Contextualized Embeddings

Introduction to Natural Language Processing and Deep Learning

IT-UNIVERSITETET I KØBENHAVN / north

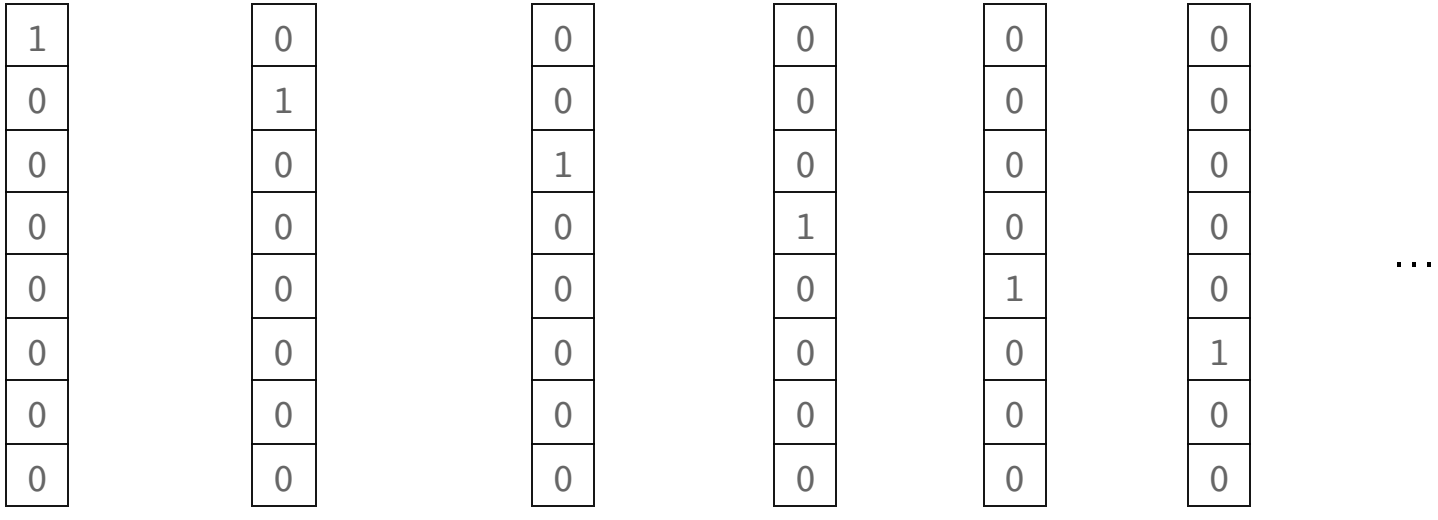
The Story So Far

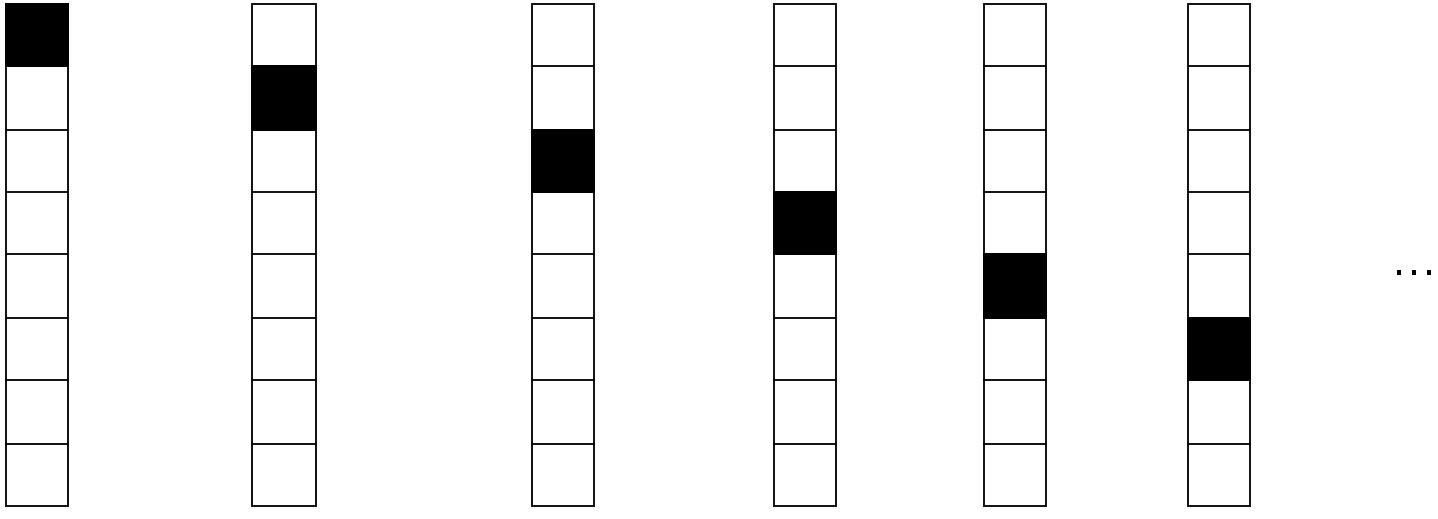
> [The Analytical Engine] might act upon other things besides number, were objects found whose mutual fundamental relations could be expressed by those of the abstract science of operations [...].

– Augusta Ada King, Countess of Lovelace (1842)

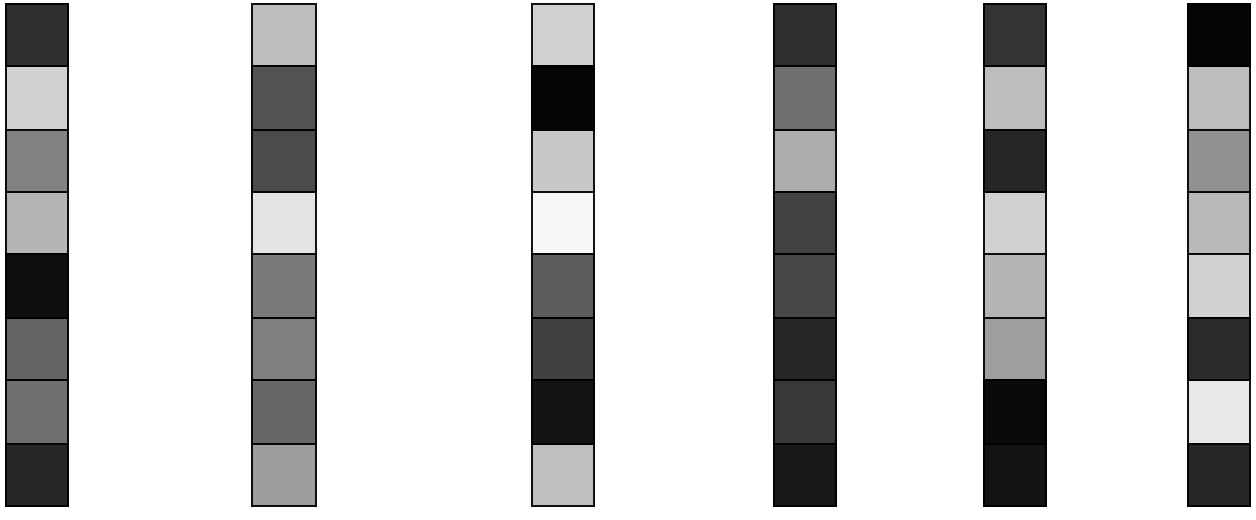
> [The Analytical Engine] might act upon other things besides number, were objects found whose mutual fundamental relations could be expressed by those of the abstract science of operations [...].

– Augusta Ada King, Countess of Lovelace (1842)

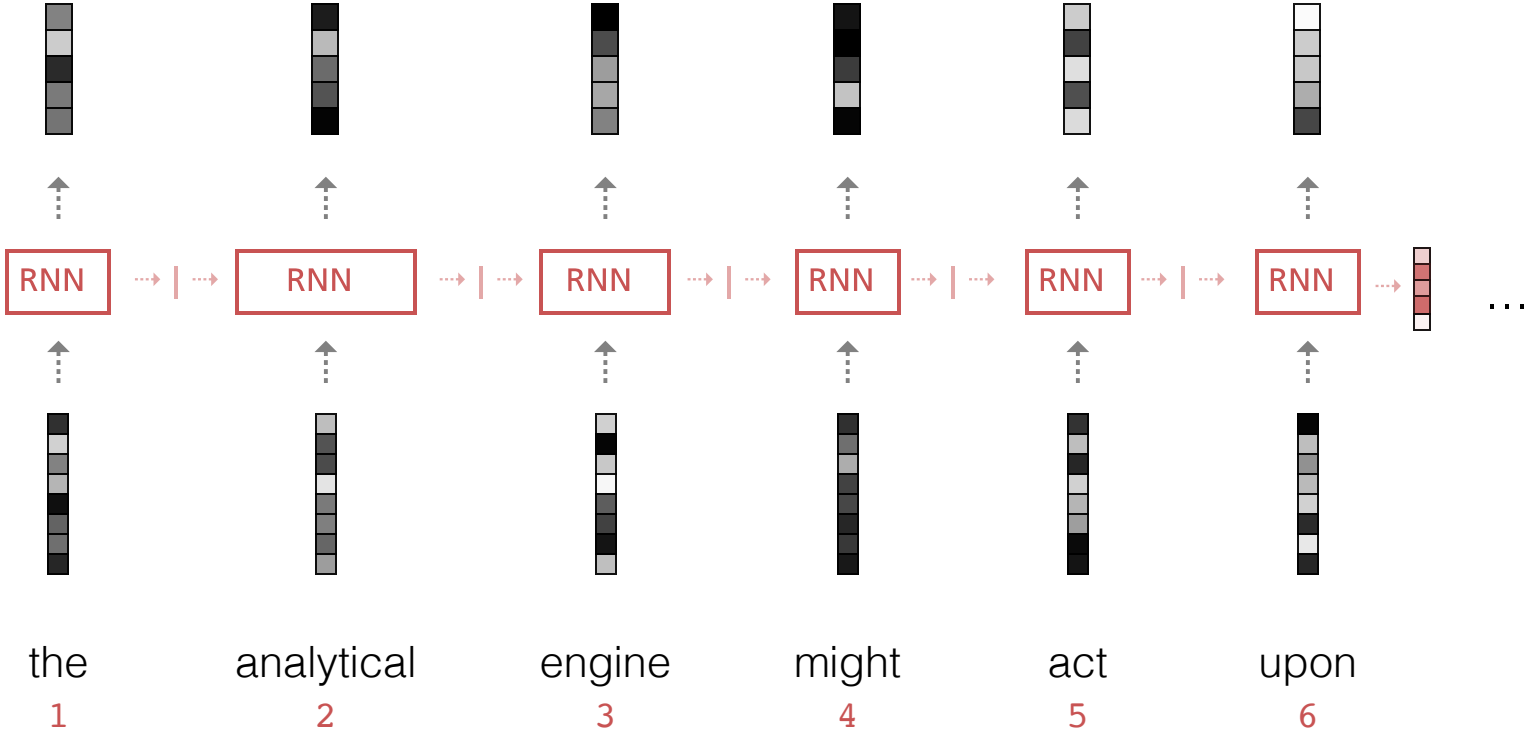

1 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0
...
the 1
analytical 2
engine 3
might 4
act 5
upon 6

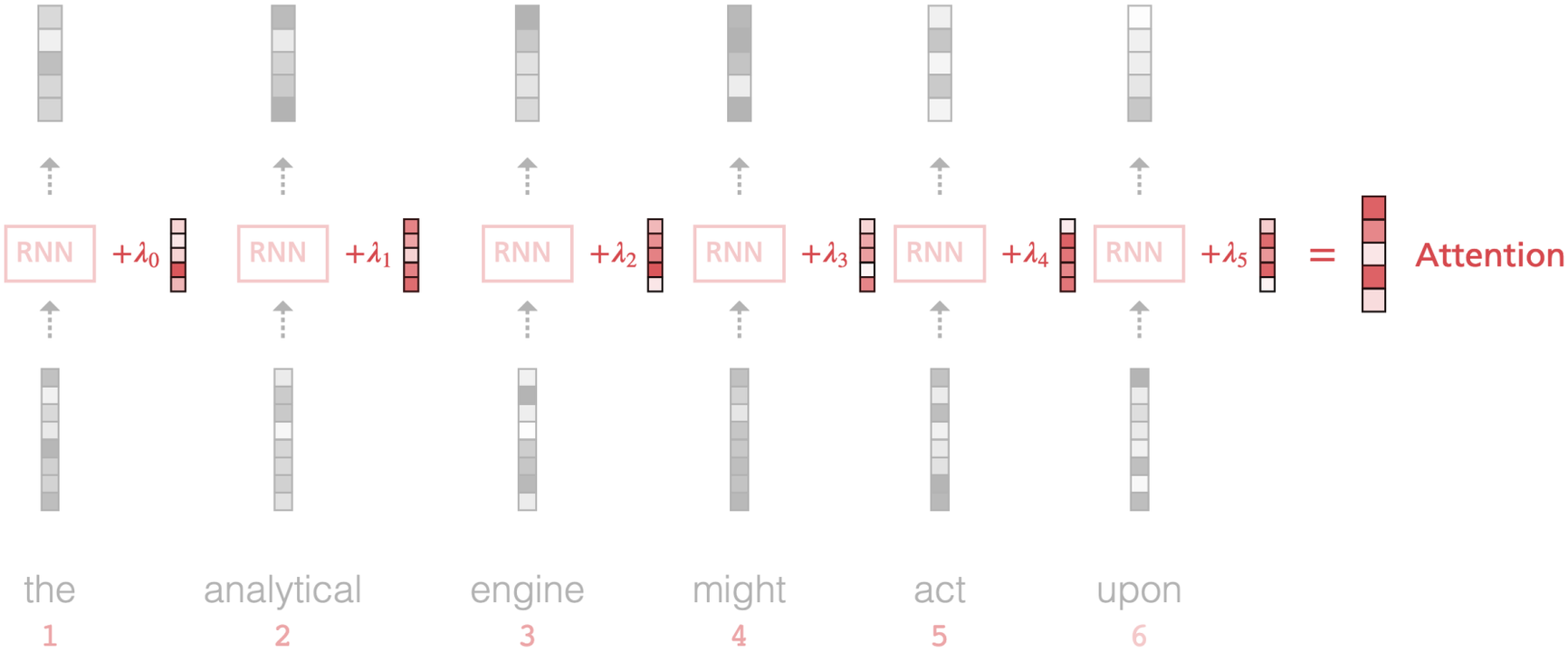


the	analytical	engine	might	act	upon
1	2	3	4	5	6



the 1 analytical 2 engine 3 might 4 act 5 upon 6 ...


RNN
RNN
RNN
RNN
RNN
RNN
...
the
1
analytical
2
engine
3
might
4
act
5
upon
6


RNN
+λ0
RNN
+λ1
RNN
+λ2
RNN
+λ3
RNN
+λ4
RNN
+λ5
=
Attention
the
1
analytical
2
engine
3
might
4
act
5
upon
6

What's missing?

No Context

Don't address me until you've given me the address.

address (verb): speak to (a person or an assembly)
address (noun): place where someone lives or an organization is situated

I'd like to order a Danish in Danish.

Danish (noun): pastry typical to Denmark
Danish (noun): Scandinavian language spoken in Denmark

Elmo is a famous character

Elmo (noun): red muppet (slightly unnerving) from Sesame Street (TV programme)

> Don't address me until you've given me the address.

address (verb): speak to (a person or an assembly)
address (noun): place where someone lives or an organization is situated

> I'd like to order a Danish in Danish.

Danish (noun): pastry typical to Denmark
Danish (noun): Scandinavian language spoken in Denmark

> Elmo is a famous character-based contextual language model.

~~Elmo (noun): red muppet (slightly unnerving) from Sesame Street (TV programme)~~
ELMo (noun): see Peters et al. (2018)

Deep Contextualized Embeddings

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee and L. Zettlemoyer (2018)

Embeddings From Language Models (ELMo)

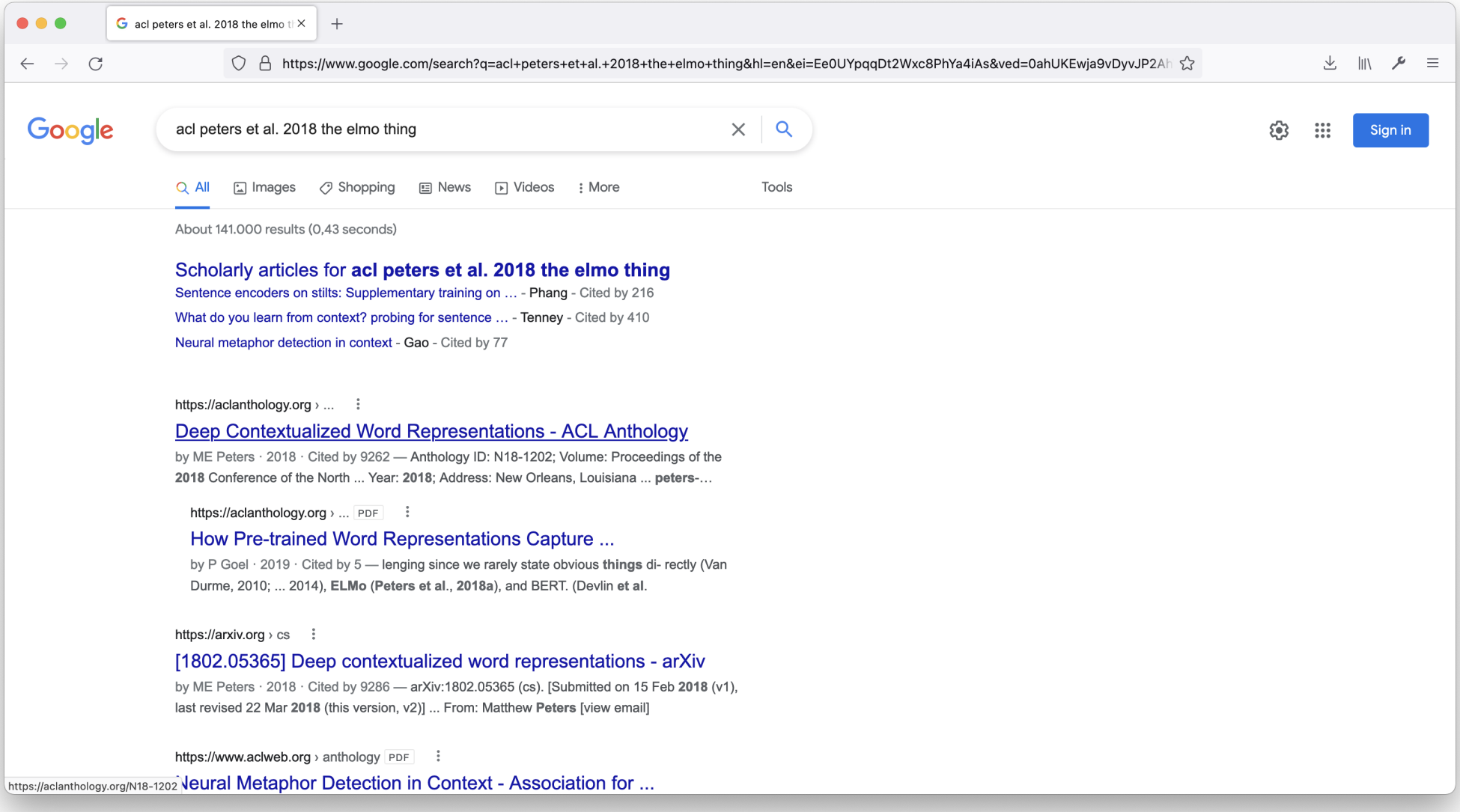
- Started the next iteration of language representations
- Still self-supervised, still pre-trained
- New hotness: token embeddings depend on surrounding context during use

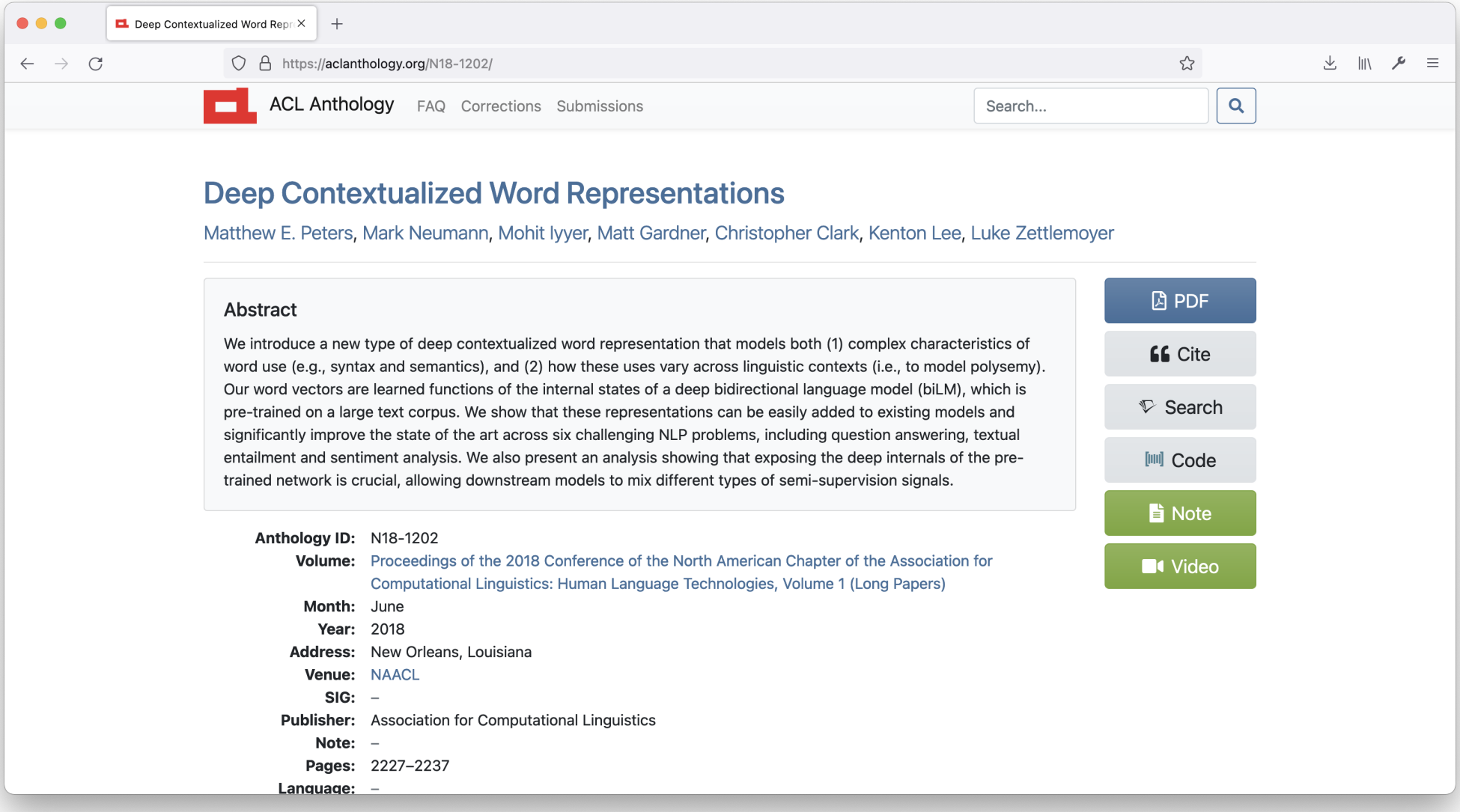
> Our representations differ from traditional word type embeddings in that each token is assigned a representation that is a function of the entire input sentence.

> [...] ELMo representations are deep [...]

