tasks (§5.2), they generalize poorly to different datasets from the same task without fine tuning (§5.3), and suffer from catastrophic forgetting when trained on multiple datasets and tasks in a continual learning setup (§5.4). We discuss the implications of our findings and conclude by outlining potential directions toward general linguistic intelligence models (§6).

2 Tasks

In this section, we describe the tasks and datasets that we consider in our experiments. We evaluate many of our models on two main tasks: reading comprehension on SQuAD 1.1¹

¹<https://rajpurkar.github.io/SQuAD-explorer/>

(Rajpurkar et al., 2016) and Multi Genre Natural Language Inference (MNLI²; Williams et al., 2018). However, we also use the following tasks and datasets for training and evaluation (e.g., in a multitask setup, for pretraining, and for evaluation on out-of-domain genre).

Reading Comprehension. We use two additional reading comprehension datasets: TriviaQA³ (Joshi et al., 2017) and QuAC⁴ (Choi et al., 2018) in our experiments. While SQuAD 1.1, TriviaQA, and QuAC are from the same task, they have different characteristics. SQuAD1.1 is a reading comprehension dataset constructed from Wikipedia articles. It includes almost 90,000 training examples and 10,000 validation examples. TriviaQA is a dataset with question-answer pairs written by trivia enthusiasts and evidence collected retrospectively from Wikipedia and the web. We use the Web section of TriviaQA that contains 76,000 training examples and 300 verified validation examples. QuAC is an information-seeking dialog-style dataset where a student asks questions about a Wikipedia article and a teacher answers with a short excerpt from the article. It has 80,000 training examples and approximately 7,000 validation examples.

Semantic Role Labeling. We consider a question-answer driven semantic role labeling task from FitzGerald et al. (2018), where semantic role labeling is presented as a span prediction problem. Given a sentence and a predicate-driven wh-question, the goal is to predict the start and end indices of the answer in the sentence. There are around 200,000 training questions and 25,000 test questions in the dataset.⁵

Relation Extraction. Relation extraction is the task of extracting factual knowledge from a corpus. Levy et al. (2017) show that it can be formulated as a question answering problem by associating questions with relation slots. We use their zero-shot relation extraction dataset⁶ which consists of over 900,000 training examples and almost 5,000 test examples.

Natural Language Inference Natural language inference is a sentence pair classification problem where the goal is to predict relationships between two sentences from ENTAIL, CONTRADICT, and NEUTRAL. In addition to the MNLI dataset, we also use the Stanford Natural Language Inference (SNLI⁷; Bowman et al., 2015) dataset. MNLI (SNLI) contains 400,000 (550,000) training pairs and 20,000 (10,000) test pairs respectively.

3 Models

We focus on two classes of models: self-attention models (i.e., Transformer; Vaswani et al., 2017) and recurrent neural networks.

For self-attention models, we start from a pretrained BERT model (Devlin et al., 2018),⁸ which has been shown to be state-of-the-art on many natural language processing tasks. BERT is a big Transformer model (Vaswani et al., 2017) which is trained on two unsupervised tasks: masked language modeling and next sentence prediction. Devlin et al. (2018) also show that these unsupervised pretraining tasks result in better models compared to the standard language modeling task (i.e., next word prediction). We use the BERT_{BASE} model which has 12 Transformer

²<https://www.nyu.edu/projects/bowman/multinli/>

³<http://nlp.cs.washington.edu/triviaqa/>

⁴<http://quac.ai/>

⁵<http://qasrl.org/>

⁶<http://nlp.cs.washington.edu/zeroshot/>

⁷<https://nlp.stanford.edu/projects/snli/>

⁸<https://github.com/google-research/bert>

layers, 12 self-attention heads, and 768 hidden dimensions. There are 110 million parameters in total in this pretrained model. For each task that we consider, following the original BERT model, we add an additional task-specific final layer to output relevant predictions. Note that the final layer is task specific, as opposed to dataset specific. For BERT, we leave investigations of the ability to learn to understand new words to future work and use the default BERT vocabulary in our experiments. We denote this model BERT.

