Dissecting Contextual Word Embeddings:
Architecture and Representation

Matthew E. Peters[* 1], **Mark Neumann**[* 1], **Luke Zettlemoyer**[2], **Wen-tau Yih**[1]

[1]Allen Institute for Artificial Intelligence, Seattle, WA, USA

[2]Paul G. Allen Computer Science & Engineering, University of Washington

{matthewp,markn,scottyih}@allenai.org lsz@cs.washington.edu

Abstract

Contextual word representations derived from pre-trained bidirectional language models (biLMs) have recently been shown to provide significant improvements to the state of the art for a wide range of NLP tasks. However, many questions remain as to how and why these models are so effective. In this paper, we present a detailed empirical study of how the choice of neural architecture (e.g. LSTM, CNN, or self attention) influences both end task accuracy and qualitative properties of the representations that are learned. We show there is a tradeoff between speed and accuracy, but all architectures learn high quality contextual representations that outperform word embeddings for four challenging NLP tasks. Additionally, all architectures learn representations that vary with network depth, from exclusively morphological based at the word embedding layer through local syntax based in the lower contextual layers to longer range semantics such coreference at the upper layers. Together, these results suggest that unsupervised biLMs, independent of architecture, are learning much more about the structure of language than previously appreciated.

1   Introduction

Contextualized word embeddings (Peters et al., 2018) derived from pre-trained bidirectional language models (biLMs) have been shown to substantially improve performance for many NLP tasks including question answering, entailment and sentiment classification (Peters et al., 2018), constituency parsing (Kitaev and Klein, 2018; Joshi et al., 2018), named entity recognition (Peters et al., 2017), and text classification (Howard and Ruder, 2018). Despite large gains (typical relative error reductions range from 10–25%), we do not yet fully understand why or how these models

work in practice. In this paper, we take a step towards such understanding by empirically studying how the choice of neural architecture (e.g. LSTM, CNN, or self attention) influences both direct end-task accuracies and the types of neural representations that are induced (e.g. how do they encode notions of syntax and semantics).

Previous work on learning contextual representations has used LSTM-based biLMs, but there is no prior reason to believe this is the best possible architecture. More computationally efficient networks have been introduced for sequence modeling including including gated CNNs for language modeling (Dauphin et al., 2017) and feed forward self-attention based approaches for machine translation (Transformer; Vaswani et al., 2017). As RNNs are forced to compress the entire history into a hidden state vector before making predictions while CNNs with a large receptive field and the Transformer may directly reference previous tokens, each architecture will represent information in a different manner.

Given such differences, we study whether more efficient architectures can also be used to learn high quality contextual vectors. We show empirically that all three approaches provide large improvements over traditional word vectors when used in state-of-the-art models across four benchmark NLP tasks. We do see the expected tradeoff between speed and accuracy between LSTMs and the other alternatives, but the effect is relatively modest and all three networks work well in practice.

Given this result, it is important to better understand what the different networks learn. In a detailed quantitative evaluation, we probe the learned representations and show that, in every case, they represent a rich hierarchy of contextual information throughout the layers of the network in an analogous manner to how deep CNNs

---

*These authors contributed equally to this work.

trained for image classification learn a hierarchy of image features (Zeiler and Fergus, 2014). For example, we show that in contrast to traditional word vectors which encode some semantic information, the word embedding layer of deep biLMs focuses exclusively on word morphology. Moving upward in the network, the lowest contextual layers of biLMs focus on local syntax, while the upper layers can be used to induce more semantic content such as within-sentence pronominal coreferent clusters. We also show that the biLM activations can be used to form phrase representations useful for syntactic tasks. Together, these results suggest that large scale biLMs, independent of architecture, are learning much more about the structure of language than previous appreciated.

2 Contextual word representations from biLMs

To learn contextual word representations, we follow previous work by first training a biLM on a large text corpus (Sec. 2.1). Then, the internal layer activations from the biLM are transferred to downstream tasks (Sec. 2.3).

2.1 Bidirectional language models

Given a sequence of N tokens, $(t_1, t_2, ..., t_N)$, a biLM combines a forward and backward language model to jointly maximize the log likelihood of both directions:

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

where $\overrightarrow{\Theta}$ and $\overleftarrow{\Theta}$ are the parameters of the forward and backward LMs respectively.