– Peters et al. (2018)

In Their Own Words


acl peters et al. 2018 the elmo th
https://www.google.com/search?q=acl+peters+et+al.+2018+the+elmo+thing&hl=en&ei=Ee0UYpqqDt2Wxc8PhYa4iAs&ved=0ahUKEwja9vDyvJP2Ah
Google
acl peters et al. 2018 the elmo thing
Sign in
All
Images
Shopping
News
Videos
More
Tools
About 141.000 results (0,43 seconds)
Scholarly articles for acl peters et al. 2018 the elmo thing
Sentence encoders on stilts: Supplementary training on … - Phang - Cited by 216
What do you learn from context? probing for sentence … - Tenney - Cited by 410
Neural metaphor detection in context - Gao - Cited by 77
https://aclanthology.org › ...
Deep Contextualized Word Representations - ACL Anthology
by ME Peters · 2018 · Cited by 9262 — Anthology ID: N18-1202; Volume: Proceedings of the 2018 Conference of the North ... Year: 2018; Address: New Orleans, Louisiana ... peters-…
https://aclanthology.org › ... PDF
How Pre-trained Word Representations Capture ...
by P Goel · 2019 · Cited by 5 — lenging since we rarely state obvious things di- rectly (Van Durme, 2010; ... 2014), ELMo (Peters et al., 2018a), and BERT. (Devlin et al.
https://arxiv.org › cs
[1802.05365] Deep contextualized word representations - arXiv
by ME Peters · 2018 · Cited by 9286 — arXiv:1802.05365 (cs). [Submitted on 15 Feb 2018 (v1), last revised 22 Mar 2018 (this version, v2)] ... From: Matthew Peters [view email]
https://www.aclweb.org › anthology PDF
Neural Metaphor Detection in Context - Association for ...
https://aclanthology.org/N18-1202


Deep Contextualized Word Repr
https://aclanthology.org/N18-1202/
ACL Anthology
FAQ
Corrections
Submissions
Search...
Deep Contextualized Word Representations
Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer
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.
PDF
Cite
Search
Code
Note
Video
Anthology ID: N18-1202
Volume: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)
Month: June
Year: 2018
Address: New Orleans, Louisiana
Venue: NAACL
SIG: –
Publisher: Association for Computational Linguistics
Note: –
Pages: 2227–2237
Language: –

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

Proceedings of NAACL-HLT 2018, pages 2227–2237
New Orleans, Louisiana, June 1 - 6, 2018.

> Currently, it's like this. However, there's a limitation.

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.

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-

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

2228

Currently, it's like this. However, there's a limitation.

> Currently, it's like this. However, there's a limitation.

> Wouldn't it be nice if...?

Currently, it's like this. However, there's a limitation.

Wouldn't it be nice if...?

Our novel technique does in fact do all the nice things!

Currently, it's like this. However, there's a limitation.

Wouldn't it be nice if...?

Our novel technique does in fact do all the nice things!

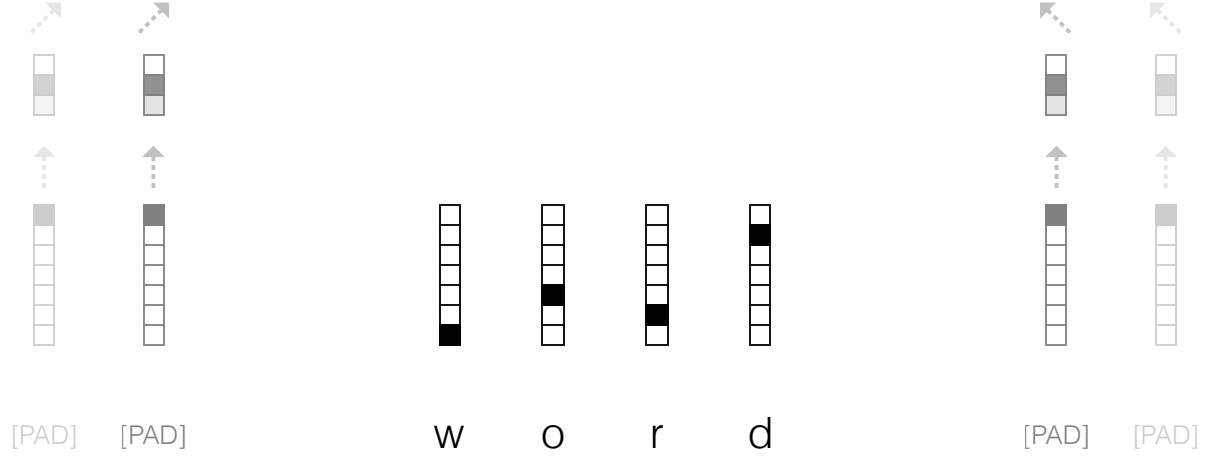
And we'll prove it to you by doing this.

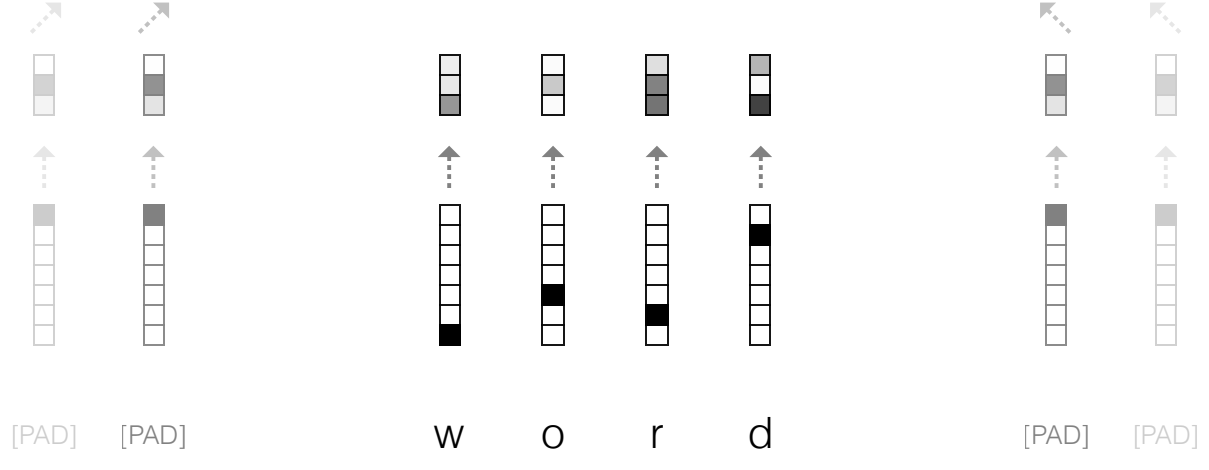
Let's build ELMo and see what it's all about!

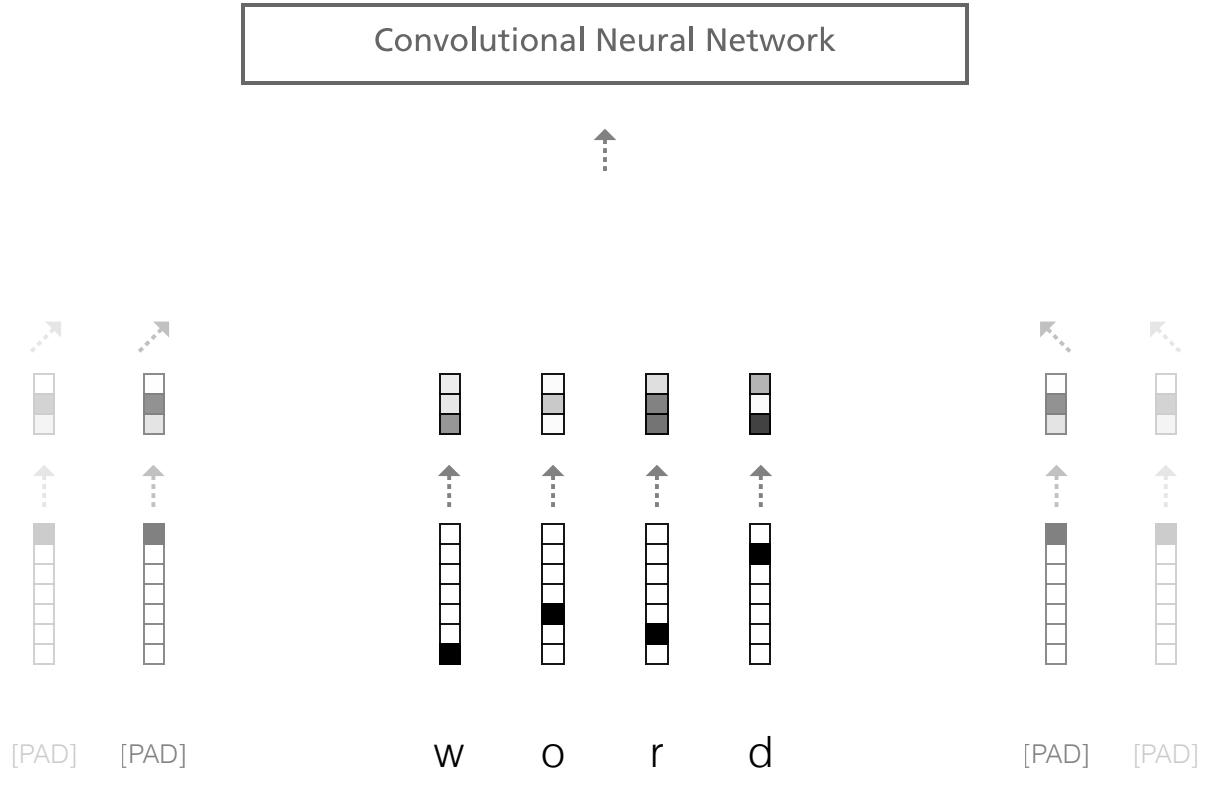
Feeding ELMo

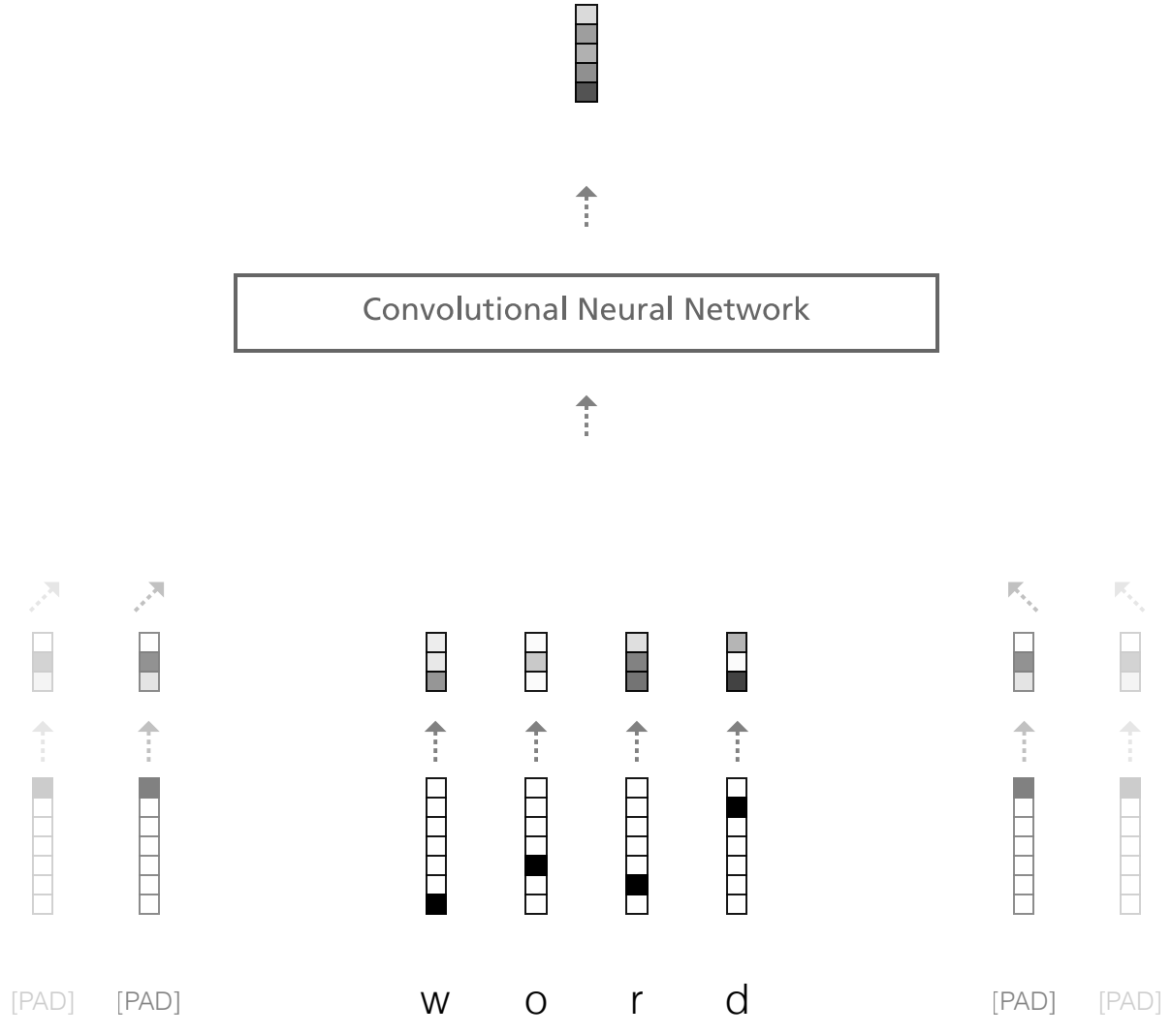
From Characters to Token Embeddings

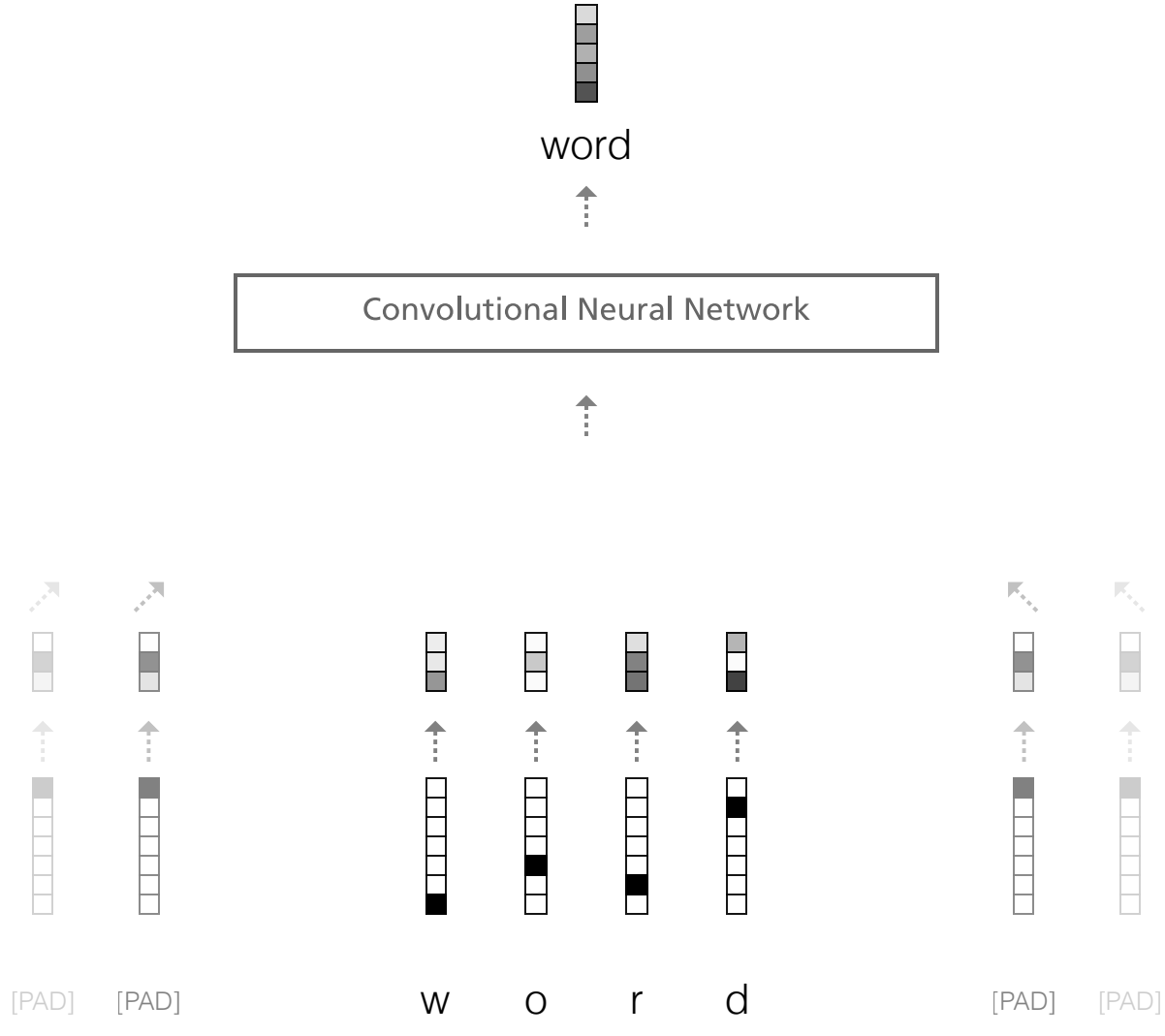

[PAD]
[PAD]
w o r d
[PAD]
[PAD]

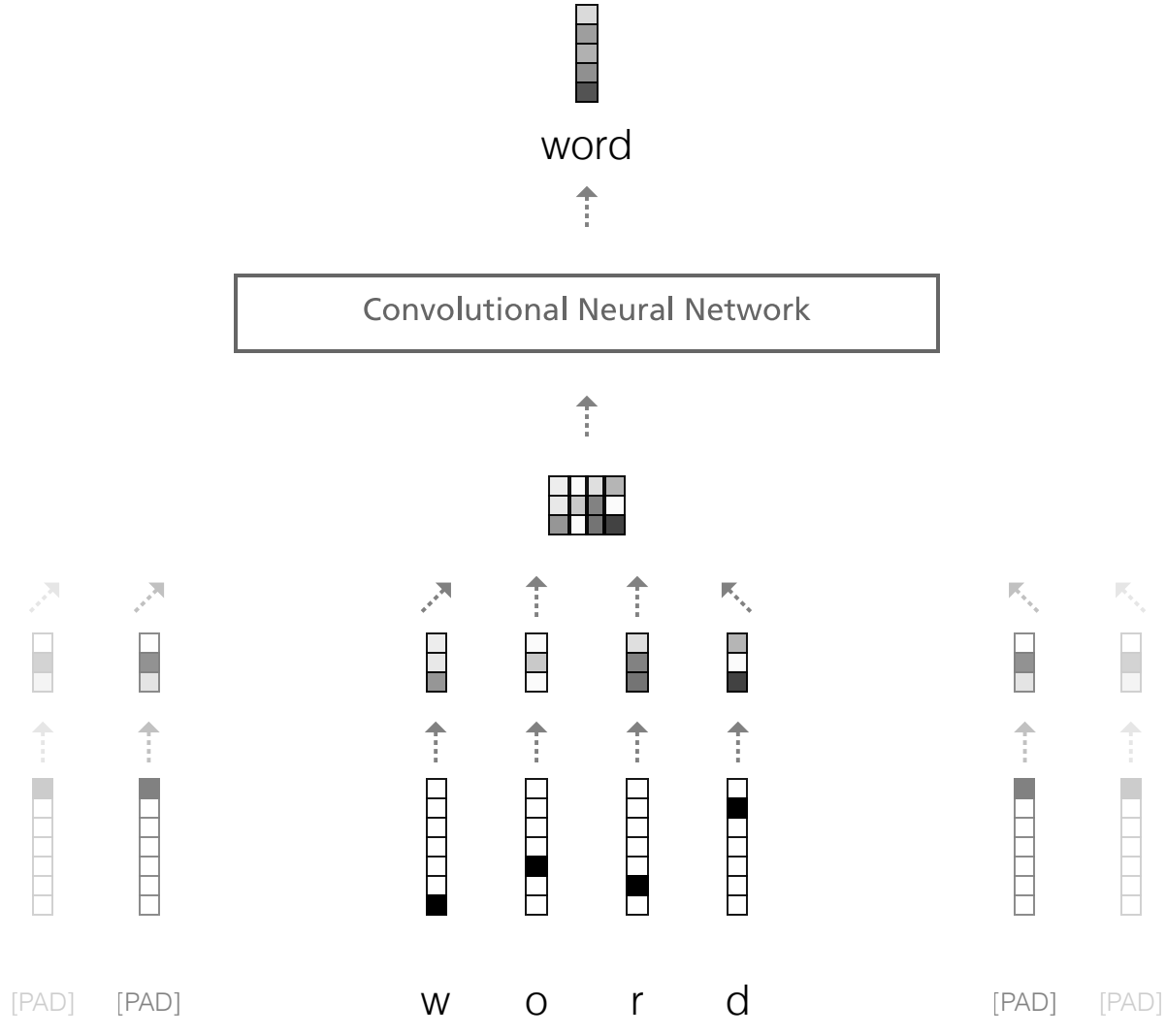

[PAD]
[PAD]
w
o
r
d
[PAD]
[PAD]

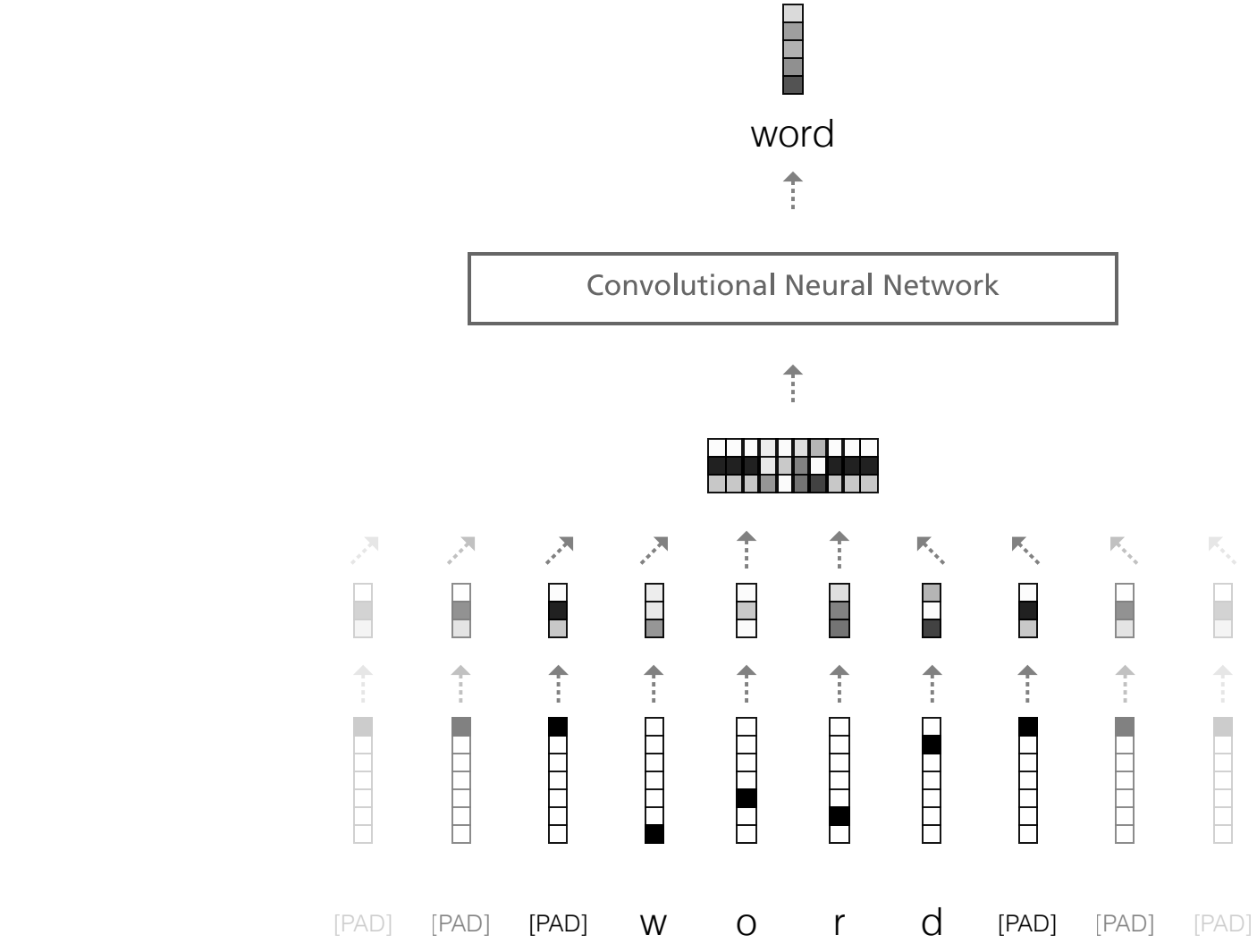

[PAD]
[PAD]
w
o
r
d
[PAD]
[PAD]