For recurrent neural networks, we augment a pretrained ELMo model—which has almost 100 million parameters—with a 300-dimensional bidirectional LSTM (Hochreiter & Schmidhuber, 1997). We then use a bidirectional attention flow network (BiDAF; Seo et al., 2017) to aggregate context (premise) and question (hypothesis) representations to predict an answer span (a label) in question answering (natural language inference). We set the dimensions of the bidirectional LSTMs in BiDAF’s modeling layer to 300 and output layer to 100. Unlike BERT, ELMo is a character-based model that is able to learn representations of new words. We refer to this model as ELMo in our experiments.

Our hyperparameters for both models are batch size, learning rate, dropout rate, and ℓ_2 regularization constant.

4 Evaluating Transfer

Existing NLP models are typically evaluated in terms of on their performance, at the end of training, on the task of interest as measured by the performance on a held-out test set. Optimization of these metrics over years is important to drive ongoing progress on a task. Aggregates of these metrics have also been used to measure (and drive) performance of models on multiple tasks. For example, the decaScore (McCann et al., 2018) is a simple summation of various metrics such as classification accuracy, F_1 score, ROUGE score, and others across a number of tasks. These performance metrics capture an essential aspect of intelligence: being able to generalize from experience with a class of inputs to new inputs. However, they are incomplete as none of them assess a defining attribute of general linguistic intelligence models: the ability to generalize rapidly to a new task by using previously acquired knowledge. For example, a model that only requires one hundred in-domain training examples to achieve 90% accuracy and does not improve again with more training examples from the same domain arguably exhibits more desirable learning capabilities than a model that takes one million examples to get to 90% before plateauing at 92%.

To quantify the difference in such learning behavior, we use a new online (prequential) code (Blier & Ollivier, 2018) to evaluate model performance in our experiments, in addition to traditional performance metrics. The online code is rooted in information theory and is based on connections between generalization, compression, and comprehension (Wolff, 1982; Chaitin, 2007). In general, the separation between training and test sets is arbitrary, and there is only one set of examples for a task that we wish to learn rapidly. Denote the set of N examples $\mathcal{A} = \{(x_i, y_i)\}_{i=1}^N$ and the model parameters \mathbf{W} . Consider different splits of the data $\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_N$, where \mathcal{A}_i is a subset that contains $\{(x_j, y_j)\}_{j=1}^i$ for some ordering of the examples (i.e., \mathcal{A}_i is the subset that contains i examples from (x_1, y_1) to (x_i, y_i)). Denote parameters of the model trained on a particular subset as $\hat{\mathbf{W}}_{\mathcal{A}_i}$.

We consider an online setup where the codelength is computed as:

$$\ell(\mathcal{A}) = \log_2 |\mathcal{Y}| - \sum_{i=2}^N \log_2 p(y_i | x_i; \hat{\mathbf{W}}_{\mathcal{A}_{i-1}}), \quad (1)$$

where $|\mathcal{Y}|$ is the number of possible classes (labels) in the data.

The codelength $\ell(\mathcal{A})$ can be interpreted as follows. Assume Alice has all (x_i, y_i) pairs in \mathcal{A} , Bob has just the x_i 's from \mathcal{A} , and that Alice wishes to communicate the y_i 's to Bob. One procedure would be for Alice to train a neural network, and send its parameters to Bob, but neural networks have large numbers of parameters that are expensive to communicate. The online procedure solves this problem by having Alice and Bob agree on the form of the model, its initial random seeds, and its learning algorithm. Alice starts by communicating y_1 with a uniform code, and then both Alice and Bob learn a model that predicts y from x , and Alice uses that model to communicate y_2 . The process repeats where both parties learn from ever larger datasets; since both models are trained using the same procedure, they stay in sync. This process continues until the entire dataset has been transmitted. In this sequential evaluation, a model that performs well with a limited number of training examples will be rewarded by having a shorter codelength (Alice will require fewer bits to transmit the subsequent y_i to Bob).