To compute the probability of the next token, state-of-the-art neural LMs first produce a context-insensitive token representation or word embedding, $\mathbf{x}_k$, (with either an embedding lookup or in our case a character aware encoder, see below). Then, they compute L layers of context-dependent representations $\overrightarrow{\mathbf{h}}_{k,i}$ where $i \in [1, L]$ using a RNN, CNN or feed forward network (see Sec. 3). The top layer output $\overrightarrow{\mathbf{h}}_{k,L}$ is used to predict the next token using a Softmax layer. The backward LM operates in an analogous manner to the forward LM. Finally, we can concatenate the forward and backward states to form L layers of contextual representations, or context vectors, at each to-

ken position: $\mathbf{h}_{k,i} = [\overrightarrow{\mathbf{h}}_{k,i}; \overleftarrow{\mathbf{h}}_{k,i}]$. When training, we tie the weights of the word embedding layers and Softmax in each direction but maintain separate weights for the contextual layers.

2.2 Character based language models

Fully character aware models (Kim et al., 2015) are considerably more parameter efficient then word based models but more computationally expensive then word embedding based methods when training. During inference, these differences can be largely eliminated by pre-computing embeddings for a large vocabulary and only falling back to the full character based method for rare words. Overall, for a large English language news benchmark, character aware models have slightly better perplexities then word based ones, although the differences tend to be small (Józefowicz et al., 2016).

Similar to Kim et al. (2015), our character-to-word encoder is a five-layer sub-module that first embeds single characters with an embedding layer then passes them through 2048 character n-gram CNN filters with max pooling, two highway layers (Srivastava et al., 2015), and a linear projection down to the model dimension.

2.3 Deep contextual word representations

After pre-training on a large data set, the internal representations from the biLM can be transferred to a downstream model of interest as contextual word representations. To effectively use all of the biLM layers, Peters et al. (2018) introduced ELMo word representations, whereby all of the layers are combined with a weighted average pooling operation, $\mathbf{ELMo}_k = \gamma \sum_{j=0}^{L} s_j \mathbf{h}_{k,j}$. The parameters $\mathbf{s}$ are optimized as part of the task model so that it may preferentially mix different types of contextual information represented in different layers of the biLM. In Sec. 4 we evaluate the relative effectiveness of ELMo representations from three different biLM architectures vs. pre-trained word vectors in four different state-of-the-art models.

3 Architectures for deep biLMs

The primary design choice when training deep biLMs for learning context vectors is the choice of the architecture for the contextual layers. However, it is unknown if the architecture choice is important for the quality of learned representations. To study this question, we consider two alterna-

Architecture	# layers	Perplexity	# params. (M)	Inference (ms) 1 sentence	Inference (ms) 64 sentences
LSTM	2	39.7	76 / 94	44 / 46	66 / 85
LSTM	4	37.5	151 / 153	85 / 86	102 / 118
Transformer	6	40.7	38 / 56	12 / 13	22 / 44
Gated CNN	16	44.5	67 / 85	9 / 11	29 / 55

Table 1: Characteristics of the different biLMs in this study. For each model, the table shows the number of layers used for the contextual representations, the averaged forward and backward perplexities on the 1 Billion Word Benchmark, the number of parameters (in millions, excluding softmax) and the inference speed (in milliseconds with a Titan X GPU, for sentences with 20 tokens, excluding softmax). For the number of parameters and inference speeds we list both the values for just the contextual layers and all layers needed to compute context vectors.

tives to LSTMs as described below. See the appendix for the hyperparameter details.

3.1 LSTM

Among the RNN variants, LSTMs have been shown to provide state-of-the-art performance for several benchmark language modeling tasks (Józefowicz et al., 2016; Merity et al., 2018; Melis et al., 2018). In particular, the LSTM with projection introduced by Sak et al. (2014) allows the model to use a large hidden state while reducing the total number of parameters.This is the architecture adopted by Peters et al. (2018) for computing ELMo representations. In addition to the pre-trained 2-layer biLM from that work,[1] we also trained a deeper 4-layer model to examine the impact of depth using the publicly available training code.[2] To reduce the training time for this large 4-layer model, we reduced the number of parameters in the character encoder by first projecting the character CNN filters down to the model dimension before the two highway layers.

3.2 Transformer

The Transformer, introduced by Vaswani et al. (2017), is a feed forward self-attention based architecture. In addition to machine translation, it has also provided strong results for Penn Treebank constituency parsing (Kitaev and Klein, 2018) and semantic role labeling (Tan et al., 2018). Each identical layer in the encoder first computes a multi-headed attention between a given token and all other tokens in the history, then runs a position wise feed forward network.