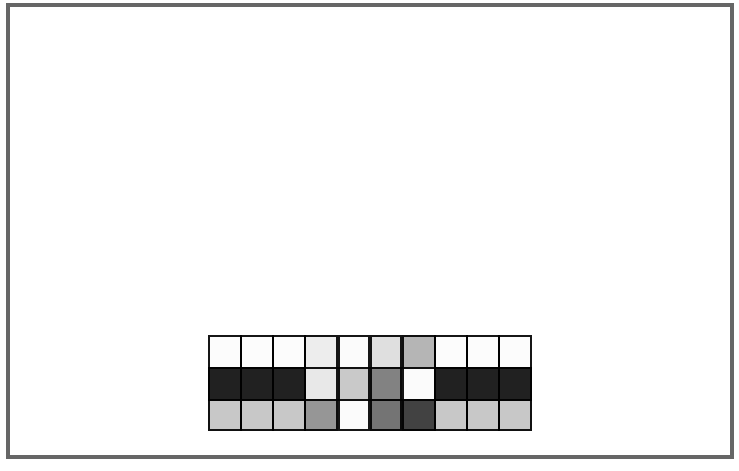

Convolutional Neural Network
[PAD]
[PAD]
w
o
r
d
[PAD]
[PAD]


Convolutional Neural Network
[PAD]
[PAD]
w
o
r
d
[PAD]
[PAD]

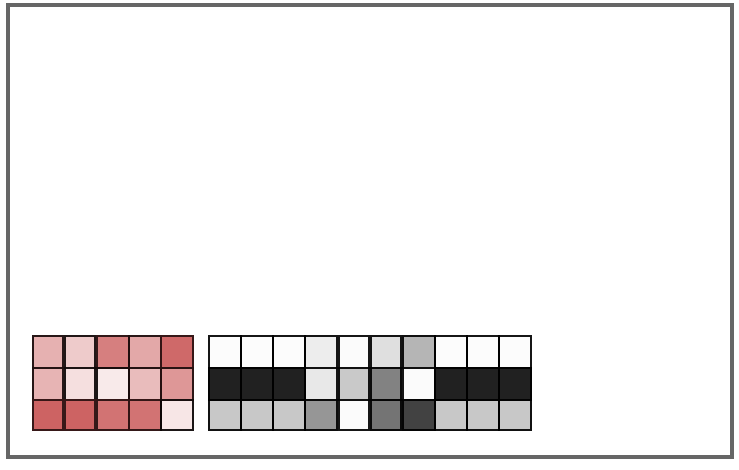

word
Convolutional Neural Network
[PAD]
[PAD]
w
o
r
d
[PAD]
[PAD]


word
Convolutional Neural Network
[PAD]
[PAD]
w
o
r
d
[PAD]
[PAD]

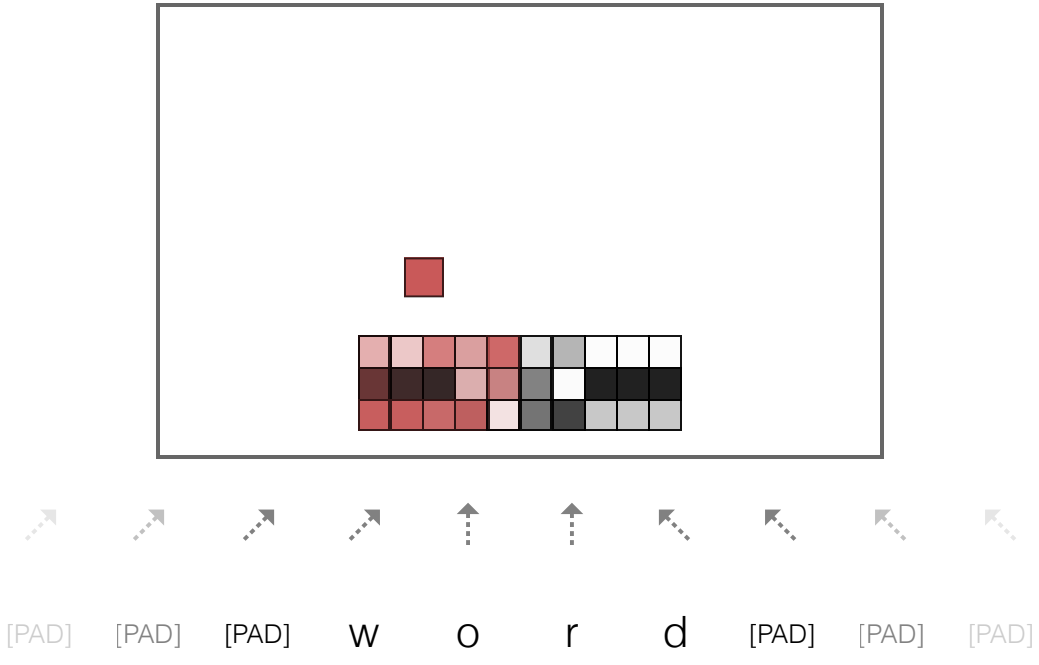

word
Convolutional Neural Network
[PAD]
[PAD]
[PAD]
w
o
r
d
[PAD]
[PAD]
[PAD]

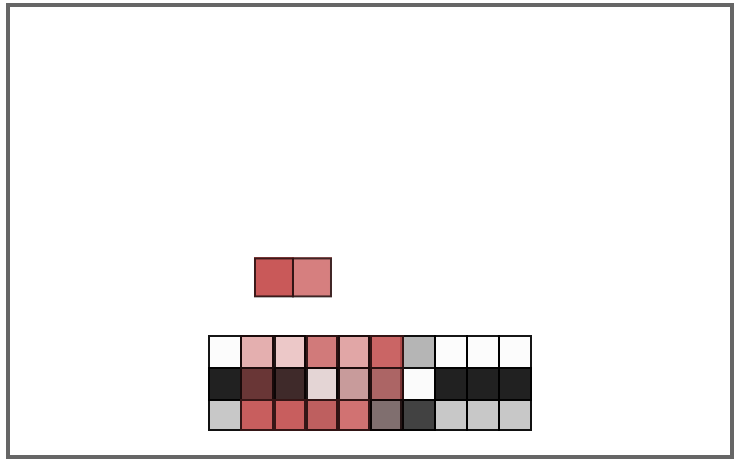


[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]

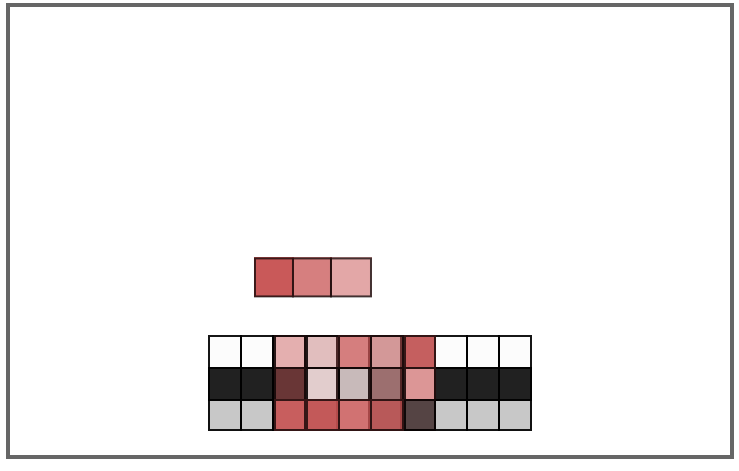


[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]

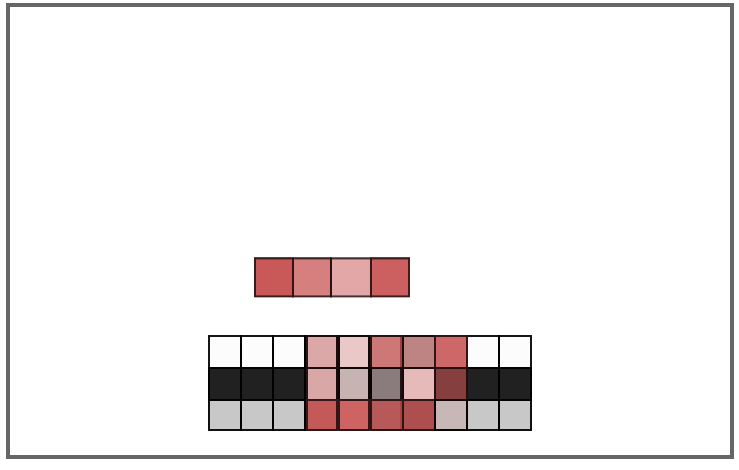

[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]



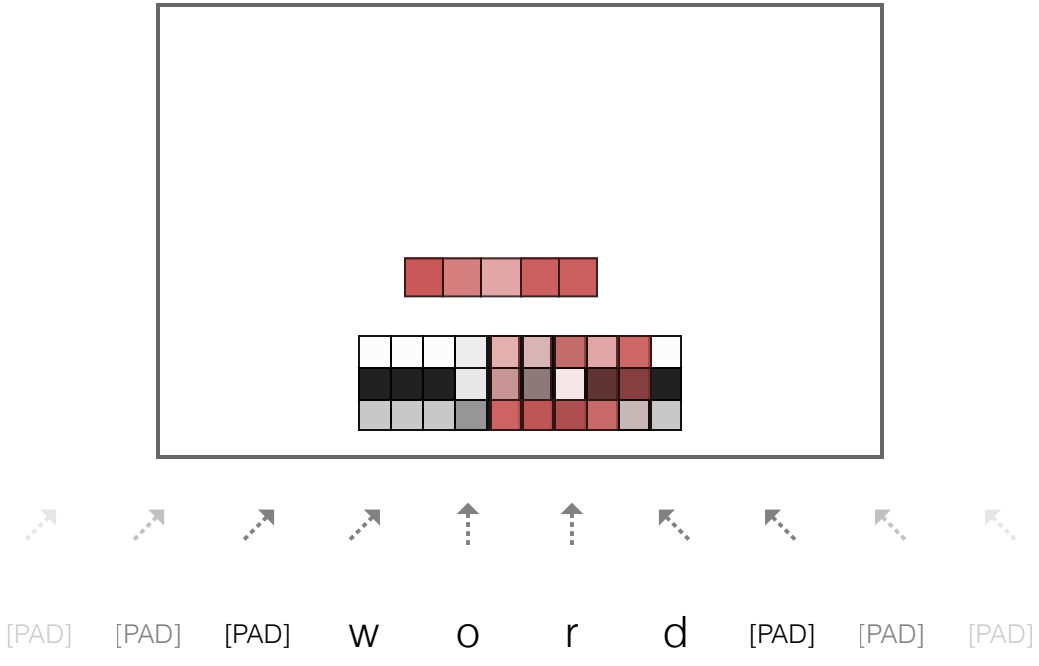
[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]

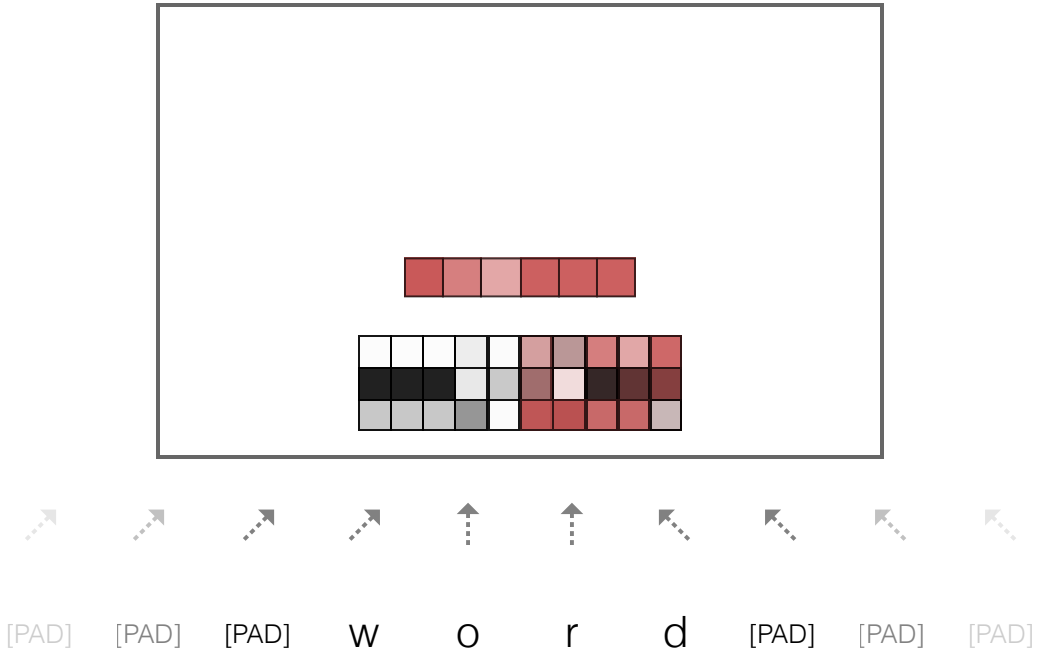


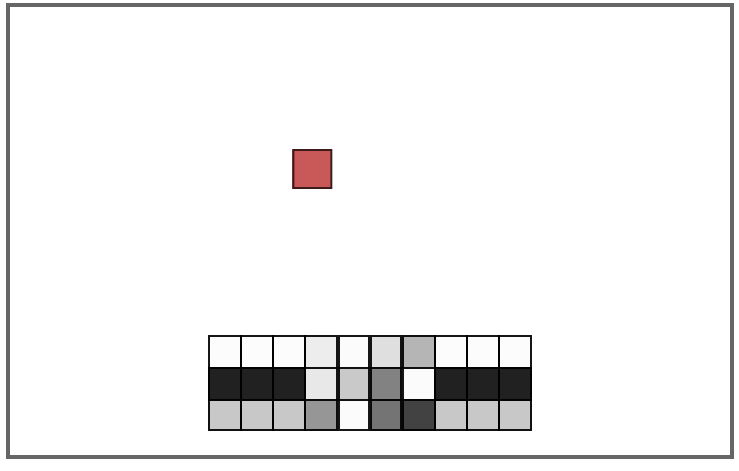
[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]



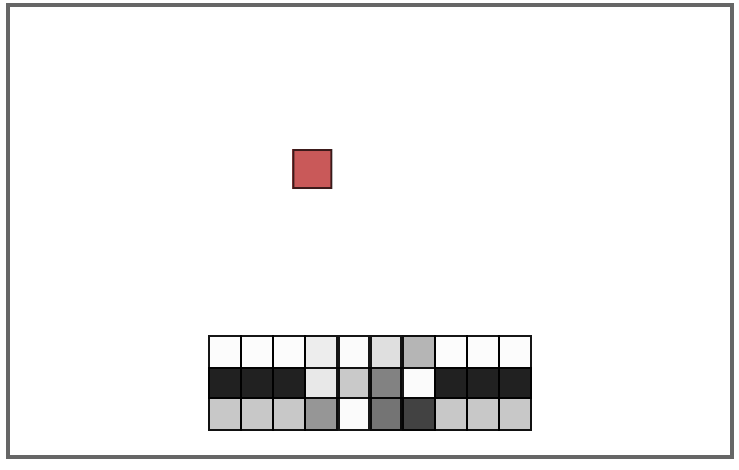
[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]


[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]

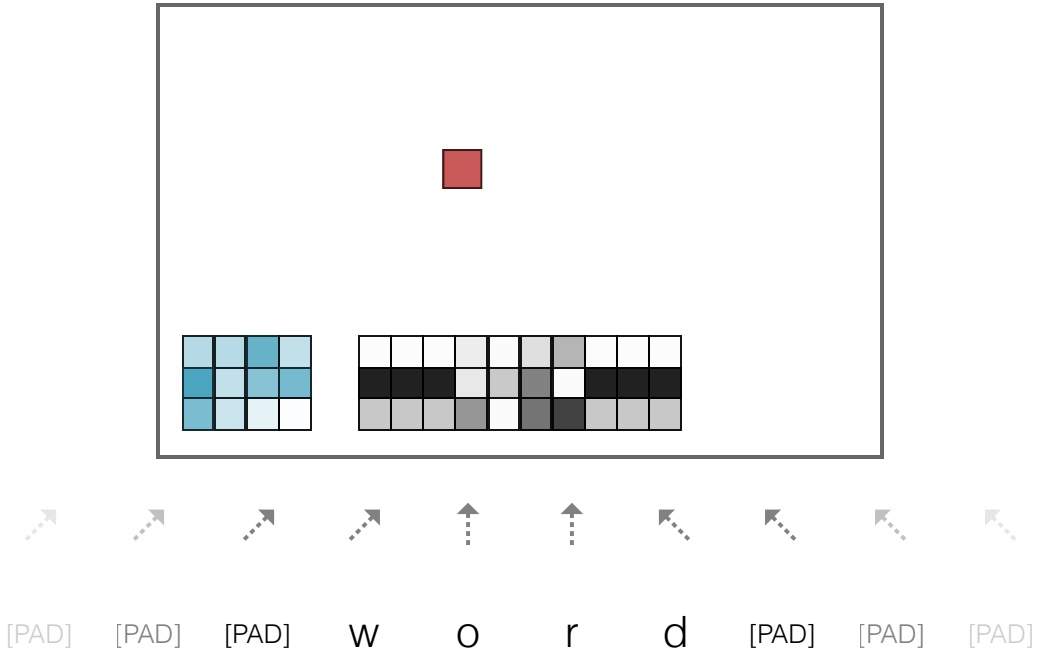

[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]

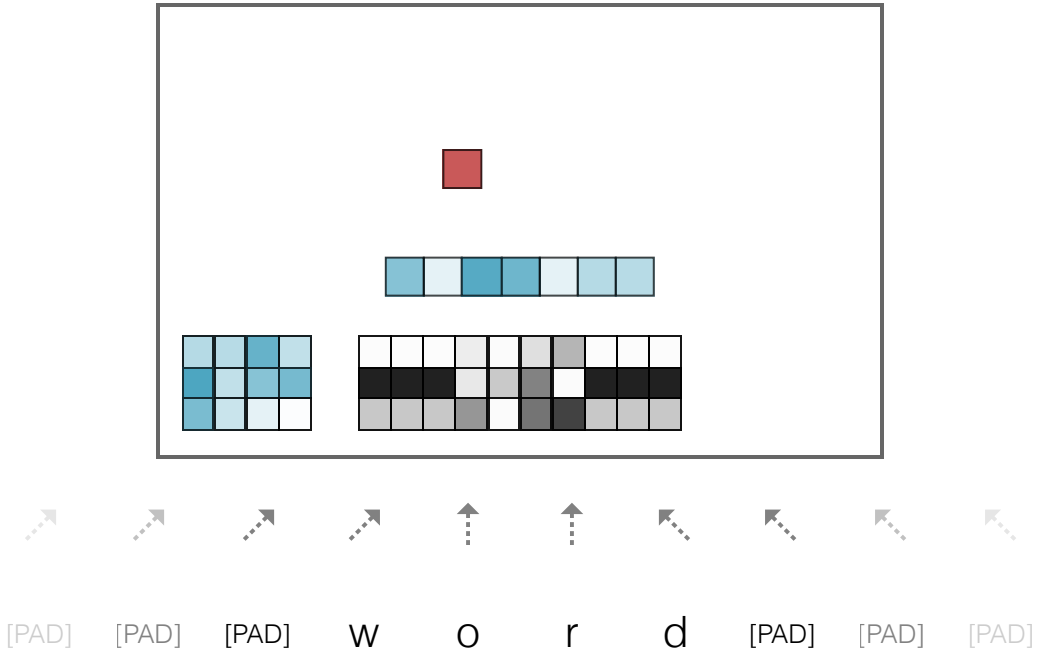


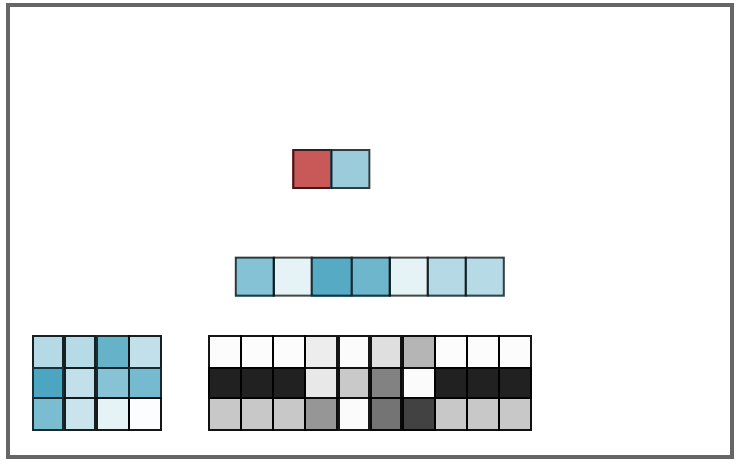
[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]



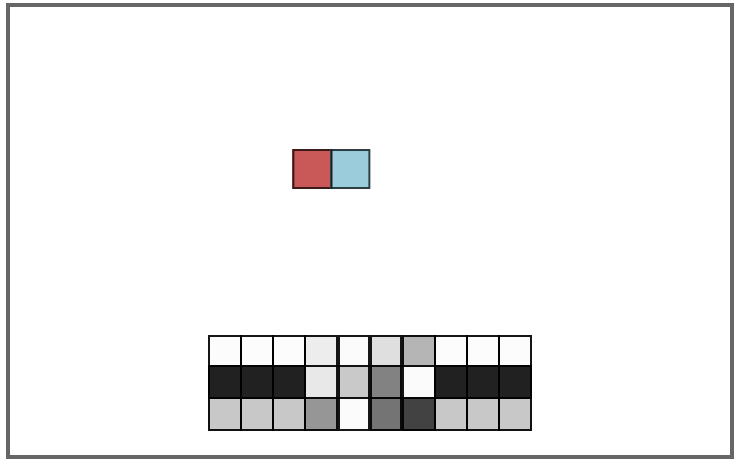
[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]


[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]

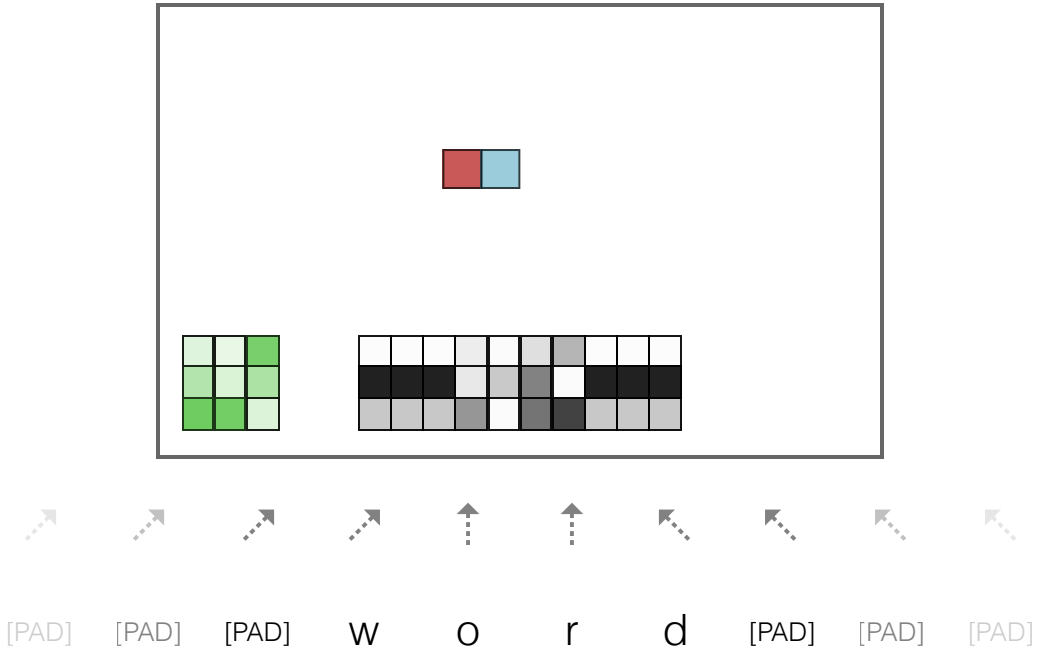

[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]