In the above formulation, the model is evaluated for every example in the training data, which can be very expensive in practice. A more realistic approach is to split \mathcal{A} into M increasing subsets, $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_M$ where $\mathcal{S}_M = \mathcal{A}$ and only evaluate the model M times. In this case, we have:

$$\ell(\mathcal{A}) = |\mathcal{S}_1| \log_2 |\mathcal{Y}| - \sum_{i=2}^M \log_2 p(y_{\mathcal{S}_i} \mid x_{\mathcal{S}_i}; \hat{\mathbf{W}}_{\mathcal{S}_{i-1}}),$$

where \mathcal{S}_i denotes the i -th data subset.

Online codelength is related to area under the learning curve (Guyon et al., 2011), as they share similar motivations, and also to dynamic sequence model evaluation (Krause et al., 2018). Like our online codelength, both also require a predefined dataset ordering. We note that while all these metrics consider the number of examples required to achieve a certain performance, they still do not fully consider model complexity (e.g., the number of parameters in the model) and training cost (e.g., how much computational resources are needed, how many hours, etc.). However, three nice features of online codelength are that: (i) it can be used across a number of tasks by any *probabilistic* model, (ii) it allows seamless incorporation of other model and training properties (e.g., model complexity, training cost) if desired, and (iii) it will reflect improvements in generalization performance due to better architecture, better initialization, and better learning algorithms. In our experiments below, we also show that it correlates well with standard evaluation metrics such as accuracy and F_1 scores.

5 Experiments

5.1 Unsupervised Pretraining

How much in-domain training data is needed to obtain good performance?

Our first set of experiments is designed to investigate how much training data is needed to achieve high performance on two language understanding tasks: reading comprehension on SQuAD 1.1 and natural language inference on MNLI. We show learning curves of two models as a function of the number of training examples on SQuAD1.1 and MNLI validation sets in Figure 1 and Figure 2. On both SQuAD and MNLI, both models are able to approach their asymptotic errors after seeing approximately 40,000 training examples, a surprisingly high number of examples for models that rely on pretrained modules (i.e., BERT and ELMo). While adding more training examples improves both models, the performance gains are not as significant as the first 40,000 examples. Between these two models, BERT outperforms ELMo on

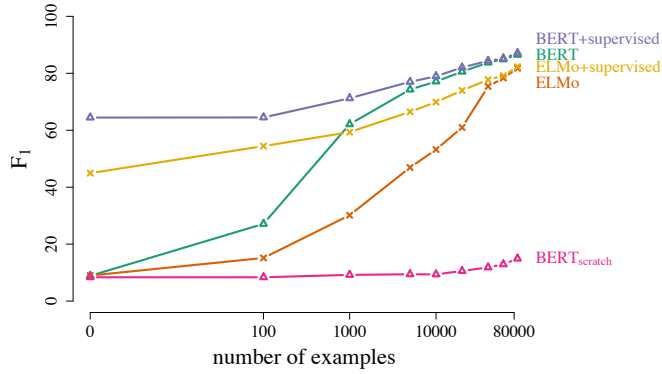


Figure 1: F_1 scores on SQuAD as a function of the number of training examples (log scale). BERT+supervised and ELMo+supervised denote BERT and ELMo models that are pretrained on other datasets and tasks (see §5.2), BERT_{scratch} denotes a Transformer with a similar architecture to BERT that is not pretrained on any unsupervised task at all (i.e., trained from scratch).

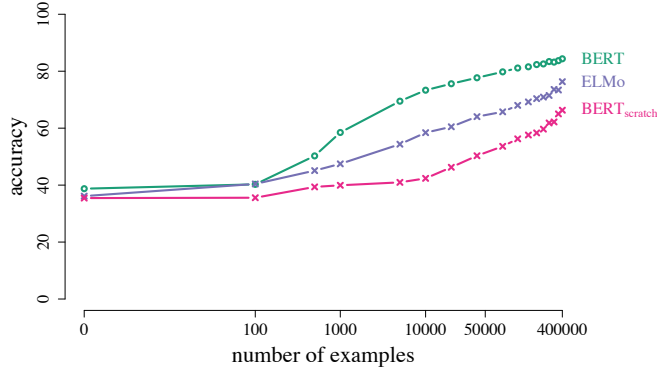


Figure 2: Classification accuracy on MNLI as a function of the number of training examples (log scale). BERT_{scratch} denotes a Transformer with a similar architecture to BERT that is not pretrained on any unsupervised task at all (i.e., trained from scratch).