To adapt the Transformer for bidirectional language modeling, we modified a PyTorch based re-implementation (Klein et al., 2017)[3] to mask out future tokens for the forward language model and previous tokens for the backward language model, in a similar manner to the decoder masking in the original implementation. We adopted hyper-parameters from the "base" configuration in Vaswani et al. (2017), providing six layers of 512 dimensional representations for each direction.

Concurrent with our work, Radford et al. (2018) trained a large forward Transformer LM and fine tuned it for a variety of NLP tasks.

3.3 Gated CNN

Convolutional architectures have also been shown to provide competitive results for sequence modeling including sequence-to-sequence machine translation (Gehring et al., 2017). Dauphin et al. (2017) showed that architectures using Gated Linear Units (GLU) that compute hidden representations as the element wise product of a convolution and sigmoid gate provide perplexities comparable to large LSTMs on large scale language modeling tasks.

To adapt the Gated CNN for bidirectional language modeling, we closely followed the publicly available ConvSeq2Seq implementation,[4] modified to support causal convolutions (van den Oord et al., 2016) for both the forward and backward directions. In order to model a wide receptive field at the top layer, we used a 16-layer deep model, where each layer is a $[4, 512]$ residual block.

3.4 Pre-trained biLMs

Table 1 compares the biLMs used in the remainder of this study. All models were trained on the 1

---

[1] http://allennlp.org/elmo
[2] https://github.com/allenai/bilm-tf

[3] http://nlp.seas.harvard.edu/2018/04/03/attention.html
[4] https://github.com/pytorch/fairseq

Architecture	MultiNLI	SRL	Constituency Parsing	NER
GloVe	77.0 / 76.0	81.4	91.8	89.9 ± 0.35
LSTM 2-layer	79.6 / 79.3	84.6	**93.9**	**91.7 ± 0.26**
LSTM 4-layer	**80.1 / 79.7**	**84.7**	**93.9**	91.5 ± 0.12
Transformer	79.4 / 78.7	84.1	93.7	91.1 ± 0.26
Gated CNN	78.3 / 77.9	84.1	92.9	91.2 ± 0.14

Table 2: Test set performance comparison using different pre-trained biLM architectures. The performance metric is accuracy for MultiNLI and F_1 score for the other tasks. For MultiNLI, the table shows accuracy on both the matched and mismatched portions of the test set.

Billion Word Benchmark (Chelba et al., 2014) using a sampled softmax with 8192 negative samples per batch. Overall, the averaged forward and backward perplexities are comparable across the models with values ranging from 37.5 for the 4-layer LSTM to 44.5 for the Gated CNN. To our knowledge, this is the first time that the Transformer has been shown to provide competitive results for language modeling. While it is possible to reduce perplexities for all models by scaling up, our goal is to compare representations across architectures for biLMs of approximately equal skill, as measured by perplexity.

The Transformer and CNN based models are faster than the LSTM based ones for our hyperparameter choices, with speed ups of 3-5X for the contextual layers over the 2-layer LSTM model.[5] Speed ups are relatively faster in the single element batch scenario where the sequential LSTM is most disadvantaged, but are still 2.3-3X for a 64 sentence batch. As the inference speed for the character based word embeddings could be mostly eliminated in a production setting, the table lists timings for both the contextual layers and all layers of the biLM necessary to compute context vectors. We also note that the faster architectures will allow training to scale to large unlabeled corpora, which has been shown to improve the quality of biLM representations for syntactic tasks (Zhang and Bowman, 2018).

4 Evaluation as word representations

In this section, we evaluate the quality of the pre-trained biLM representations as ELMo-like contextual word vectors in state-of-the-art models across a suite of four benchmark NLP tasks. To do so, we ran a series of controlled trials by swapping out pre-trained GloVe vectors (Pennington et al., 2014) for contextualized word vectors from each biLM computed by applying the learned weighted average ELMo pooling from Peters et al. (2018).[6] Each task model only includes one type of pre-trained word representation, either GloVe or ELMo-like, this is a direct test of the transferability of the word representations. In addition, to isolate the general purpose LM representations from any task specific supervision, we did not fine tune the LM weights.

Table 2 shows the results. Across all tasks, the LSTM architectures perform the best. All architectures improve significantly over the GloVe only baseline, with relative improvements of 13% − 25% for most tasks and architectures. The gains for MultiNLI are more modest, with relative improvements over GloVe ranging from 6% for the Gated CNN to 13% for the 4-layer LSTM. The remainder of this section provides a description of the individual tasks and models with details in the Appendix.