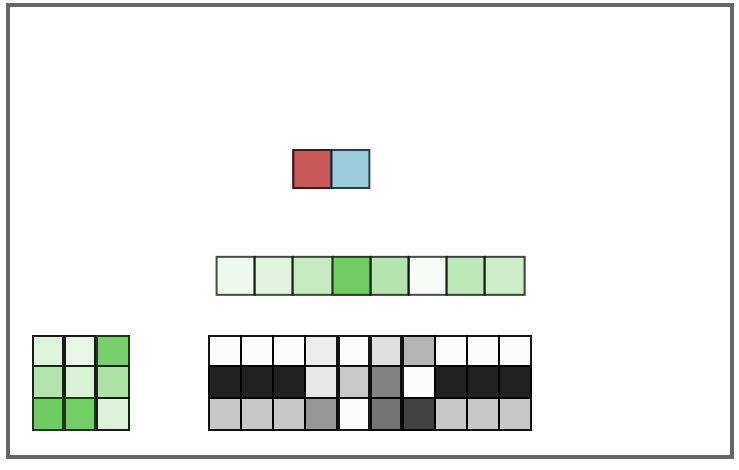


[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]

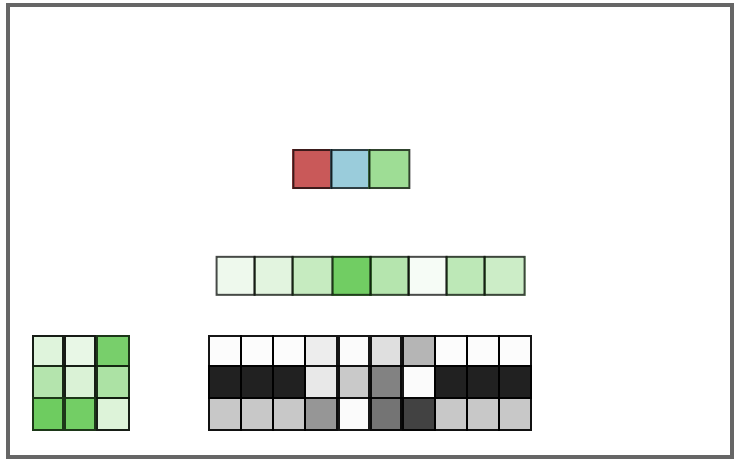


[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]

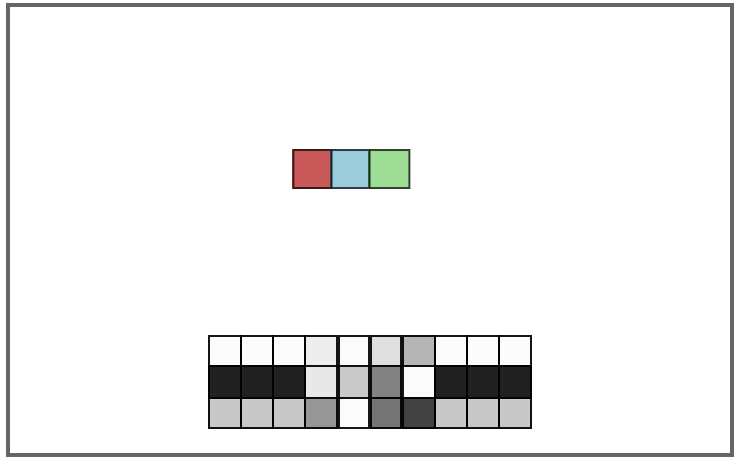

[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]



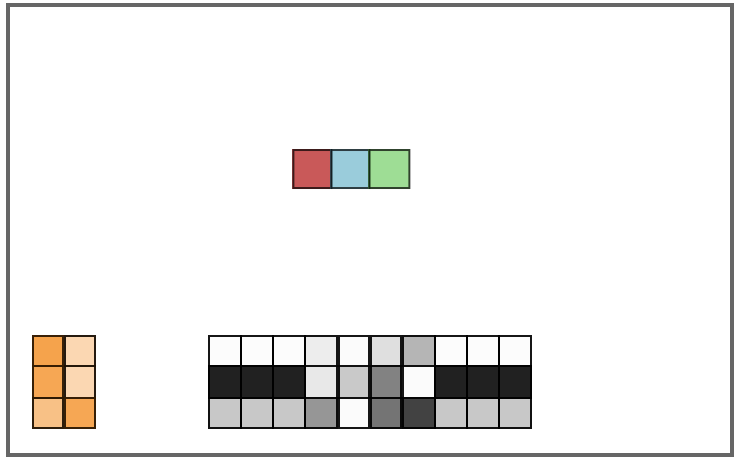
[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]



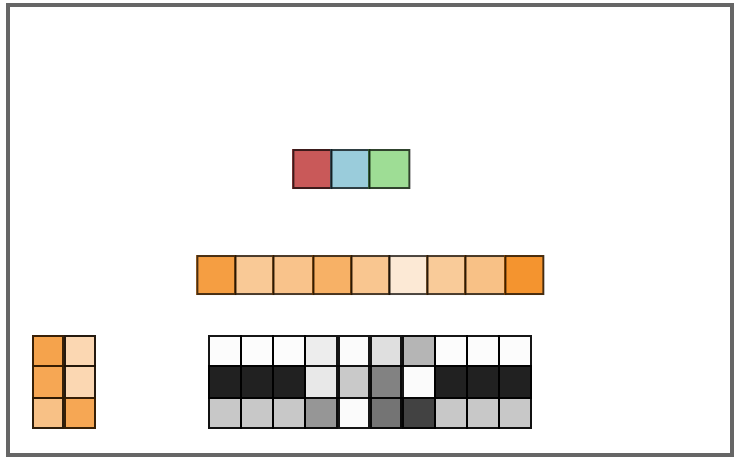
[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]



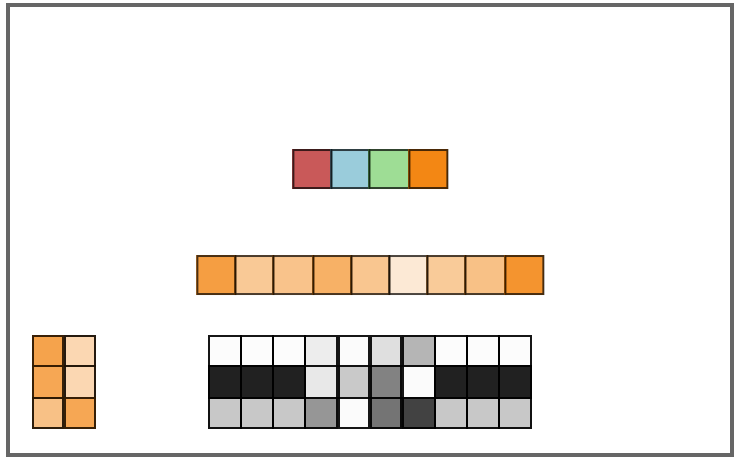
[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]



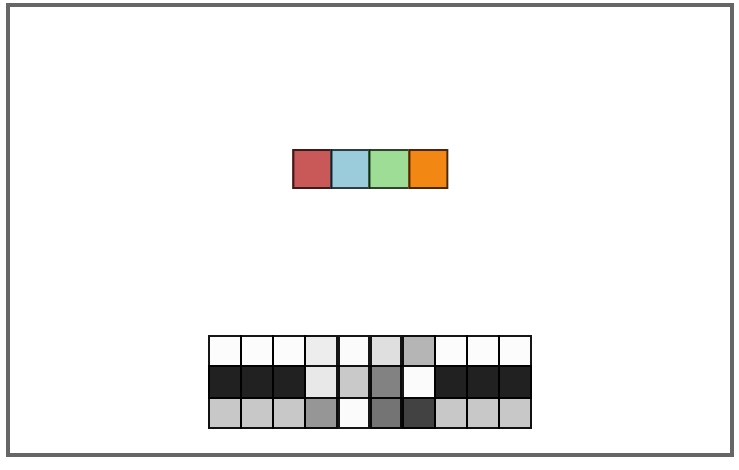
[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]



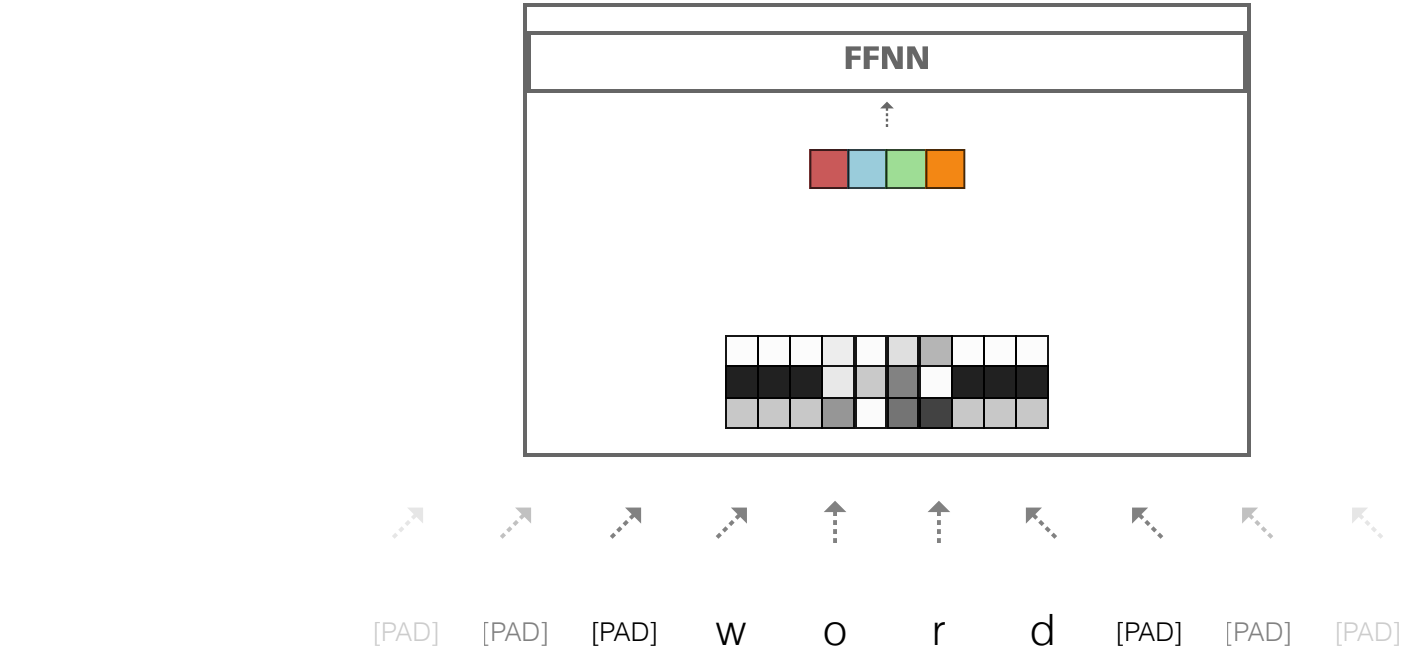
[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]

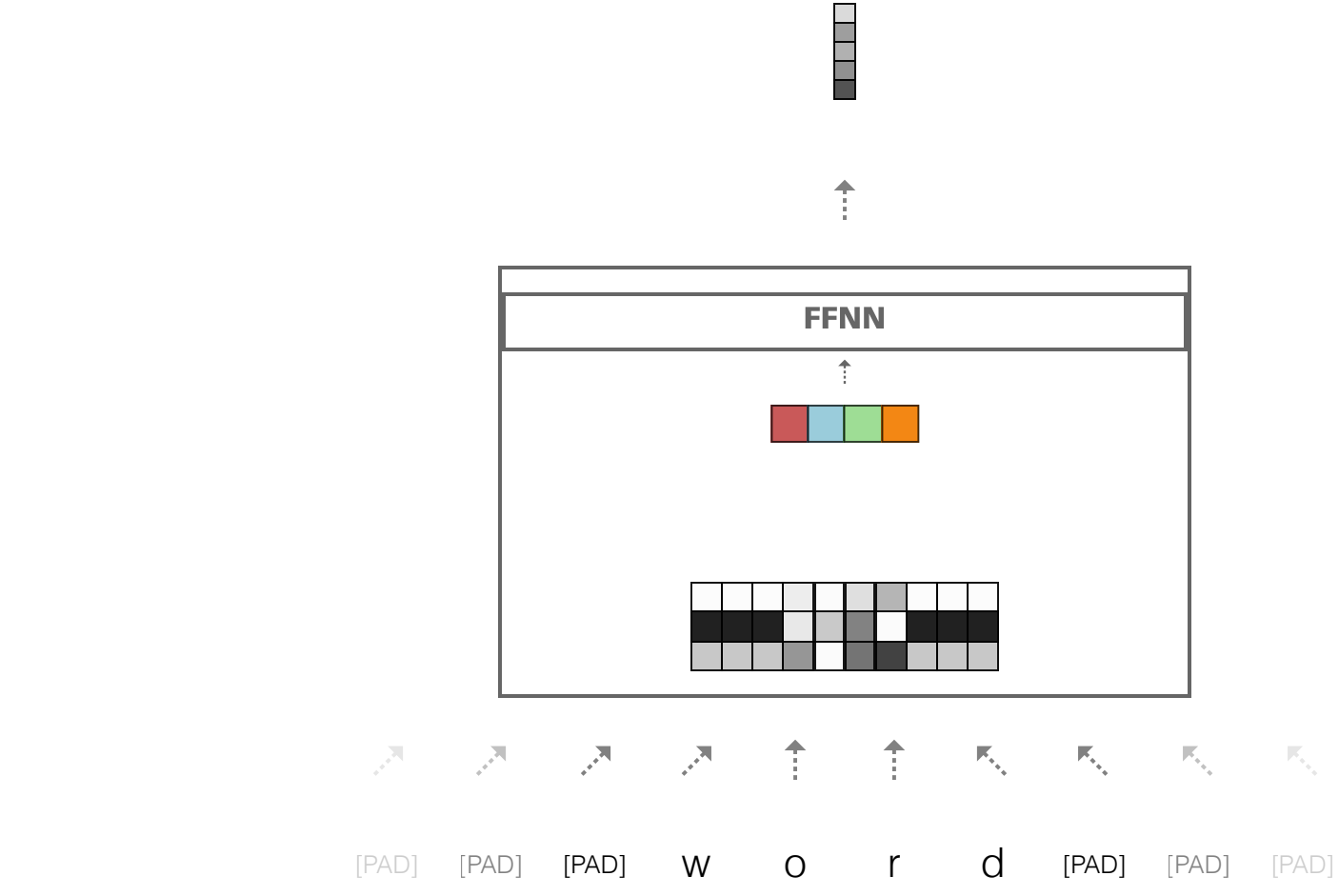


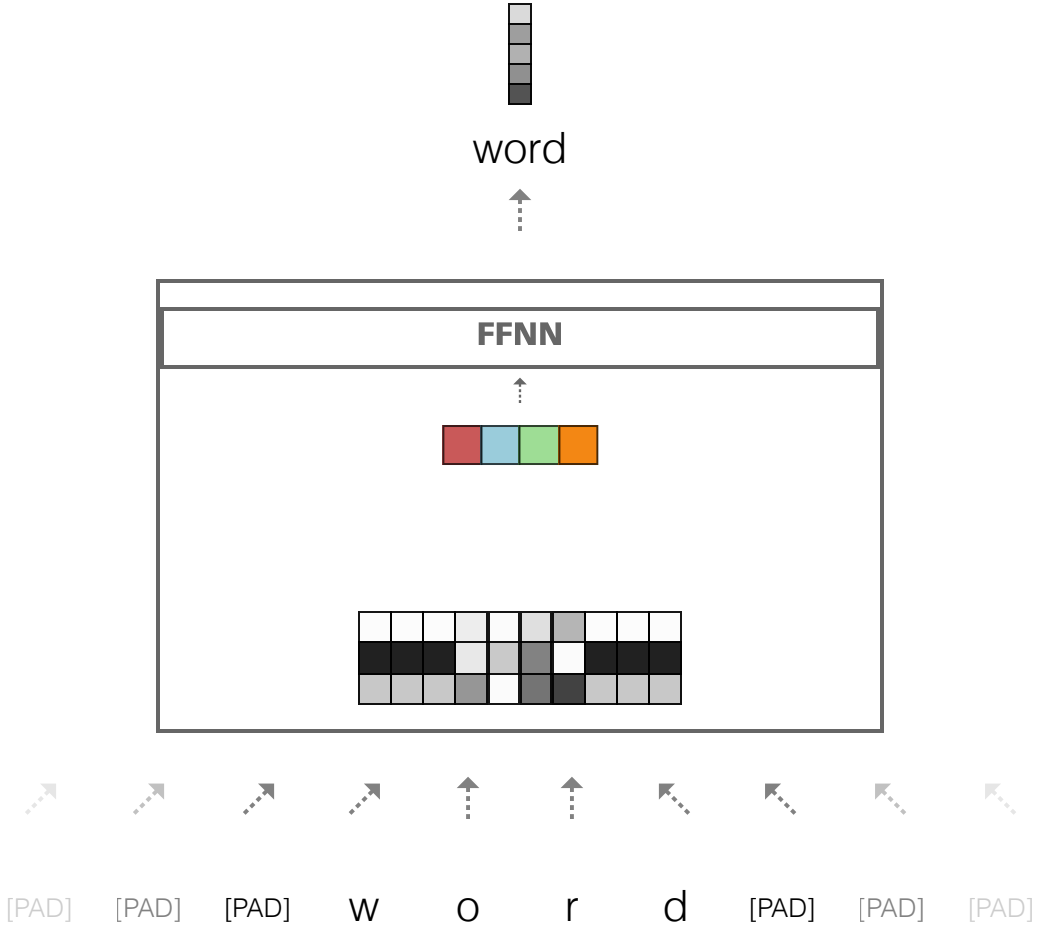
[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]



[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]


FFNN
[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]


FFNN
[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]

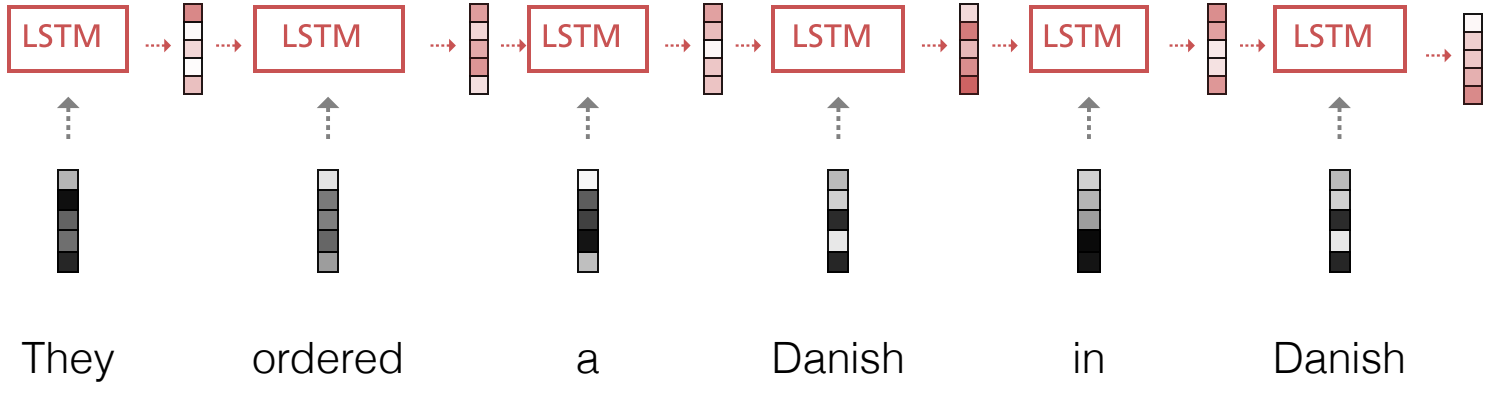

word
FFNN
[PAD] [PAD] [PAD] w o r d [PAD] [PAD] [PAD]

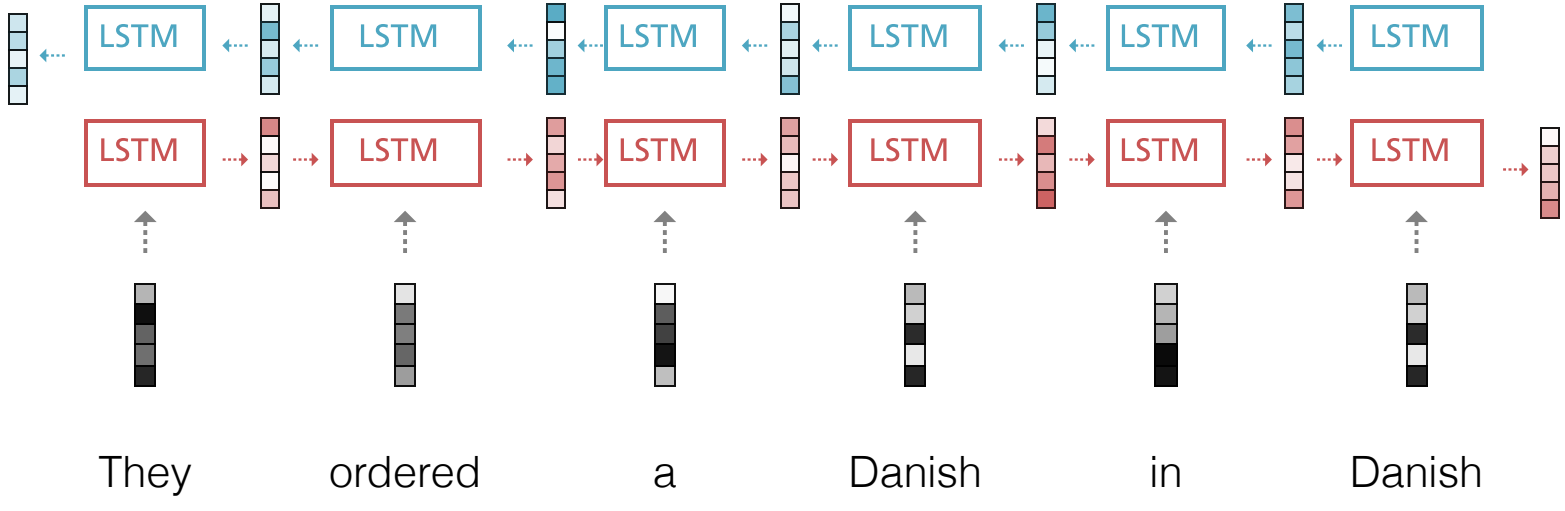

$\mathbf{x}_0^{LM}$
$\mathbf{x}_1^{LM}$
$\mathbf{x}_2^{LM}$
$\mathbf{x}_3^{LM}$
$\mathbf{x}_4^{LM}$
$\mathbf{x}_5^{LM}$
They
ordered
a
Danish
in
Danish