both datasets and approaches its asymptotic error faster. Previous work have established the benefits of transfer learning from unsupervised tasks both in terms of sample complexity and overall performance (Dai & Le, 2015; Radford et al., 2018; Devlin et al., 2018). We also verify this in our experiments by training a Transformer with the same architecture as BERT on SQuAD and MNLI from scratch on the entire training set. This model can only achieve 14.9 F_1 score on SQuAD and 66.3 accuracy on MNLI after 500,000 training iterations, far below the performance of a similar model pretrained on unsupervised tasks. We note that our F_1 , exact match, and accuracy scores on SQuAD and MNLI validation sets are upper bound best results since we tune hyperparameters on the respective validation sets.

We next split the original SQuAD training set into 8 example subsets of 10,000 examples each and MNLI into 10 example subsets of 40,000 examples. We use uniform encoding for predicting the first example subset,⁹ use a “default” hyperparameter configuration for training the first model to predict the second example subset (i.e., we choose reasonable hyperparameter values based on intuitions), and select hyperparameters of the best model from the previous subset

⁹We observe that using uniform encoding for the first subset is better than using predictions from pretrained models with randomly initialized final layers.

for subsequent subsets to avoid tuning on evaluation subsets for predictions. We obtain online codelengths of 102.42 kbits (BERT) and 112.96 kbits (ELMo) for SQuAD1.1 and 89.25 kbits (BERT) and 132.17 kbits (ELMo) for MNLI. We can see that codelengths correlate with other evaluation metrics such as F_1 , exact match, and accuracy on these datasets (lower codelengths generally means higher F_1 , exact match, or accuracy), consistent with the findings of (Blier & Ollivier, 2018). We use these as baseline codelengths for our next set of experiments in §5.2.

5.2 Beyond Unsupervised Pretraining

Can pretraining on other datasets and tasks improve performance?

We next investigate a more general transfer learning strategy using related tasks beyond unsupervised objectives. Devlin et al. (2018) show that jointly training BERT on SQuAD and TriviaQA slightly improves final performance. Here, we focus on a pretraining setup to highlight the benefit of transfer. We discuss joint (multitask) training in §5.4.

We pretrain BERT and ELMo models on semantic role labeling, relation extraction, natural language inference, and other reading comprehension datasets (TriviaQA and QuAC) described in §2. We randomly sample a batch of examples from one of the tasks above in the pretraining phase for 100,000 iterations. We then take these pretrained models and train on only SQuAD. Table 1 show overall validation set results on SQuAD for each model.

Model	EM (\uparrow)	F_1 (\uparrow)	codelength (\downarrow)
BERT	78.5	86.5	102.4
BERT + supervised	79.4	87.1	31.7
ELMo	72.1	81.8	113.0
ELMo + supervised	72.8	82.3	54.5

Table 1: Results on SQuAD for BERT and ELMo pretrained on other supervised tasks vs. from only unsupervised tasks

The results show that pretraining on other datasets and tasks slightly improve performance in terms of final exact match and F_1 score. However, in terms of codelength, the models pretrained on other tasks significantly outperform the non-pretrained ones. A major reason for this is that pretraining on other question answering datasets trains the final layer (i.e., the decoder) used in SQuAD (since many of the pretrained datasets are question answering datasets), so the first example subset when computing the codelength can be predicted without using the uniform encoding. It is analogous to learning what a question answering task is, even though they are learning from different datasets.

As shown in Figure 3, the models are able to predict answers reasonably well even before seeing any SQuAD-specific training examples. BERT and ELMo achieve 62.9 and 43.3 F_1 scores after training on other tasks and datasets before seeing any SQuAD examples. These results give rise to an interesting question about how well models that are trained only on SQuAD perform on these other datasets, which we answer in §5.3. This experiment also shows that an online evaluation paradigm such as online codelength highlights different model characteristics that are not captured by metrics such as F_1 score or accuracy. A characteristic of the codelength metric is that models that perform significantly worse at the beginning can have problems catching up to the performance of models that start well, often referred to as the catch-up phenomenon (Erven et al., 2012). If such a property is not desired, we can switch between models by always choosing the best model to encode a particular subset when computing the codelength (we do not do this in our experiments).

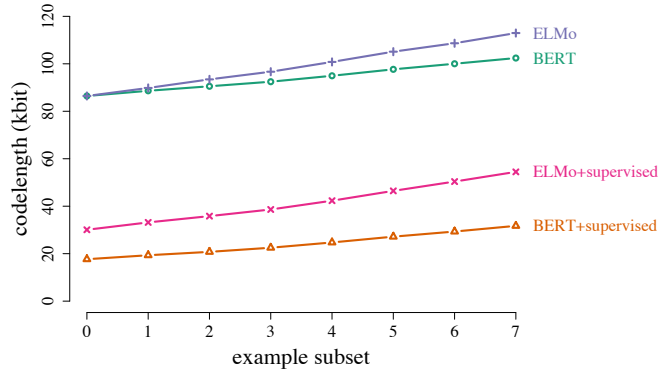


Figure 3: Online codelengths for different example subsets on SQuAD for BERT and ELMo pretrained on other supervised tasks vs. from only unsupervised tasks.

Since we observe benefits of transfer learning from other tasks, another important question is how to select tasks to pretrain on, and whether there is a training curriculum that a model should follow. Ideally, a general linguistic intelligence model should be able to consume examples from any kind of task in any order. Our experiments in §5.4 will provide some insights into training curriculum.

5.3 Generalization

Do these models generalize to other datasets from the same task?

Our next set of experiments is to analyze generalization properties of existing models. We investigate whether our models overfit to a specific dataset (i.e., solving the dataset) it is trained on or whether they are able to learn general representations and modules (i.e., solving the task).

We take our best SQuAD models from §5.1 and evaluate them on other question-answering datasets to see whether these models generalize to other datasets from the same task. Table 2 shows results on TriviaQA, QuAC, QA-SRL, and QA-ZRE.¹⁰ We see that high-performing SQuAD models do not perform well on other datasets without being provided with training examples from these datasets. These results are not surprising since the examples come from different distributions, but they do highlight the fact that there is a substantial gap between learning a task and learning a dataset.

Our results indicate that models that work well on SQuAD still require training examples from other datasets before it can serve as a general-purpose question answering model. However, our previous experiments in §5.2 show that training on other tasks and datasets speed up learning on SQuAD greatly, so there is useful information that can be extracted from these tasks. We next discuss the interaction between training curriculum and model performance.

	SQuAD	Trivia	QuAC	QA-SRL	QA-ZRE
BERT	86.5 (78.5)	35.6 (13.4)	56.2 (43.9)	77.5 (65.0)	55.3 (40.0)
ELMo	81.8 (72.2)	32.9 (12.6)	45.0 (34.5)	68.7 (52.3)	60.2 (42.0)

Table 2: F_1 (exact match) scores of the best BERT and ELMo models trained on SQuAD and evaluated on other question answering datasets.

¹⁰Our results on these four datasets are not necessarily comparable with results from other work since we remove unanswerable questions (as identified by the dataset creator) for simplicity.