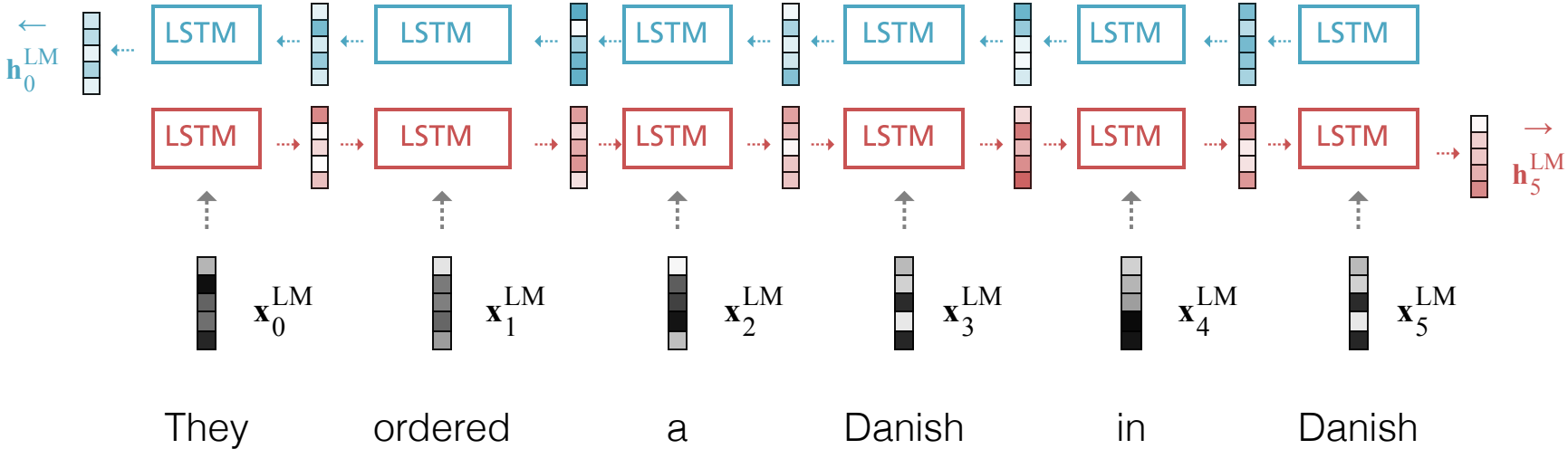
ELMo Needs Some Context

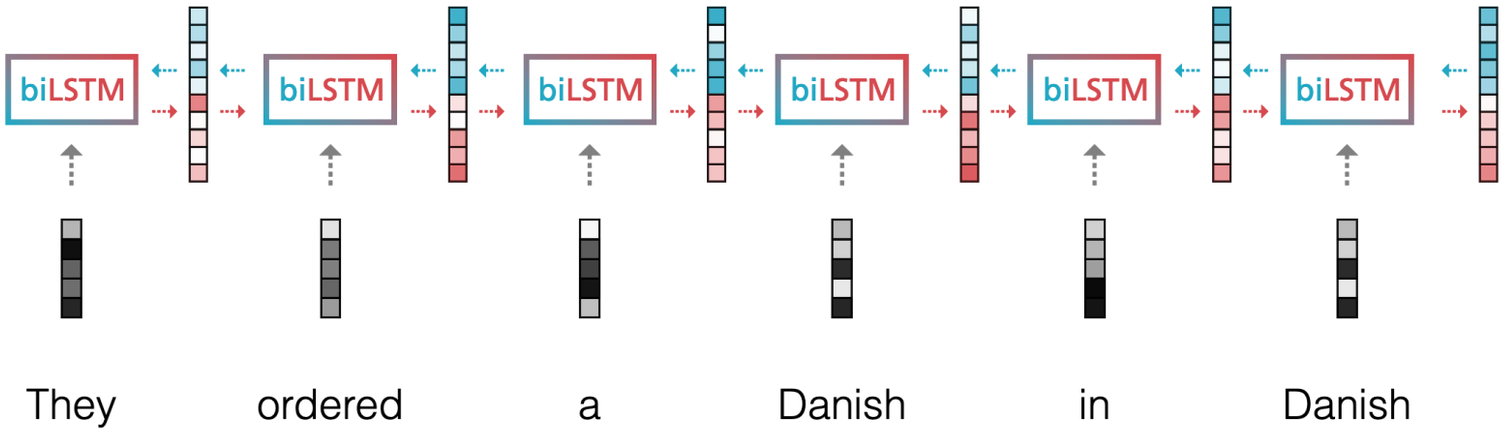
Long Short-term Memory

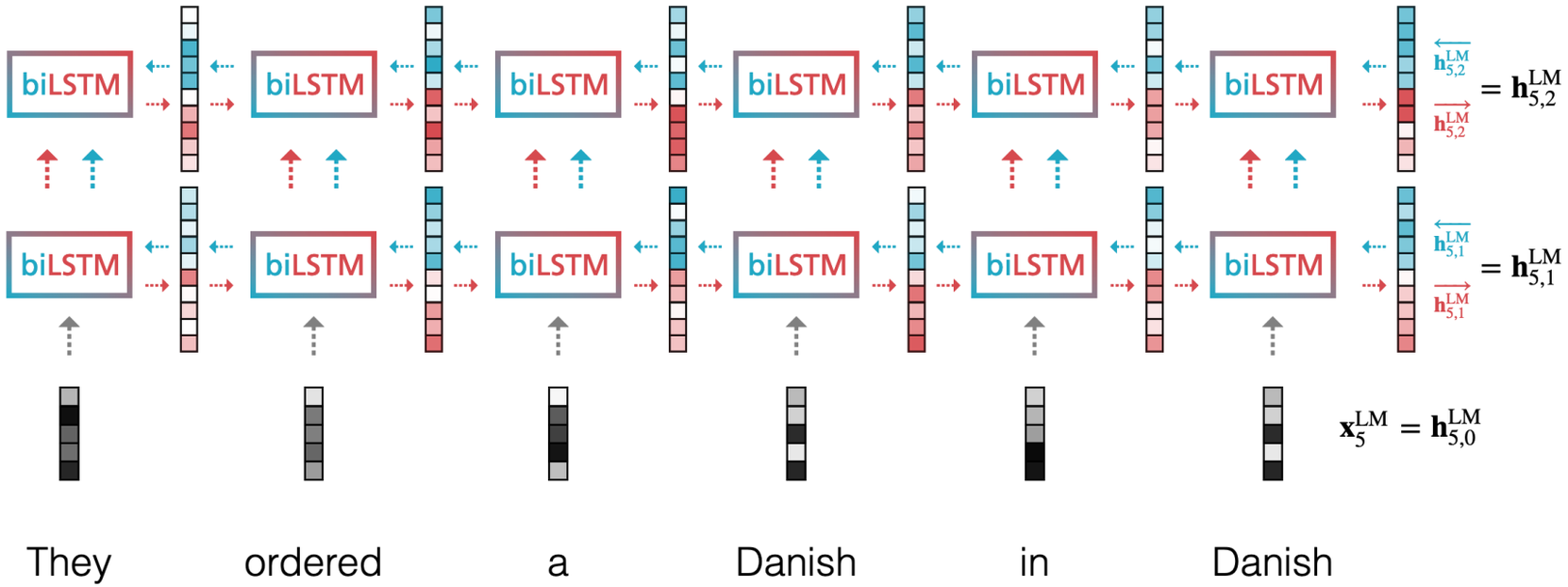
They ordered a Danish in Danish


LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
They
ordered
a
Danish
in
Danish


LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
They
ordered
a
Danish
in
Danish


$\overleftarrow{\mathbf{h}}_0^{LM}$
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
$\overrightarrow{\mathbf{h}}_5^{LM}$
$\mathbf{x}_0^{LM}$
$\mathbf{x}_1^{LM}$
$\mathbf{x}_2^{LM}$
$\mathbf{x}_3^{LM}$
$\mathbf{x}_4^{LM}$
$\mathbf{x}_5^{LM}$
They
ordered
a
Danish
in
Danish


biLSTM
biLSTM
biLSTM
biLSTM
biLSTM
biLSTM
They
ordered
a
Danish
in
Danish


biLSTM
$= \mathbf{h}^{\text{LM}}_{5,2}$
$= \mathbf{h}^{\text{LM}}_{5,1}$
$\mathbf{x}^{\text{LM}}_5 = \mathbf{h}^{\text{LM}}_{5,0}$
They
ordered
a
Danish
in
Danish

$$\mathbf{h}_{k,j}^{\mathrm{LM}} = \begin{cases} \mathbf{x}_k^{\mathrm{LM}} & \text{if } j = 0 \\ \overleftarrow{\quad} \quad \overrightarrow{\quad} & \\ [\mathbf{h}_{k,j}^{\mathrm{LM}}; \mathbf{h}_{k,j}^{\mathrm{LM}}] & \text{else } j \in [1, L] \end{cases}$$

Is this in the paper?

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, $\mathbf{ELMo}_k = E(R_k; \Theta_e)$. 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:

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

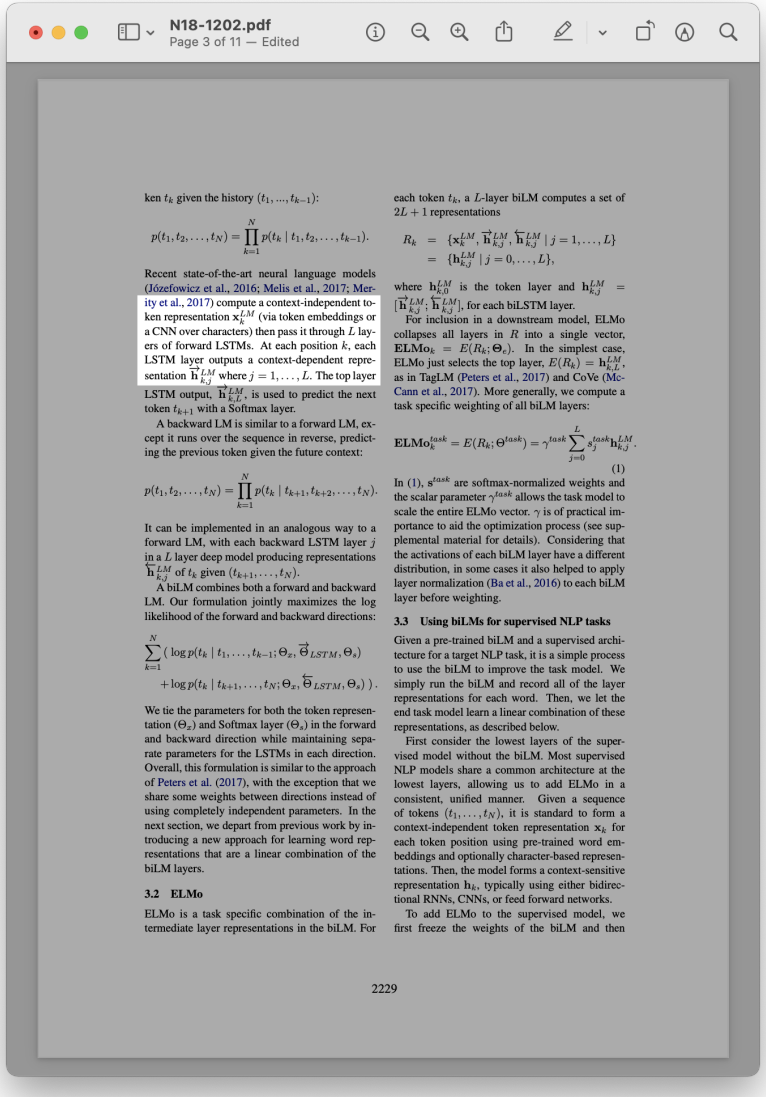
In (1), $\mathbf{s}^{task}$ are softmax-normalized weights and the scalar parameter γ^{task} allows the task model to scale the entire ELMo vector. γ is of practical importance to aid the optimization process (see supplemental material for details). Considering that the activations of each biLM layer 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



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, $\mathbf{ELMo}_k = E(R_k; \Theta_e)$. 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.

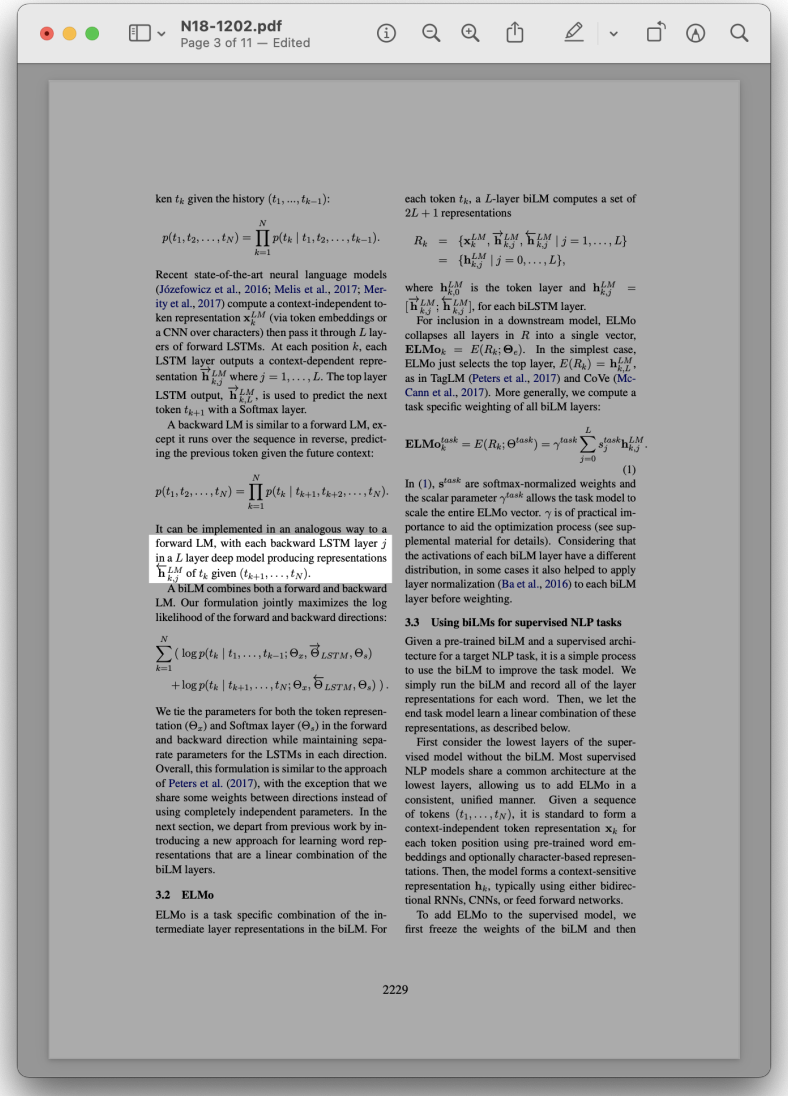
First consider the lowest layers of the supervised model without the biLM. Most supervised NLP models share a common architecture at the lowest layers, allowing us to add ELMo in a consistent, unified manner. Given a sequence of tokens $(t_1, \ldots, t_N)$, it is standard to form a context-independent 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

Non-contextual representations be like $\mathbf{x}_k^{\mathrm{LM}}$.

$\rightarrow$

LSTM goes $\mathbf{h}_{k,j}^{\mathrm{LM}}$.



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, $\mathbf{ELMo}_k = E(R_k; \Theta_e)$. 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.

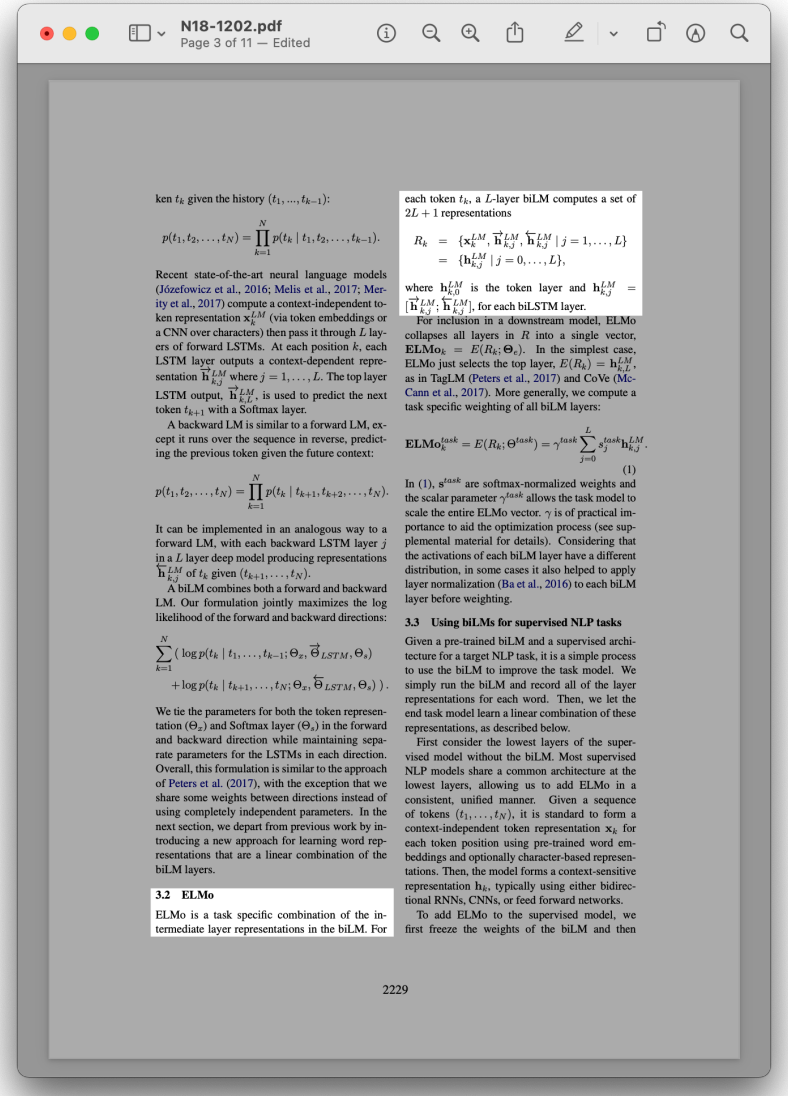
First consider the lowest layers of the supervised model without the biLM. Most supervised NLP models share a common architecture at the lowest layers, allowing us to add ELMo in a consistent, unified manner. Given a sequence of tokens $(t_1, \ldots, t_N)$, it is standard to form a context-independent 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

Non-contextual representations be like $\mathbf{x}_k^{\mathrm{LM}}$.

LSTM goes $\overrightarrow{\mathbf{h}}_{k,j}^{\mathrm{LM}}$.

BiLSTM also goes $\overleftarrow{\mathbf{h}}_{k,j}^{\mathrm{LM}}$.



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:

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

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, $\mathbf{ELMo}_k = E(R_k; \Theta_e)$. 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.

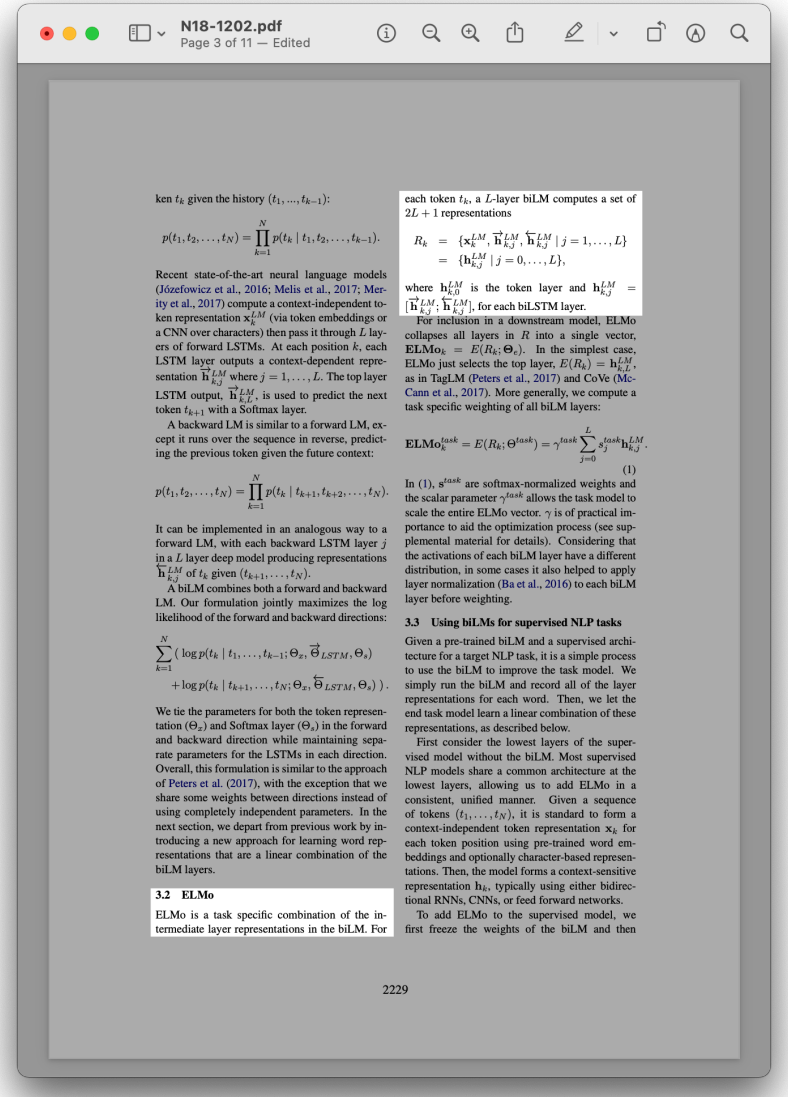
To add ELMo to the supervised model, we first freeze the weights of the biLM and then

Non-contextual representations be like $\mathbf{x}_k^{\mathrm{LM}}$.

LSTM goes $\overrightarrow{\mathbf{h}}_{k,j}^{\mathrm{LM}}$.

BiLSTM also goes $\overleftarrow{\mathbf{h}}_{k,j}^{\mathrm{LM}}$.

Putting it all together, t_k becomes R_k.



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, $\mathbf{ELMo}_k = E(R_k; \Theta_e)$. 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

Non-contextual representations be like $\mathbf{x}_k^{\mathrm{LM}}$.

LSTM goes $\overrightarrow{\mathbf{h}}_{k,j}^{\mathrm{LM}}$.

BiLSTM also goes $\overleftarrow{\mathbf{h}}_{k,j}^{\mathrm{LM}}$.

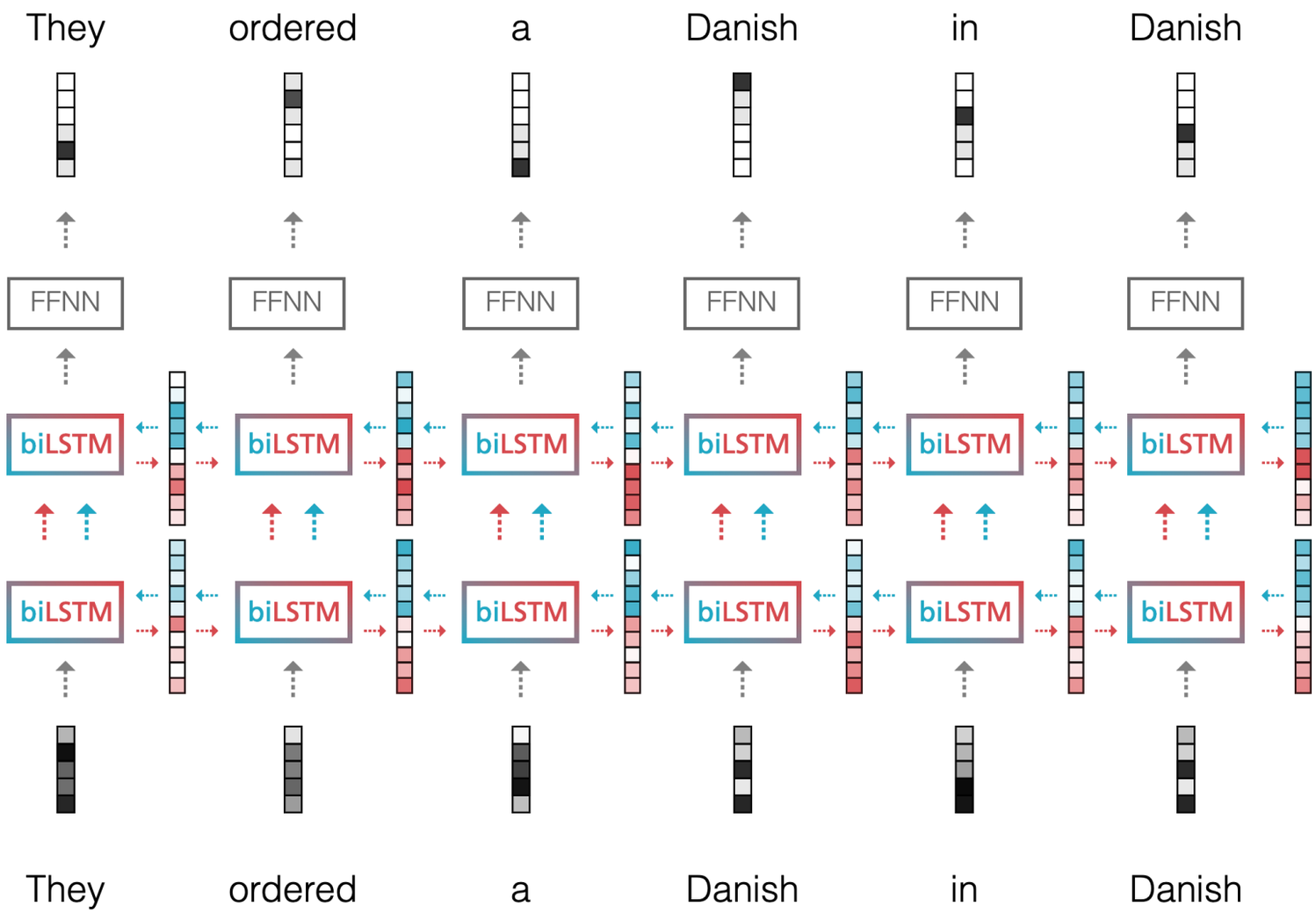
Putting it all together, t_k becomes R_k.