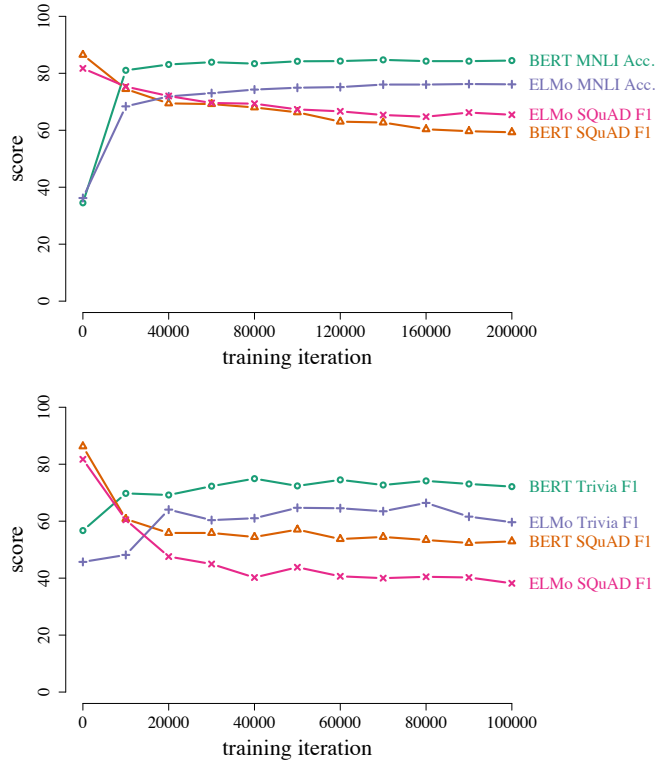


Figure 4: Catastrophic forgetting in a continual learning setup on unsupervised \rightarrow SQuAD \rightarrow MNLI (top) and unsupervised \rightarrow SQuAD \rightarrow TriviaQA (bottom).

5.4 Curriculum and Catastrophic Forgetting

How fast do these models forget their previously acquired linguistic knowledge?

How does curriculum affect performance and how do we design this curriculum?

An important characteristic of general linguistic intelligence models is the ability to store and reuse linguistic knowledge throughout their lifetime and avoid catastrophic forgetting. First, we consider a continual learning setup, where we train our best SQuAD-trained BERT and ELMo from §5.1 on a new task, TriviaQA or MNLI. Figure 4 shows results of our experiments. We can see that in a continual learning (transfer) setup, the performance on both models on the first supervised task they are trained on (SQuAD) rapidly degrades. This is true even for MNLI, which has a task-specific final layer for classification that is not shared with the original question-answering tasks (thus indicating that the underlying representations being computed are changing). We search over a wide range of possible learning rate initial values and none of the models that obtain good results on the new task are able to maintain performance on SQuAD. For BERT, we also include an extended continual learning result for: unsupervised \rightarrow SQuAD \rightarrow TriviaQA \rightarrow MNLI in Figure 5 (using the model from unsupervised \rightarrow SQuAD \rightarrow TriviaQA in the previous experiment). Interestingly, adding a new MNLI task does not decrease the performance on SQuAD further, although it does make the model forget how to do well on TriviaQA.

We next investigate whether there is a curriculum that allows these models to perform well on all tasks. We train BERT and ELMo on all tasks in §2, where at each iteration we sample a batch of examples from a randomly chosen task (with uniform probability). Table 3 shows performance of

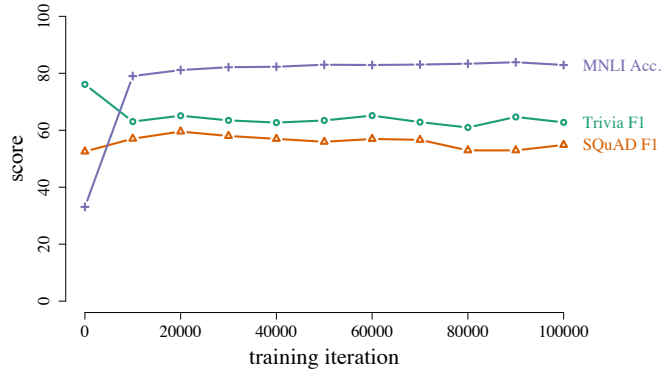


Figure 5: Catastrophic forgetting in a continual learning setup for BERT on unsupervised \rightarrow SQuAD \rightarrow TriviaQA \rightarrow MNLI (bottom).

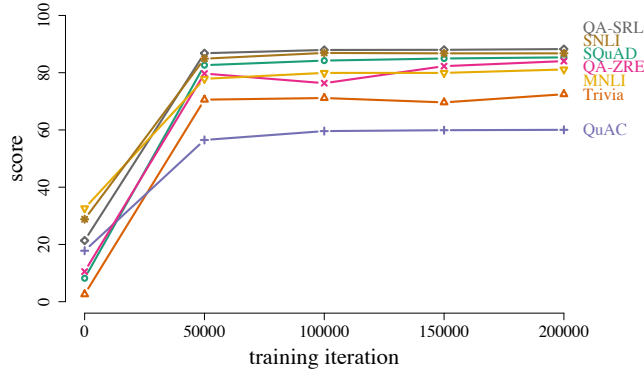


Figure 6: Model scores on various datasets with a random training curriculum.

these models after 200,000 training iterations. Interestingly, unlike in the continual learning setup, these models are able to perform reasonably well on many tasks (especially BERT). Figure 6 shows performance of BERT on all tasks as training progresses. We can see that using a random training curriculum allows the model to not forget previously acquired knowledge, and that after about 50,000 iterations the model can work reasonably well on all tasks. However, the drawback of such a curriculum is that it either requires all tasks to be presented at the beginning or retraining on all tasks after seeing a new task. It is therefore still an open question how to transfer effectively in a continual learning setup.

	SQuAD	Trivia	QuAC	QA-SRL	QA-ZRE	MNLI	SNLI
BERT	85.4	72.5	60.0	85.0	88.2	81.1	88.0
ELMo	78.3	57.1	54.3	67.3	88.5	69.1	77.9

Table 3: Results on multiple tasks for BERT and ELMo trained with a random training curriculum for 200,000 iterations.

6 Discussion

Our series of experiments show that while we are making great progress as a field, we are still missing several fundamental components to achieve general linguistic intelligence models. Exist-

ing state-of-the-art models are large parametric deep learning models trained in a discriminative fashion on many (unsupervised and supervised) training examples. They typically include task-specific components to perform multiple tasks, and assume that the data distribution is stationary within a task. While such an approach is reasonable, we see that fine tuning these components still require a fair number of training examples, and that these models are very prone to catastrophic forgetting.

Our results show that we need more sophisticated **transfer and continual learning methods**. For instance, techniques such as elastic weight consolidation (Kirkpatrick et al., 2017) and progress and compress (Schwarz et al., 2018) may hold promise for improving our models’ robustness to forgetting. Another crucial component that is currently underexplored is a **memory module** that rapidly adapts to domain shifts and generalizes across tasks. Recent work on neural cache (Grave et al., 2017) and dynamic evaluation (Krause et al., 2018) provide some evidence of the effectiveness of this approach for language modeling. Investigating how to design similar models that operate beyond the token level and are reusable across tasks is a promising direction.

Learning to learn (i.e., **meta learning**) is another approach to learn models that adapt to new tasks rapidly. In this setup, parameters of the model are trained to ensure that a small number of gradient steps on a new task results in good performance on the task. There has not been much work on meta learning natural language processing models. Standard meta learning methods (Finn et al., 2017; Nichol et al., 2018) require the distribution of tasks that we want the model to adapt to to be known in advance (i.e., we need to know all the tasks that we want to evaluate on). However, since NLP models take the same kind of inputs for all tasks (i.e., a sequence of characters or words), it is conceivable that meta-learned NLP models generalize to new unseen tasks. In particular, our codelength objective offers an intriguing meta learning objective since it quantifies generalization speed.

When presented with multiple training examples from multiple tasks, our experiments also show that **training curriculum** can have a considerable effect on model performance. In a scenario where we can see all the tasks beforehand (i.e., similar to the multitask training setup), designing a curriculum that allows the model to learn better and faster becomes important. Recent progress on methods such as Population Based Training (PBT, Jaderberg et al., 2017) could be used to design a curriculum that improves training. PBT requires a lot of computational resources, but the cost could be amortized if it can be used to establish a fixed schedule that becomes a standard curriculum for training general linguistic intelligence models.

One key reason that our models generalize poorly to other tasks is that they rely on task-specific components (among others). A perfect language model should in theory be able to do any linguistic task (e.g., by formulating the task as a question and querying the language model to generate answers). McCann et al. (2018) proposed such a model that can work on multiple tasks, although the overall performance of their model is still below task specific models. Unsupervised pretraining of language models that are then used as a backbone of task-specific models has also been shown to considerably improve performance on downstream tasks (Radford et al., 2018; Peters et al., 2018; Devlin et al., 2018). Therefore, we believe that continued progress on **generative language models** will drive progress on general linguistic intelligence.