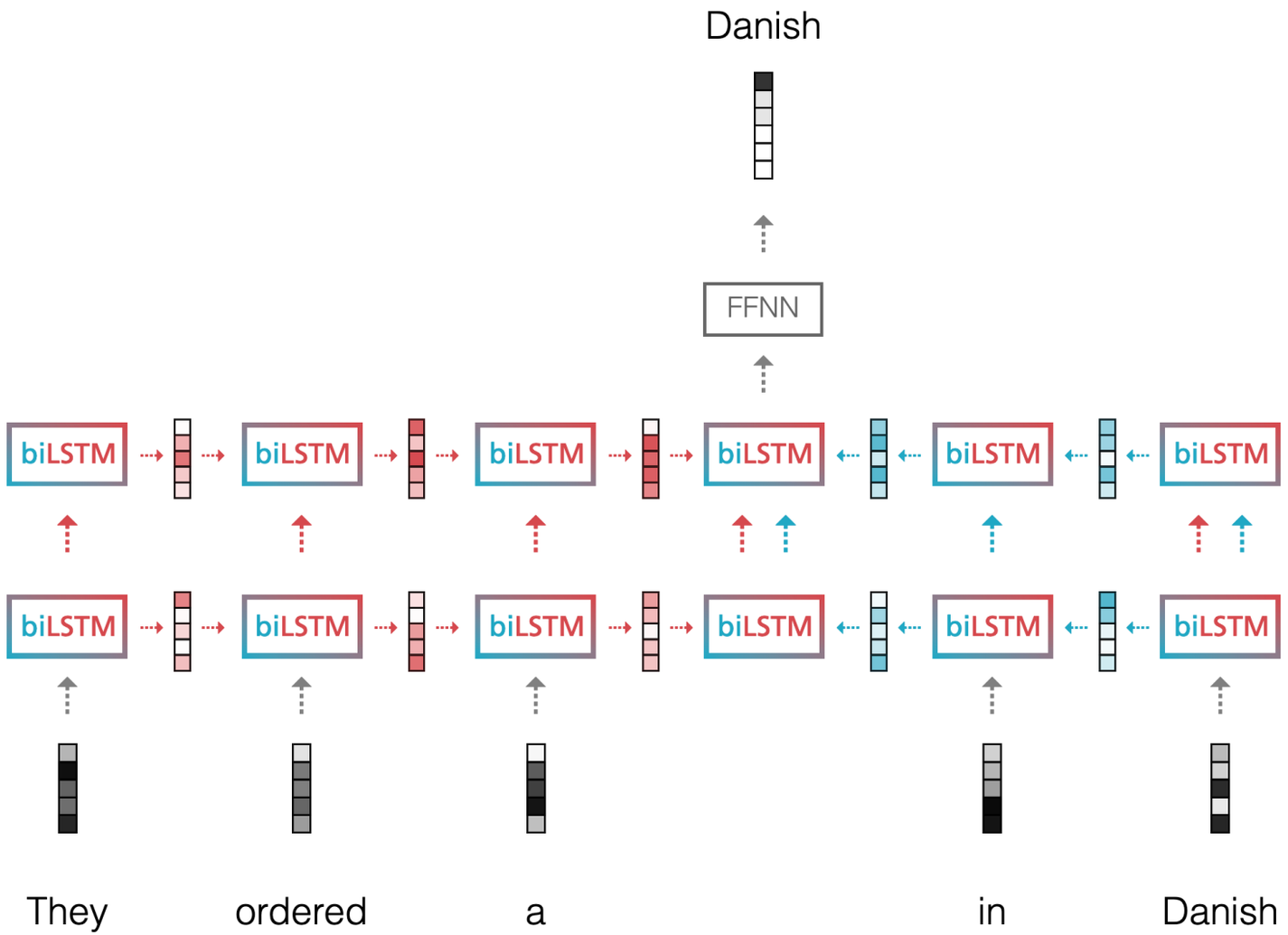
Clearly define each component of your model!
(Figures can help a lot.)

Training ELMo

Language Modeling Objective

- Word2Vec trains word by word
  - Uses context during training
  - No context during use
- ELMo trains on word sequences
  - Predict each token using its context
  - Softmax layer applied to the top layer's output
  - Uses sequence as input during use


They
ordered
a
Danish
in
Danish
FFNN
FFNN
FFNN
FFNN
FFNN
FFNN
biLSTM
biLSTM
biLSTM
biLSTM
biLSTM
biLSTM
biLSTM
biLSTM
biLSTM
biLSTM
biLSTM
biLSTM
They
ordered
a
Danish
in
Danish


Danish
FFNN
biLSTM
biLSTM
biLSTM
biLSTM
biLSTM
biLSTM
biLSTM
biLSTM
biLSTM
biLSTM
biLSTM
biLSTM
They
ordered
a
in
Danish

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

Did we get that right?

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, $\mathbf{ELMo}_k = E(R_k; \Theta_e)$. 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:

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

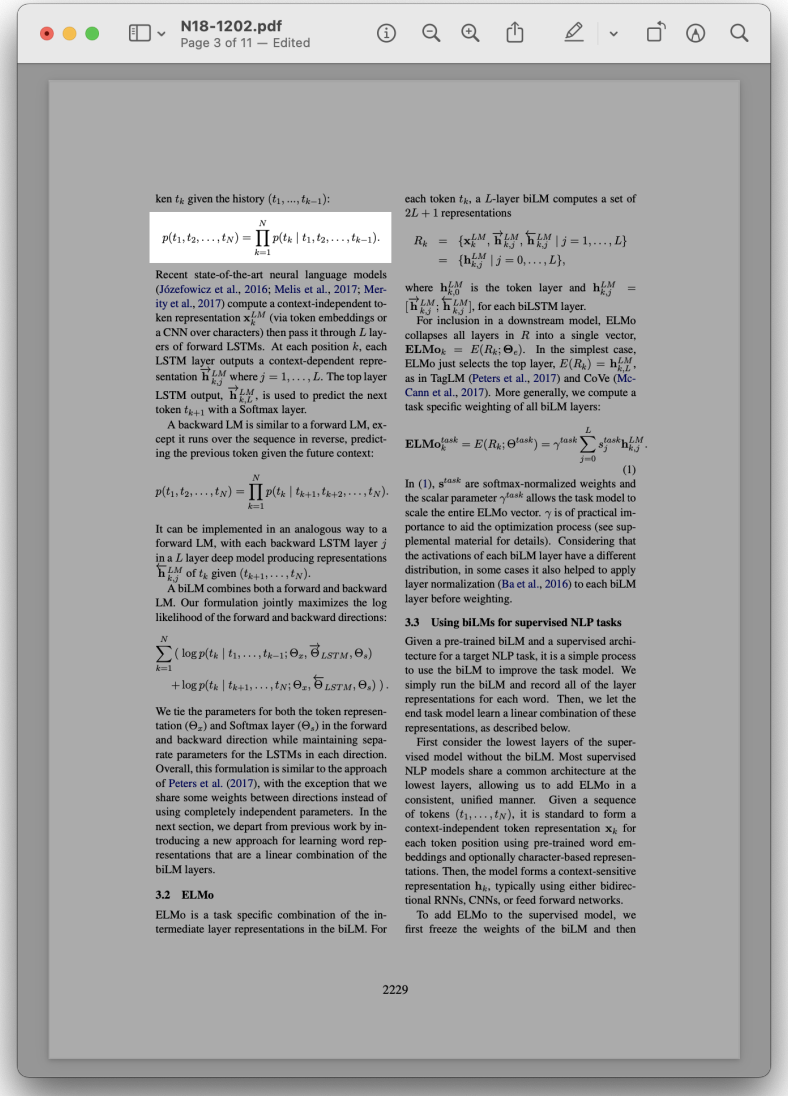
In (1), $\mathbf{s}^{task}$ are softmax-normalized weights and the scalar parameter γ^{task} allows the task model to scale the entire ELMo vector. γ is of practical importance to aid the optimization process (see supplemental material for details). Considering that the activations of each biLM layer 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



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, $\mathbf{ELMo}_k = E(R_k; \Theta_e)$. 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.

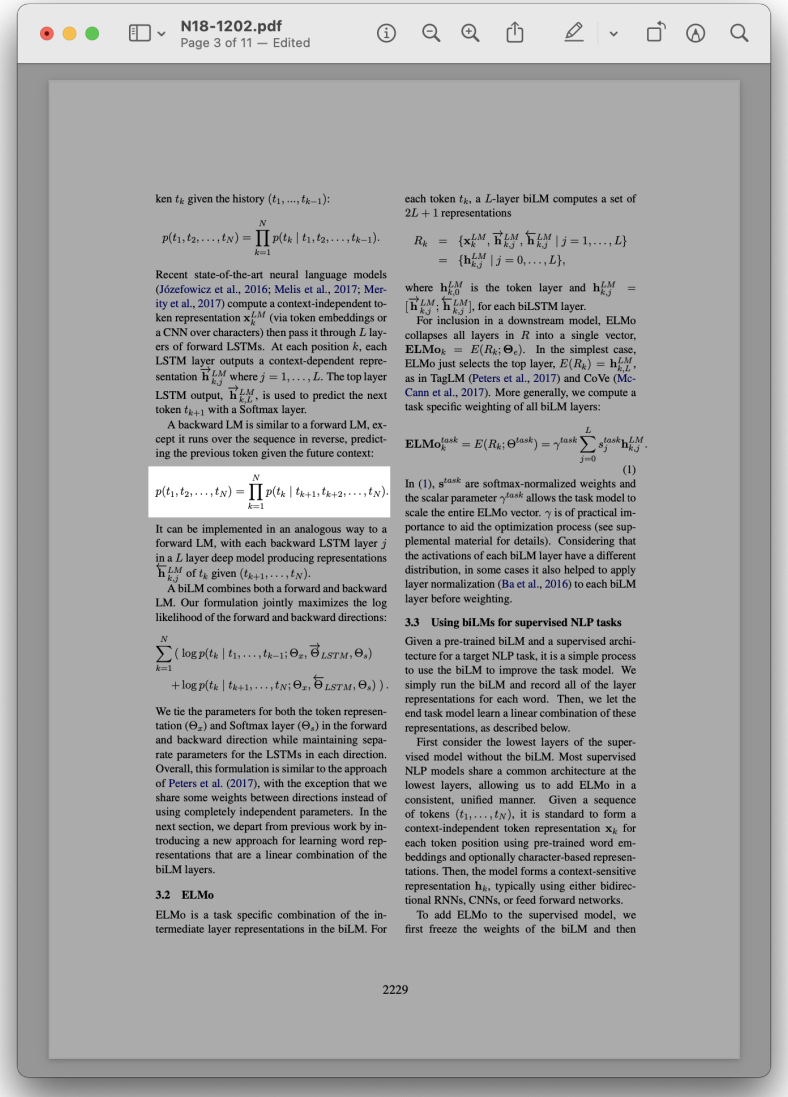
3.3 Using biLMs for supervised NLP tasks

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

Predict current word using previous words.



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, $\mathbf{ELMo}_k = E(R_k; \Theta_e)$. 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

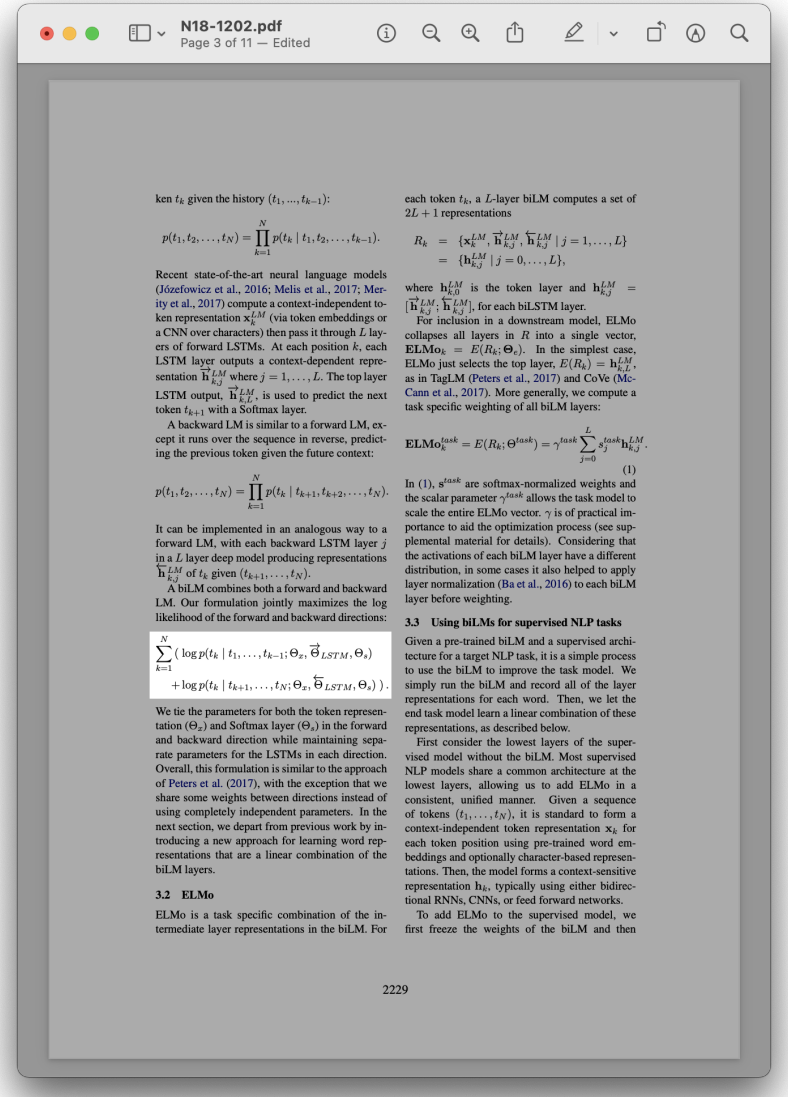
Given a pre-trained biLM and a supervised architecture for a target NLP task, it is a simple process to use the biLM to improve the task model. We simply run the biLM and record all of the layer representations for each word. Then, we let the end task model learn a linear combination of these representations,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

Predict current word using previous words.

Predict current word using following words.



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:

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

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, $\mathbf{ELMo}_k = E(R_k; \Theta_e)$. 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.

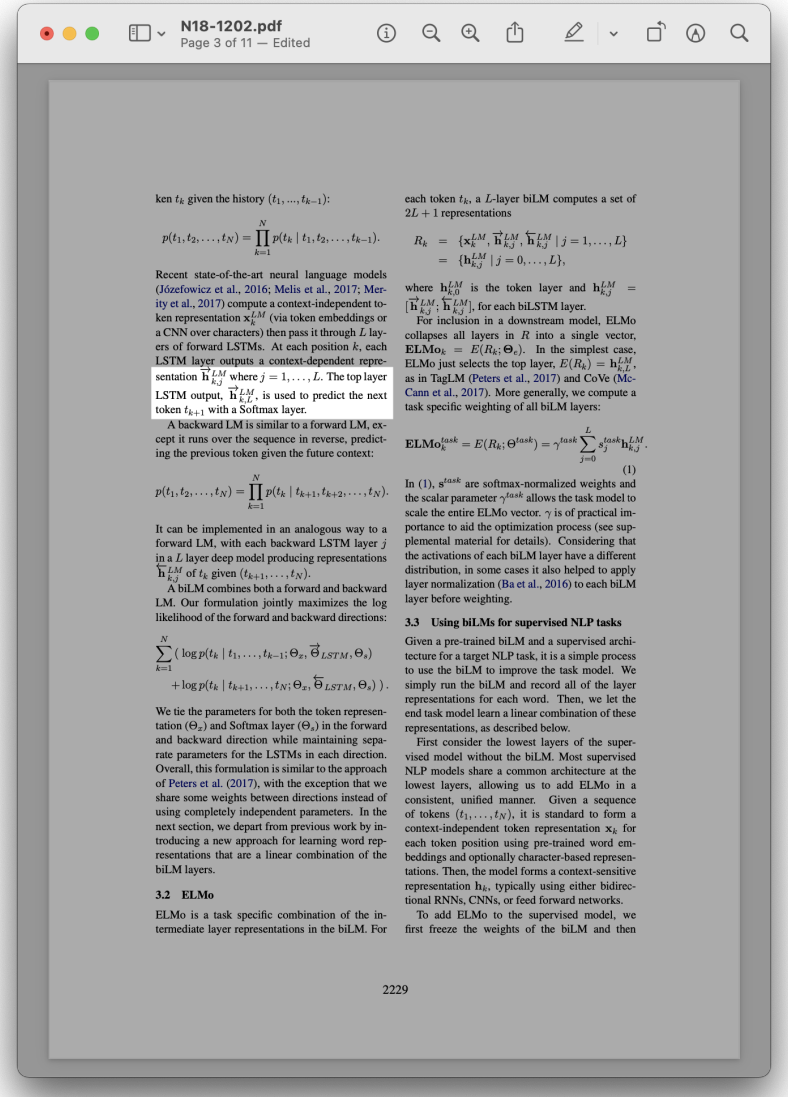
First consider the lowest layers of the supervised model without the biLM. Most supervised NLP models share a common architecture at the lowest layers, allowing us to add ELMo in a consistent, unified manner. Given a sequence of tokens $(t_1, \ldots, t_N)$, it is standard to form a context-independent 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

Predict current word using previous words.

Predict current word using following words.

Take a $\log$ to $\sum$, you know the game.



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, $\mathbf{ELMo}_k = E(R_k; \Theta_e)$. 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

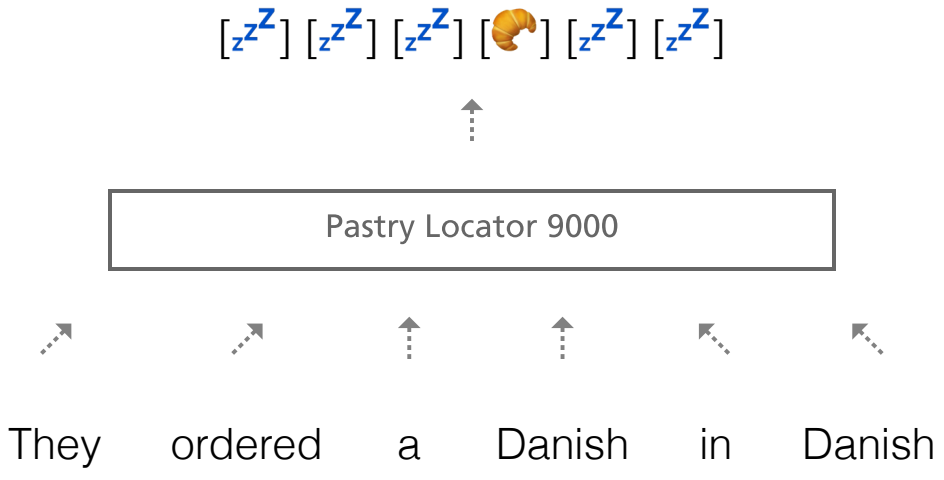
Predict current word using previous words.

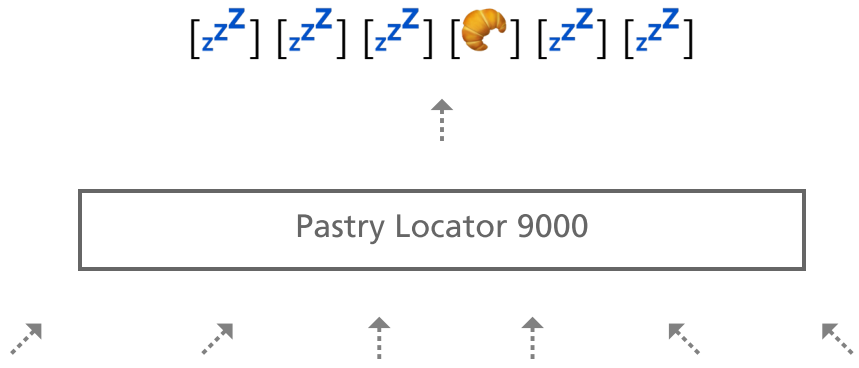
Predict current word using following words.

Take a $\log$ to $\sum$, you know the game.

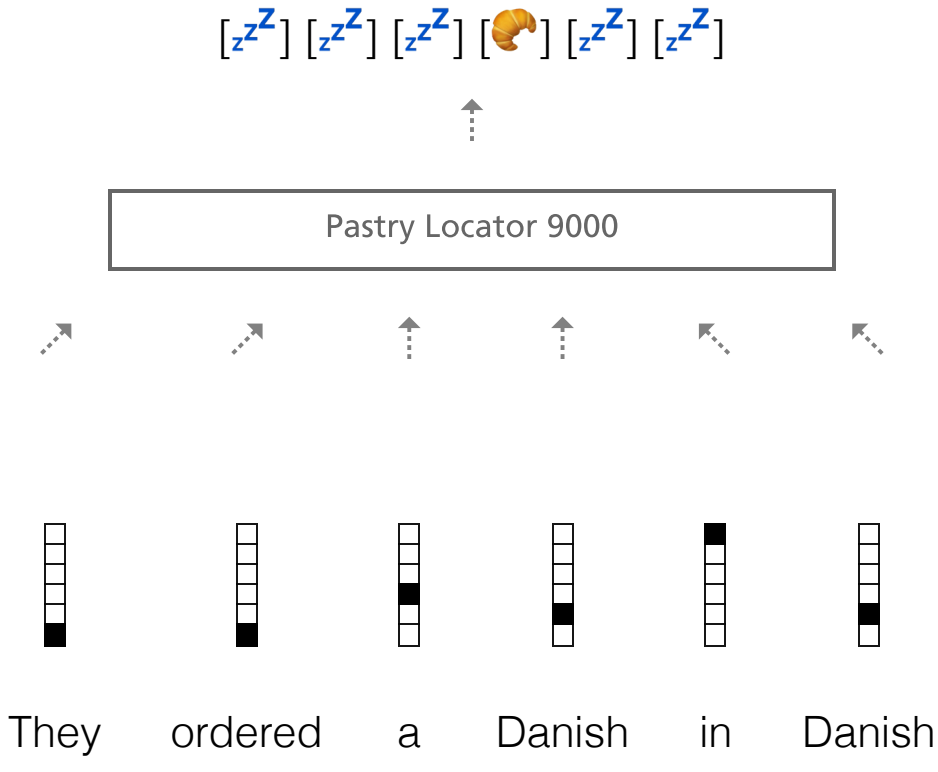
Turn logits into probabilities.