In this paper, we mainly consider models that learn passively by observing environments (i.e., from static corpora) and we evaluate them on sample efficiency, generalization to other tasks, and their ability to avoid catastrophic forgetting. There are other important skills that we do not consider such as robustness to adversarial examples or the ability to understand multiple languages. Jia & Liang (2017) show that reading comprehension models trained on SQuAD are very prone to adversarial attacks, although some adversaries are constructed by shifting the distribution of the inputs in non-trivial but meaning-preserving ways. Improving cross-dataset

transfer inside a single task will likely see this classes of adversaries weakened. Ideally, a general linguistic intelligence model should also be able to provide human-understandable reasoning of its decisions and intentions (interpretability and explainability). We leave investigations of such abilities for future work.

Acknowledgements

We thank Stephen Clark for helpful comments on an earlier draft of this paper.

References

- Charles Beattie, Joel Z. Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Kuttler, Andrew Lefrancq, Simon Green, Víctor Valdes, Amir Sadik, Julian Schrittwieser, Keith Anderson, Sarah York, Max Cant, Adam Cain, Adrian Bolton, Stephen Gaffney, Helen King, Demis Hassabis, Shane Legg, and Stig Petersen. DeepMind Lab. *arXiv preprint*, 2016.
- Leonard Blier and Yann Ollivier. The description length of deep learning models. In *Proc. of NIPS*, 2018.
- Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proc. of EMNLP*, 2015.
- Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint*, 2016.
- Gregory J. Chaitin. On the intelligibility of the universe and the notions of simplicity, complexity and irreducibility. In *Thinking about Godel and Turing: Essays on Complexity, 1970–2007*. World Scientific, 2007.
- Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. QuAC: Question answering in context. In *Proc. of EMNLP*, 2018.
- Andrew M. Dai and Quoc V. Le. Semi-supervised sequence learning. In *Proc. of NIPS*, 2015.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint*, 2018.
- Tim Van Erven, Peter Grunwald, and Steven De Rooij. Catching up faster by switching sooner: A predictive approach to adaptive estimation with an application to the AIC-BIC dilemma. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(3):361–417, 2012.
- Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. of ICML*, 2017.
- Nicholas FitzGerald, Julian Michael, Luheng He, and Luke Zettlemoyer. Large-scale QA-SRL parsing. In *Proc. of ACL*, 2018.
- Robert M French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.
- Edouard Grave, Armand Joulin, and Nicolas Usunier. Improving neural language models with a continuous cache. In *Proc. of ICLR*, 2017.

- Isabelle Guyon, Gavin Cawley, Gideon Dror, and Vincent Lemaire. Results of the active learning challenge. In *Proc. of Workshop on Active Learning and Experimental Design*, 2011.
- 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. Achieving human parity on automatic chinese to english news translation. *arXiv preprint*, 2018.
- Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.
- Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks. *arXiv preprint*, 2017.
- Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proc. of EMNLP*, 2017.
- Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proc. of ACL*, 2017.
- James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114(13):3521–3526, 2017.
- Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. Dynamic evaluation of sequence models. In *Proc. of ICML*, 2018.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Rhinehart, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural Questions: A benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.
- Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *Proc. of CoNLL*, 2017.
- Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. The natural language decathlon: Multitask learning as question answering. *arXiv preprint*, 2018.
- Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation*, 24(92):109–165, 1989.
- Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint*, 2018.
- Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *arXiv preprint*, 2018.

- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proc. of EMNLP*, 2016.
- Jonathan Schwarz, Jelena Luketina, Wojciech M. Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress and compress: A scalable framework for continual learning. In *Proc. of ICML*, 2018.
- Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hananneh Hajishirzi. Bi-directional attention flow for machine comprehension. In *Proc. of ICLR*, 2017.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. of NIPS*, 2017.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint*, 2018.
- Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proc. of NAACL-HLT*, 2018.
- J. Gerard Wolff. Language acquisition, data compression and generalization. *Language & Communication*, 2(1):57–89, 1982.