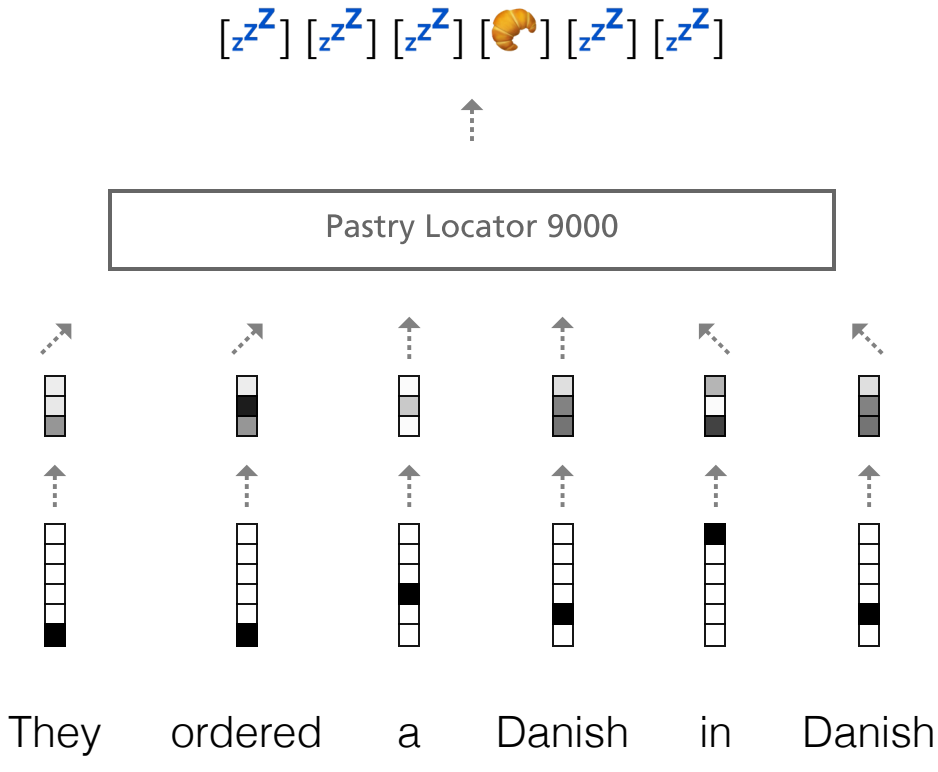
ELMo in Practice

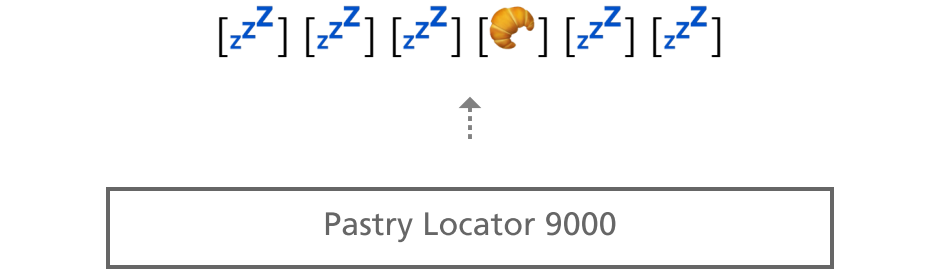

[💤] [💤] [💤] [🥐] [💤] [💤]
Pastry Locator 9000
They ordered a Danish in Danish



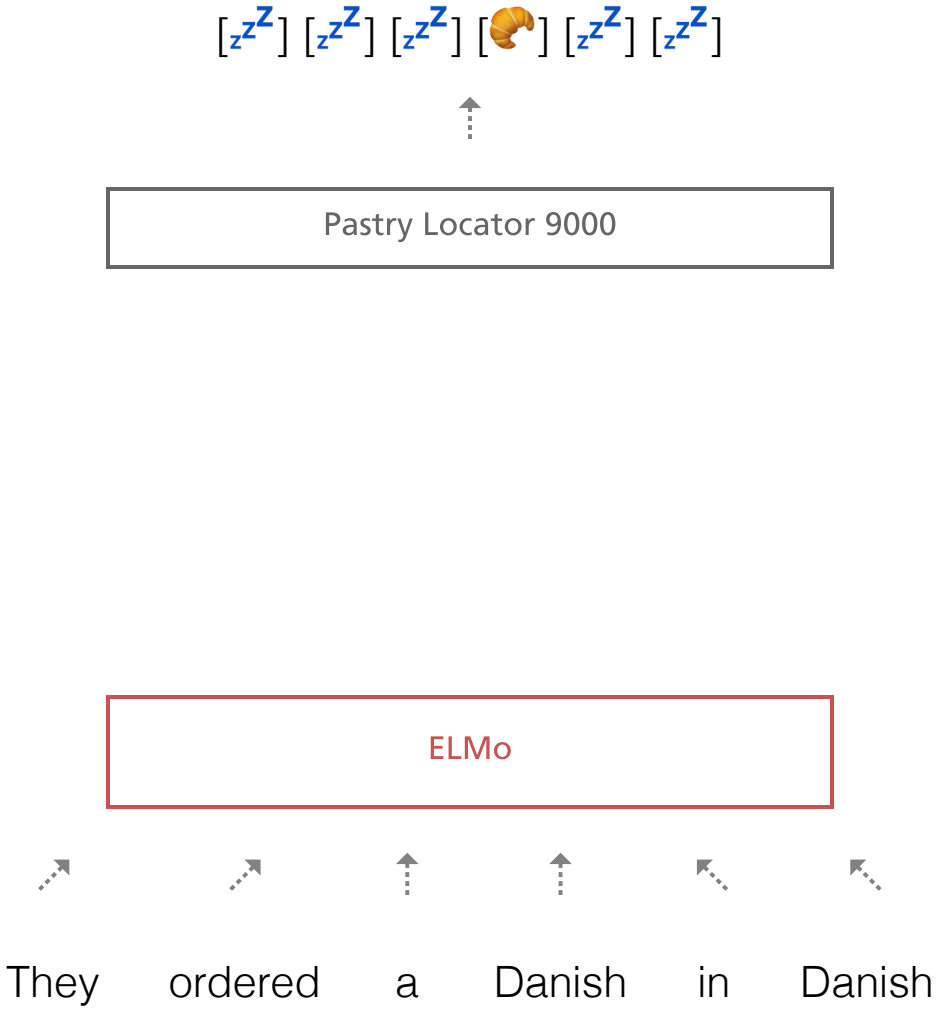
They ordered a Danish in Danish

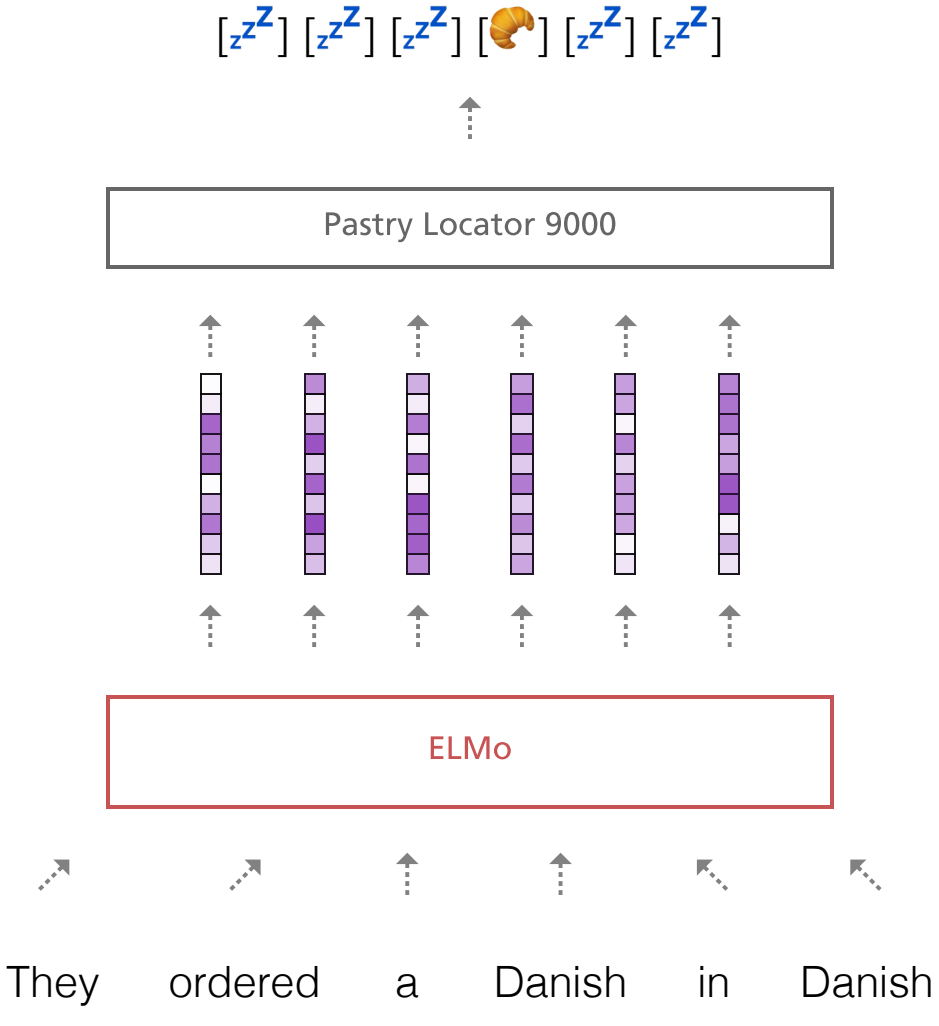

[zzz] [zzz] [zzz] [🥐] [zzz] [zzz]
Pastry Locator 9000
They
ordered
a
Danish
in
Danish

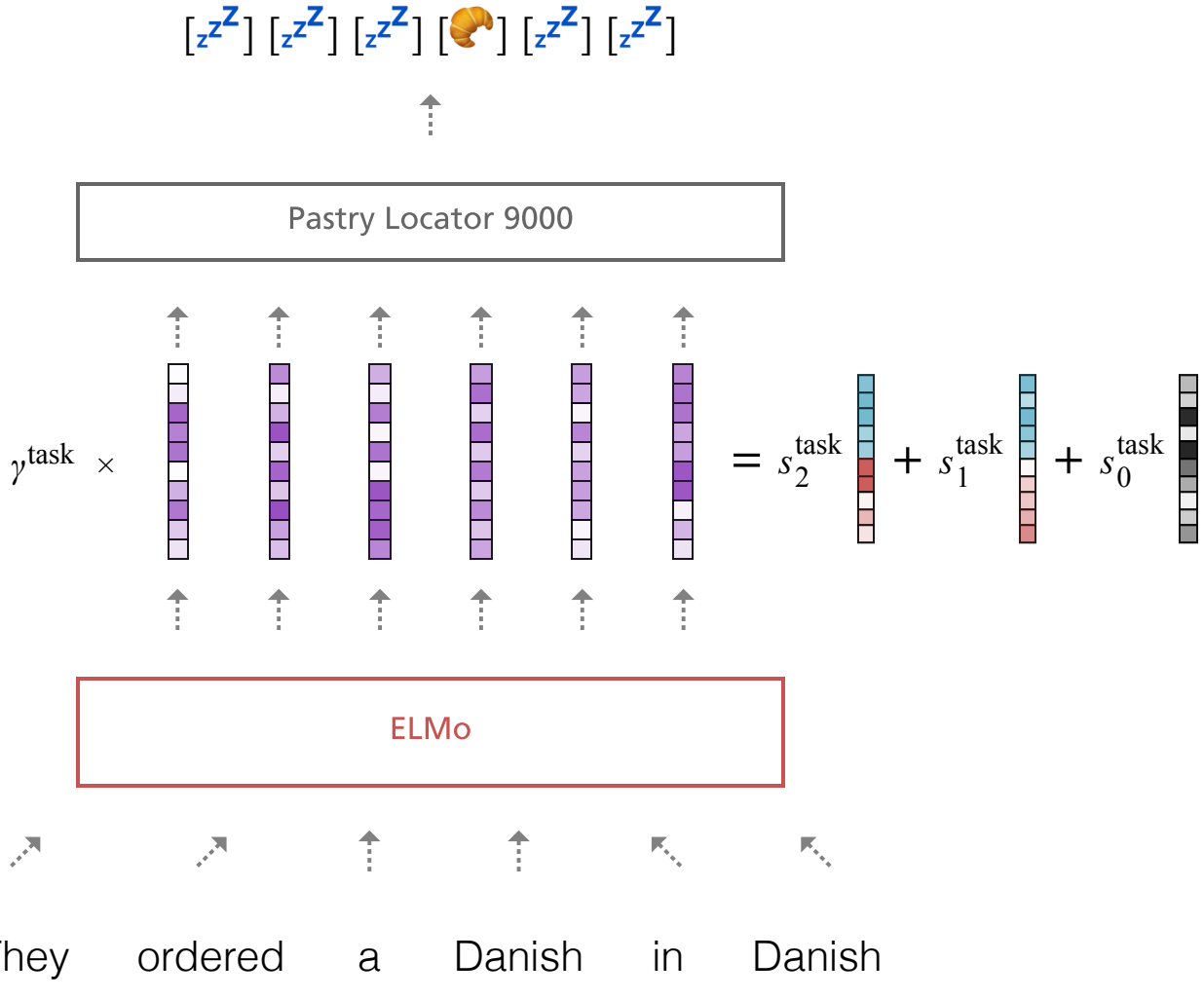

[zzz] [zzz] [zzz] [🥐] [zzz] [zzz]
Pastry Locator 9000
They
ordered
a
Danish
in
Danish

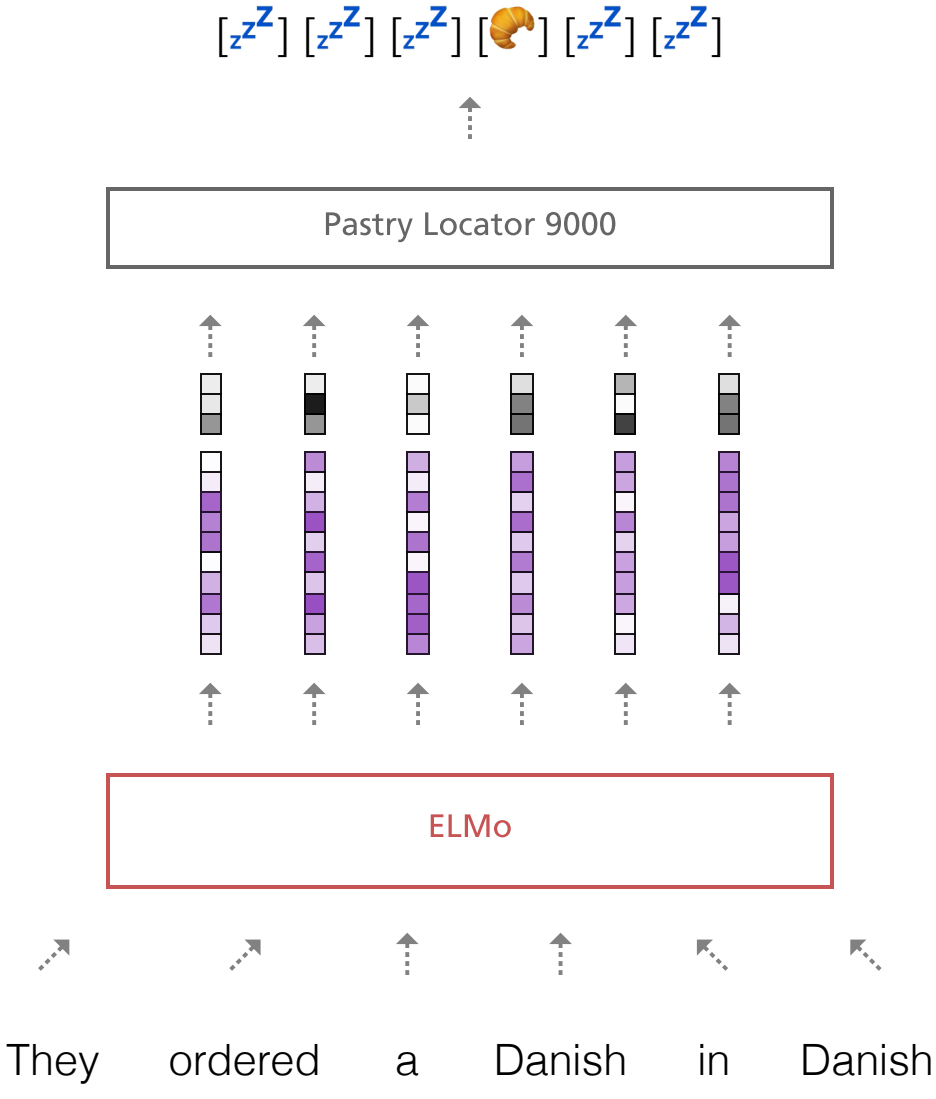


They ordered a Danish in Danish


[zzz] [zzz] [zzz] [🥐] [zzz] [zzz]
Pastry Locator 9000
ELMo
They ordered a Danish in Danish


[zZ] [zZ] [zZ] [🥐] [zZ] [zZ]
Pastry Locator 9000
ELMo
They ordered a Danish in Danish


[zzz] [zzz] [zzz] [🥐] [zzz] [zzz]
Pastry Locator 9000
$\gamma^{\text{task}} \times$
$= s_2^{\text{task}} + s_1^{\text{task}} + s_0^{\text{task}}$
ELMo
They ordered a Danish in Danish


[zzz] [zzz] [zzz] [🥐] [zzz] [zzz]
Pastry Locator 9000
ELMo
They ordered a Danish in Danish

ELMo in Practice

ELMo in Practice

1. Train ELMo on enourmous unannotated corpora
2. Plug frozen ELMo embeddings into any model that takes embeddings as input
3. Train the target model + ELMo's γ^{task} and s_j^{task}

$$[\mathbf{x}_k^{\text{old}}; \mathbf{ELMo}_k^{\text{task}}]$$

$$\mathbf{ELMo}_k^{\text{task}} = \gamma^{\text{task}} \sum_{j=0}^{L} s_j^{\text{task}} \mathbf{h}_{k,j}^{\text{LM}}$$

ELMo Unleashed

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.

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.

Explain architecture, hyperparameters, training setup.

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.

> Explain architecture, hyperparameters, training setup.

> Now entering experiment territory: Describe evaluation setup, present results. Don't judge (too much).

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.

Explain architecture, hyperparameters, training setup.

Now entering experiment territory: Describe evaluation setup, present results. Don't judge (too much).

What data are you using and how?

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.

Explain architecture, hyperparameters, training setup.

Now entering experiment territory: Describe evaluation setup, present results. Don't judge (too much).

What data are you using and how?
What baseline are you comparing against?

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.

Explain architecture, hyperparameters, training setup.

Now entering experiment territory: Describe evaluation setup, present results. Don't judge (too much).

What data are you using and how?
What baseline are you comparing against?
What do you measure? How do the methods compare?

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-

2231

Explain architecture, hyperparameters, training setup.

Now entering experiment territory: Describe evaluation setup, present results. Don't judge (too much).

What data are you using and how?
What baseline are you comparing against?
What do you measure? How do the methods compare?

Repeat.

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-

2231

Explain architecture, hyperparameters, training setup.

Now entering experiment territory: Describe evaluation setup, present results. Don't judge (too much).

What data are you using and how?
What baseline are you comparing against?
What do you measure? How do the methods compare?

Repeat.

This does not always fit. Adapt to your needs!

SQuAD

The Stanford Question Answering Dataset (Rajpukar et al., 2016)

What does the UN want to stabilize?

> The Intergovernmental Panel on Climate Change (IPCC) is a scientific intergovernmental body under the auspices of the United Nations, set up at the request of member governments. [...] The ultimate objective of the UNFCCC is to "stabilize greenhouse gas concentrations in the atmosphere at a level that would prevent dangerous anthropogenic [i.e., human-induced] interference with the climate system". [...]

SQuAD

The Stanford Question Answering Dataset (Rajpukar et al., 2016)

What does the UN want to stabilize?

> The Intergovernmental Panel on Climate Change (IPCC) is a scientific intergovernmental body under the auspices of the United Nations, set up at the request of member governments. [...] The ultimate objective of the UNFCCC is to "stabilize greenhouse gas concentrations in the atmosphere at a level that would prevent dangerous anthropogenic [i.e., human-induced] interference with the climate system". [...]

SQuAD

The Stanford Question Answering Dataset (Rajpukar et al., 2016)

What does the UN want to stabilize?

> The Intergovernmental Panel on Climate Change (IPCC) is a scientific intergovernmental body under the auspices of the United Nations, set up at the request of member governments. [...] The ultimate objective of the UNFCCC is to "stabilize greenhouse gas concentrations in the atmosphere at a level that would prevent dangerous anthropogenic [i.e., human-induced] interference with the climate system". [...]

84.4 → 85.8 (F1-Score)

SNLI

The Stanford Natural Language Inference Corpus (Bowman et al., 2015)

Do these sentences make sense within the same situation?

> A black race car starts up in front of a crowd of people.
>
> A man is driving down a lonely road.

SNLI

The Stanford Natural Language Inference Corpus (Bowman et al., 2015)

Do these sentences make sense within the same situation?

> A black race car starts up in front of a crowd of people.
> [contradicts]
> A man is driving down a lonely road.

SNLI

The Stanford Natural Language Inference Corpus (Bowman et al., 2015)

Do these sentences make sense within the same situation?

> A black race car starts up in front of a crowd of people.
> [contradicts]
> A man is driving down a lonely road.

88.6 → 88.7 (F1-Score)

SRL

Semantic Role Labeling — OntoNotes 5.0 (Pradhan et al., 2013)

Which semantic role does each token play?

> Microsoft Inc. acquired the startup last Thursday .

SRL

Semantic Role Labeling — OntoNotes 5.0 (Pradhan et al., 2013)

Which semantic role does each token play?

> Microsoft Inc. [PRED: acquired] the startup last Thursday .

SRL

Semantic Role Labeling — OntoNotes 5.0 (Pradhan et al., 2013)

Which semantic role does each token play?

> [A0: Microsoft Inc.] [PRED: acquired] the startup last Thursday .

SRL

Semantic Role Labeling — OntoNotes 5.0 (Pradhan et al., 2013)

Which semantic role does each token play?

> [A0: Microsoft Inc.] [PRED: acquired] [A1: the startup] last Thursday .

SRL

Semantic Role Labeling — OntoNotes 5.0 (Pradhan et al., 2013)

Which semantic role does each token play?

> [A0: Microsoft Inc.] [PRED: acquired] [A1: the startup] [AM-TMP: last Thursday] .

SRL

Semantic Role Labeling — OntoNotes 5.0 (Pradhan et al., 2013)

Which semantic role does each token play?

> [A0: Microsoft Inc.] [PRED: acquired] [A1: the startup] [AM-TMP: last Thursday] .

81.7 → 84.6 (F1-Score)

Coreference

CoNLL 2012 Shared Task (Pradhan et al., 2012)

Which spans refer to the same entity?

> The prime minister announced the new regulations this afternoon.
> She stated that this law was of the utmost importance.

Coreference

CoNLL 2012 Shared Task (Pradhan et al., 2012)

Which spans refer to the same entity?

> The prime minister announced the new regulations this afternoon.
> She stated that this law was of the utmost importance.

Coreference

CoNLL 2012 Shared Task (Pradhan et al., 2012)

Which spans refer to the same entity?

> The prime minister announced the new regulations this afternoon.
> She stated that this law was of the utmost importance.

Coreference

CoNLL 2012 Shared Task (Pradhan et al., 2012)

Which spans refer to the same entity?

> The prime minister announced the new regulations this afternoon.
> She stated that this law was of the utmost importance.

67.2 → 70.4 (F1-Score)

NER

CoNLL 2003 Named Entity Recognition (Sang and Meulder, 2003)

Which spans refer to a named entity?

> "Kanelsnurrer at Café Analog!", Max yelled, running past me.

NER

CoNLL 2003 Named Entity Recognition (Sang and Meulder, 2003)

Which spans refer to a named entity?

> "Kanelsnurrer at Café Analog!", Max yelled, running past me.
>
> B-MISC O B-ORG I-ORG B-PER O O O O

NER

CoNLL 2003 Named Entity Recognition (Sang and Meulder, 2003)

Which spans refer to a named entity?

> "Kanelsnurrer at Café Analog!", Max yelled, running past me.
>
> B-MISC O B-ORG I-ORG B-PER O O O O

91.93 → 92.22 (F1-Score)

SST-5

The Stanford Sentiment Treebank 5 (Socher et al., 2013)

What sentiment does the sentence convey?

> But believe it or not, it's one of the most beautiful, evocative works I've seen.

SST-5

The Stanford Sentiment Treebank 5 (Socher et al., 2013)

What sentiment does the sentence convey?

> But believe it or not, it's one of the most beautiful, evocative works I've seen. [very positive]

SST-5

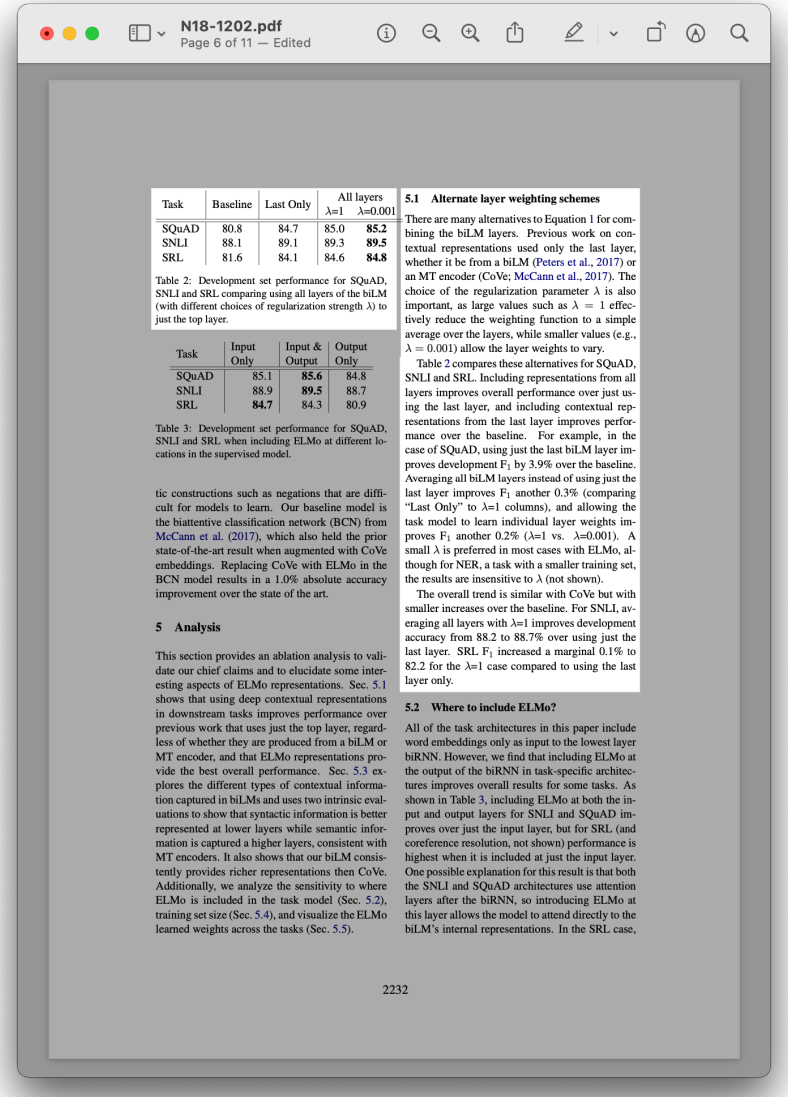
The Stanford Sentiment Treebank 5 (Socher et al., 2013)

What sentiment does the sentence convey?

> But believe it or not, it's one of the most beautiful, evocative works I've seen. [very positive]

53.7 → 54.7 (F1-Score)

Picking ELMo Apart



Task	Baseline	Last Only	All layers λ=1	All layers 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.

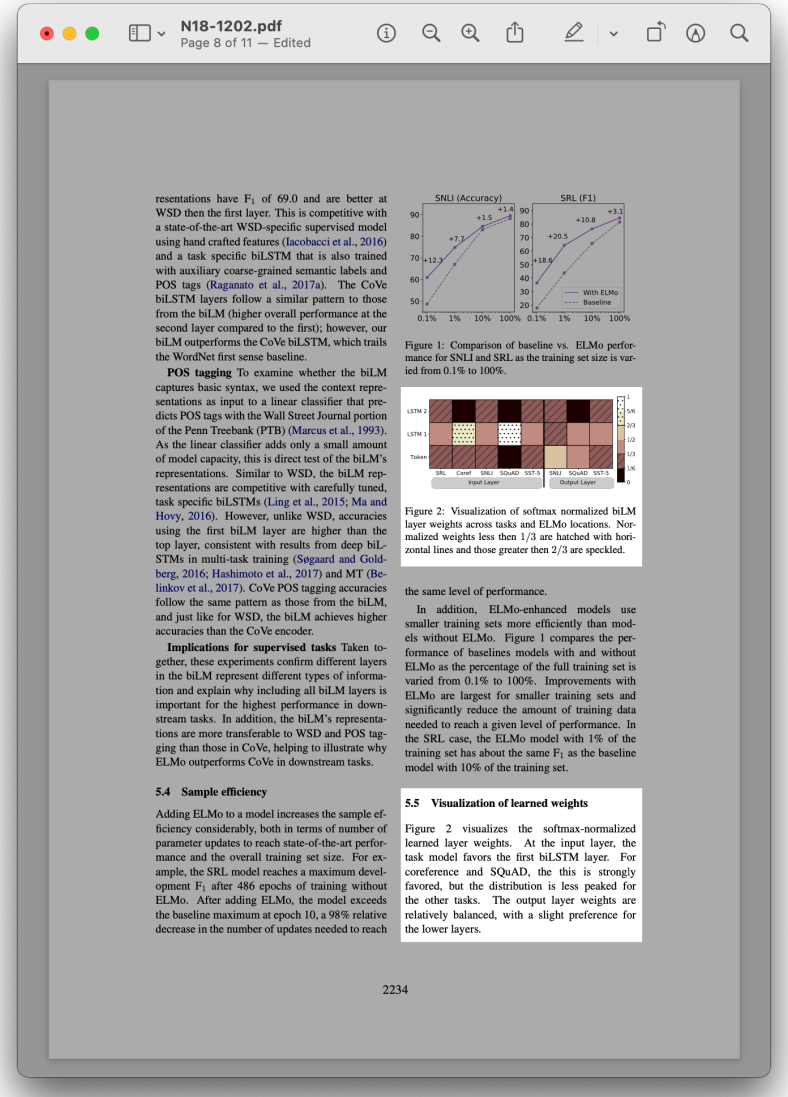
Table 2 compares these alternatives for SQuAD, SNLI and SRL. Including representations from all layers improves overall performance over just using the last layer, and including contextual representations from the last layer improves performance over the baseline. For example, in the case of SQuAD, using 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,

What's the best way to combine layers?



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.

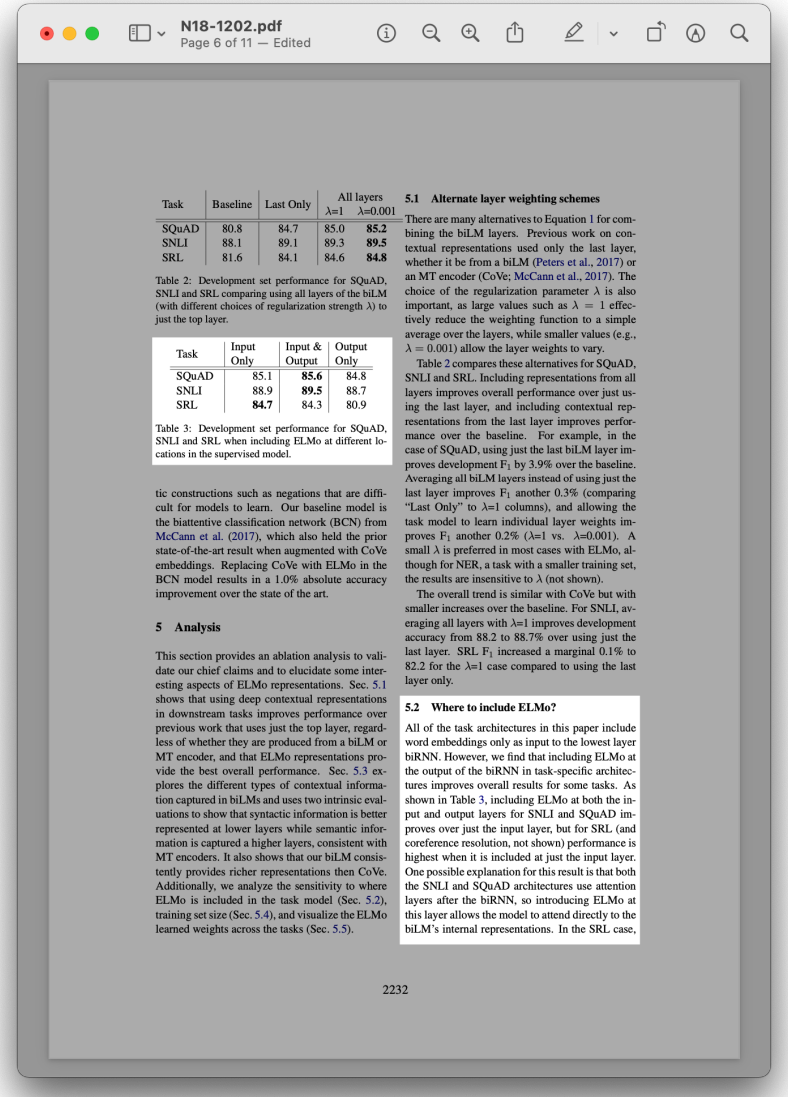
In addition, ELMo-enhanced models use smaller training sets more efficiently than models without ELMo. Figure 1 compares the performance of baselines models with and without ELMo as the percentage of the full training set is varied from 0.1% to 100%. Improvements with ELMo are largest for smaller training 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.

What's the best way to combine layers?

Which layers are useful for which tasks?



Task	Baseline	Last Only	All layers λ=1	All layers 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.

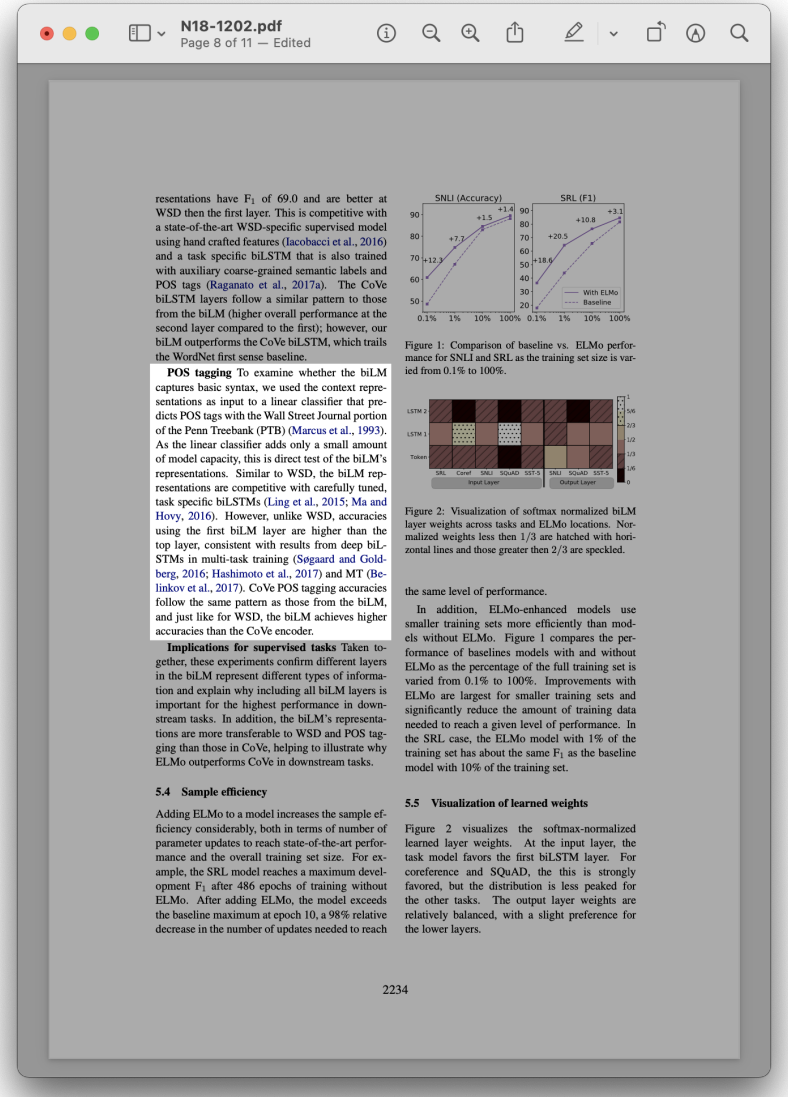
5.2 Where to include ELMo?

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,

What's the best way to combine layers?

Which layers are useful for which tasks?

Where should ELMo be added?



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.

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.

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.

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.

What's the best way to combine layers?

Which layers are useful for which tasks?

Where should ELMo be added?

Does ELMo capture syntax?

	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?

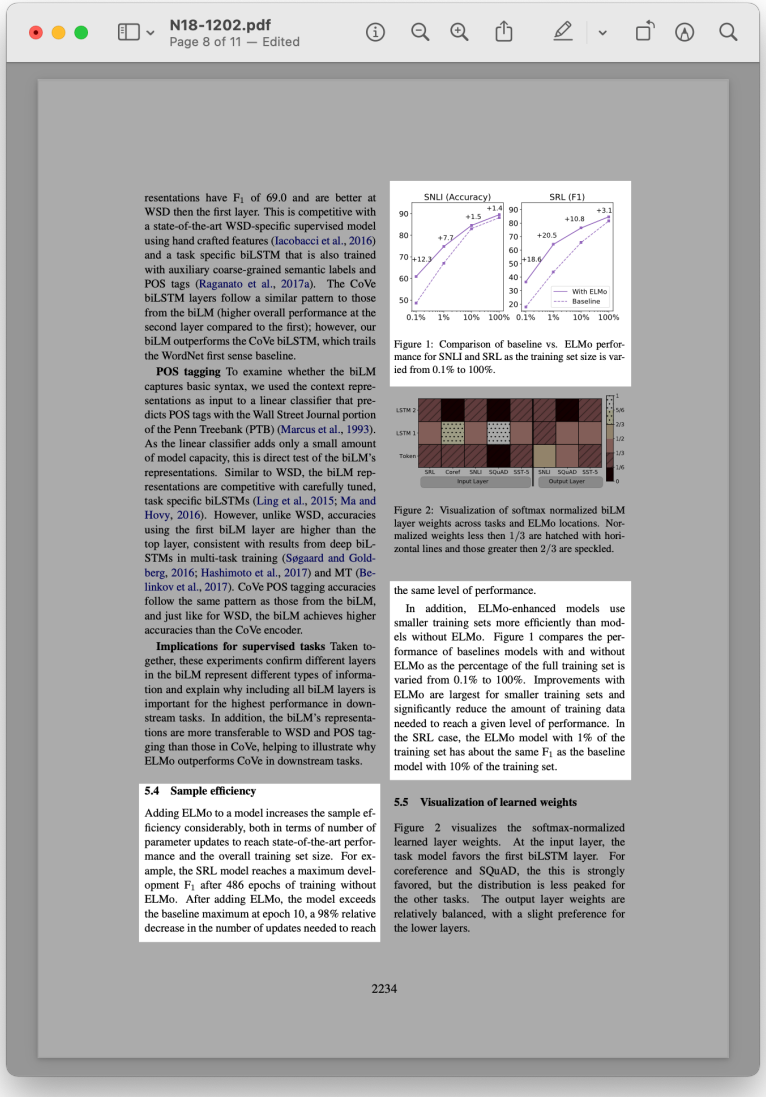
Since adding ELMo improves task performance over word vectors alone, the biLM's contextual representations must encode information generally useful for NLP tasks that is not captured in word vectors. Intuitively, the biLM must be disambiguating the meaning of words using their context. Consider "play", 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-

- What's the best way to combine layers?
- Which layers are useful for which tasks?
- Where should ELMo be added?
- Does ELMo capture syntax?
- Are the embeddings actually contextual?



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%.

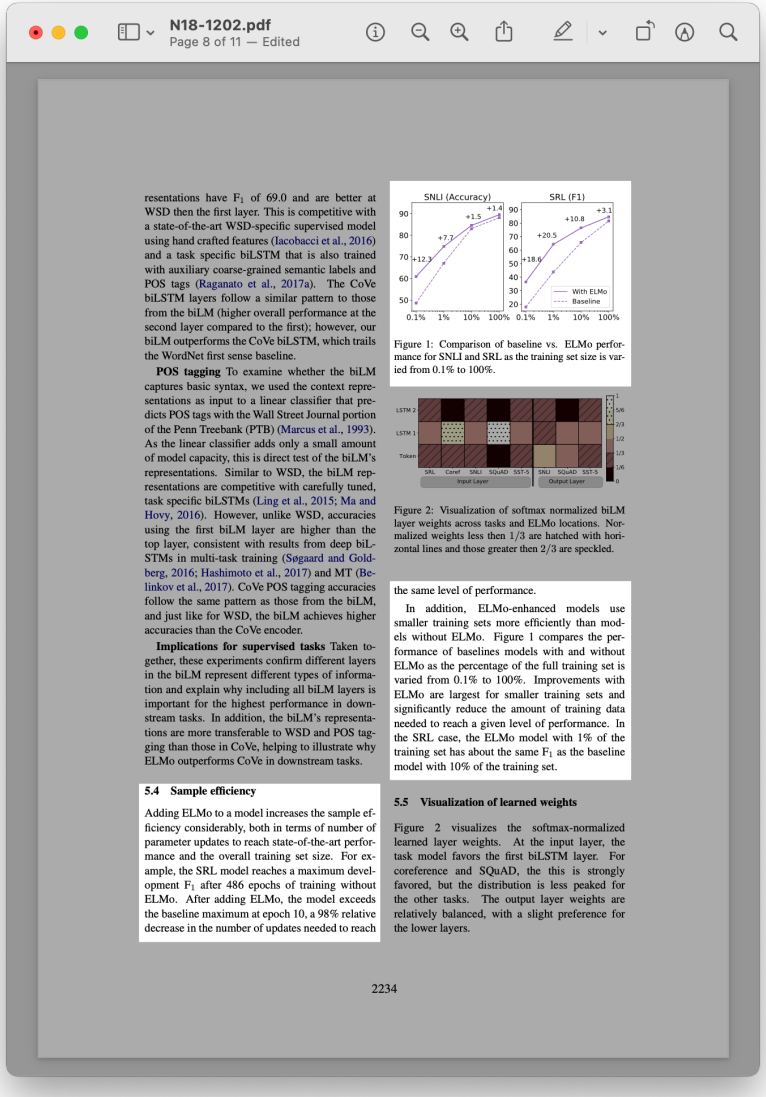
Figure 2: Visualization of softmax normalized biLM layer weights across tasks and ELMo locations. Normalized weights less then 1/3 are hatched with horizontal lines and those greater then 2/3 are speckled.

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.

With this demonstrated, it's time to flex.



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

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.

the same level of performance.

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.

With this demonstrated, it's time to flex.

> Use ELMo for faster convergence!

Task	GloVe	ELMo type	ELMo	ELMo + GloVe
SQuAD	80.8	81.4	85.3	85.6
SNLI	88.1	88.5	89.1	89.5
SRL	81.6	81.7	84.5	84.7

Table 7: Development set ablation analysis for SQuAD, SNLI and SRL comparing different choices for the context-independent type representation and contextual representation. From left to right, the table compares systems with only GloVe vectors; only the ELMo context-independent type representation without the ELMo biLSTM layers; full ELMo representations without GloVe; both GloVe and ELMo.

5.6 Contextual vs. sub-word information

In addition to the contextual information captured in the biLM's biLSTM layers, ELMo representations also contain sub-word information in the fully character based context insensitive type layer, x_k^{LM}. To analyze the relative contribution of the contextual information compared to the sub-word information, we ran an additional ablation that replaced the GloVe vectors with just the biLM character based x_k^{LM} layer without the biLM biLSTM layers. Table 7 summarizes the results for SQuAD, SNLI and SNLI. Replacing the GloVe vectors with the biLM character layer gives a slight improvement for all tasks (e.g. from 80.8 to 81.4 F_1 for SQuAD), but overall the improvements are small compared to the full ELMo model. From this, we conclude that most of the gains in the downstream tasks are due to the contextual information and not the sub-word information.

5.7 Are pre-trained vectors necessary with ELMo?

All of the results presented in Sec.4 include pre-trained word vectors in addition to ELMo representations. However, it is natural to ask whether pre-trained vectors are still necessary with high quality contextualized representations. As shown in the two right hand columns of Table 7, adding GloVe to models with ELMo generally provides a marginal improvement over ELMo only models (e.g. 0.2% F_1 improvement for SRL from 84.5 to 84.7).

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*.

With this demonstrated, it's time to flex.

> Use ELMo for faster convergence!

> Do we even need non-contextual embeddings?

Task	GloVe	ELMo type	ELMo	ELMo + GloVe
SQuAD	80.8	81.4	85.3	85.6
SNLI	88.1	88.5	89.1	89.5
SRL	81.6	81.7	84.5	84.7

Table 7: Development set ablation analysis for SQuAD, SNLI and SRL comparing different choices for the context-independent type representation and contextual representation. From left to right, the table compares systems with only GloVe vectors; only the ELMo context-independent type representation without the ELMo biLSTM layers; full ELMo representations without GloVe; both GloVe and ELMo.

5.6 Contextual vs. sub-word information

In addition to the contextual information captured in the biLM's biLSTM layers, ELMo representations also contain sub-word information in the fully character based context insensitive type layer, x_k^{LM}. To analyze the relative contribution of the contextual information compared to the sub-word information, we ran an additional ablation that replaced the GloVe vectors with just the biLM character based x_k^{LM} layer without the biLM biLSTM layers. Table 7 summarizes the results for SQuAD, SNLI and SNLI. Replacing the GloVe vectors with the biLM character layer gives a slight improvement for all tasks (e.g. from 80.8 to 81.4 F_1 for SQuAD), but overall the improvements are small compared to the full ELMo model. From this, we conclude that most of the gains in the downstream tasks are due to the contextual information and not the sub-word information.

5.7 Are pre-trained vectors necessary with ELMo?

All of the results presented in Sec.4 include pre-trained word vectors in addition to ELMo representations. However, it is natural to ask whether pre-trained vectors are still necessary with high quality contextualized representations. As shown in the two right hand columns of Table 7, adding GloVe to models with ELMo generally provides a marginal improvement over ELMo only models (e.g. 0.2% F_1 improvement for SRL from 84.5 to 84.7).

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*.

With this demonstrated, it's time to flex.

> Use ELMo for faster convergence!

> Do we even need non-contextual embeddings?

Task	GloVe	ELMo type	ELMo	ELMo + GloVe
SQuAD	80.8	81.4	85.3	85.6
SNLI	88.1	88.5	89.1	89.5
SRL	81.6	81.7	84.5	84.7

Table 7: Development set ablation analysis for SQuAD, SNLI and SRL comparing different choices for the context-independent type representation and contextual representation. From left to right, the table compares systems with only GloVe vectors; only the ELMo context-independent type representation without the ELMo biLSTM layers; full ELMo representations without GloVe; both GloVe and ELMo.

5.6 Contextual vs. sub-word information

In addition to the contextual information captured in the biLM's biLSTM layers, ELMo representations also contain sub-word information in the fully character based context insensitive type layer, x_k^{LM}. To analyze the relative contribution of the contextual information compared to the sub-word information, we ran an additional ablation that replaced the GloVe vectors with just the biLM character based x_k^{LM} layer without the biLM biLSTM layers. Table 7 summarizes the results for SQuAD, SNLI and SNLI. Replacing the GloVe vectors with the biLM character layer gives a slight improvement for all tasks (e.g. from 80.8 to 81.4 F_1 for SQuAD), but overall the improvements are small compared to the full ELMo model. From this, we conclude that most of the gains in the downstream tasks are due to the contextual information and not the sub-word information.

5.7 Are pre-trained vectors necessary with ELMo?

All of the results presented in Sec.4 include pre-trained word vectors in addition to ELMo representations. However, it is natural to ask whether pre-trained vectors are still necessary with high quality contextualized representations. As shown in the two right hand columns of Table 7, adding GloVe to models with ELMo generally provides a marginal improvement over ELMo only models (e.g. 0.2% F_1 improvement for SRL from 84.5 to 84.7).

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*.

With this demonstrated, it's time to flex.

> Use ELMo for faster convergence!

> Do we even need non-contextual embeddings?

> In conclusion, we did all the things.

Takeaways

Embeddings from Language Models

- ELMo uses context during training and inference
  - Characters are encoded into non-contextual word embeddings
  - Contextualized token embeddings from hidden states of deep BiLSTM
  - Language Modeling Objective: Predict a token based on context
- For each task, ELMo's weights remain fixed, only scaling and layer weighting are updated

Writing Academic Papers

- Motivate every decision, including why there is a question to answer in the first place
- Build on, compare to and correctly reference related work
- Document your methodology and experiments such that anyone can reproduce them
- Report results neutrally and only draw conclusions that your evidence supports
- Summarize what was done in the end (and the beginning)
- Adapt to your target audience

References

References I

1. Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (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)* (pp. 2227–2237).
2. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.
3. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.
4. Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.
5. Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015). A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

References II

6. Pradhan, S., Moschitti, A., Xue, N., Ng, H. T., Björkelund, A., Uryupina, O., ... & Zhong, Z. (2013, August). Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning* (pp. 143-152).
7. Pradhan, S., Moschitti, A., Xue, N., Uryupina, O., & Zhang, Y. (2012, July). CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL-Shared Task* (pp. 1-40).
3. Sang, E. F., & De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.
9. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013, October). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1631-1642).

Thanks

Good luck with your projects!