4.1 MultiNLI

The MultiNLI dataset (Williams et al., 2018) contains crowd sourced textual entailment annotations across five diverse domains for training and an additional five domains for testing. Our model is a re-implementation of the ESIM sequence model (Chen et al., 2017). It first uses a biLSTM to encode the premise and hypothesis, then computes an attention matrix followed by a local inference layer, another biLSTM inference composition layer, and finally a pooling operation before

---

[5] While the CNN and Transformer implementations are reasonably well optimized, the LSTM biLM is not as it does not use an optimized CUDA kernel due to the use of the projection cell.

[6] Generally speaking, we found adding pre-trained GloVe vectors in addition to the biLM representations provided a small improvement across the tasks.
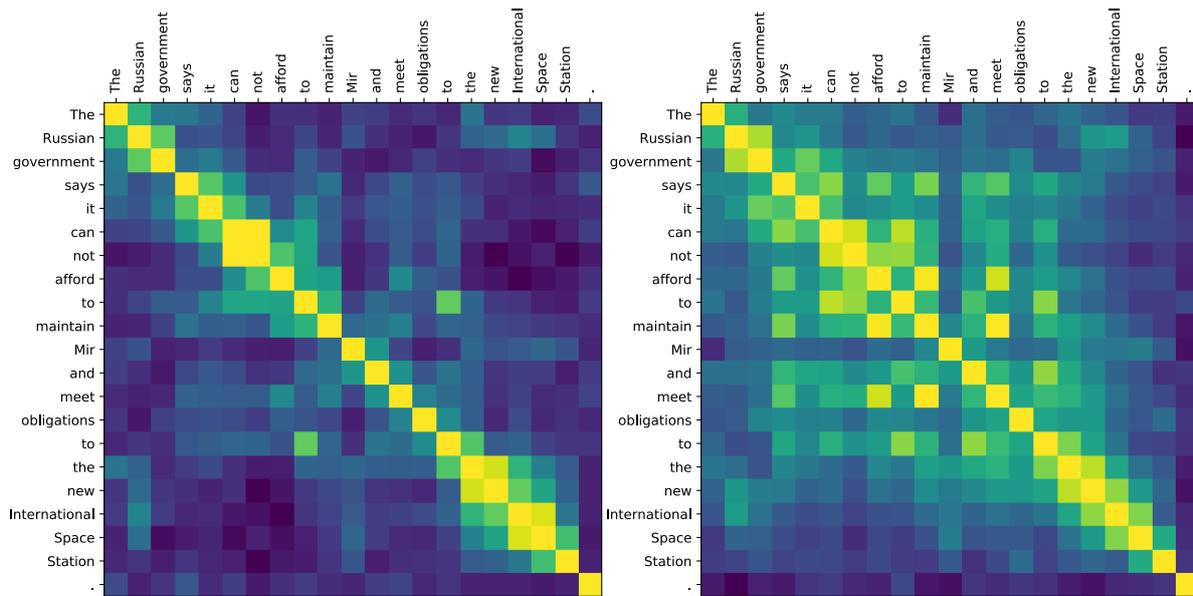
Figure 1: Visualization of contextual similarity between all word pairs in a single sentence using the 4-layer LSTM. The left panel uses context vectors from the bottom LSTM layer while the right panel uses the top LSTM layer. Lighter yellow-colored areas have higher contextual similarity.

the output layer. With the 2-layer LSTM ELMo representations, it is state-of-the-art for SNLI (Peters et al., 2018). As shown in Table 2, the LSTMs perform the best, with the Transformer accuracies 0.2% / 0.6% (matched/mismatched) less then the 2-layer LSTM. In addition, the contextual representations reduce the matched/mismatched performance differences showing that the biLMs can help mitigate domain effects. The ESIM model with the 4-layer LSTM ELMo-like embeddings sets a new state-of-the-art for this task, exceeding the highest previously published result by 1.3% matched and 1.9% mismatched from Gong et al. (2018).

4.2 Semantic Role Labeling

The Ontonotes 5.0 Dataset (Pradhan et al., 2013) contains predicate argument annotations for a variety of types of text, including conversation logs, web data, and biblical extracts. For our model, we use the deep biLSTM from He et al. (2017) who modeled SRL as a BIO tagging task. With ELMo representations, it is state-of-the-art for this task (Peters et al., 2018). For this task, the LSTM based word representations perform the best, with absolute improvements of 0.6% of the 4-layer LSTM over the Transformer and CNN.

4.3 Constituency parsing

The Penn Treebank (Marcus et al., 1993) contains phrase structure annotation for approximately 40k sentences sourced from the Wall Street Journal. Our model is the Reconciled Span Parser (RSP; Joshi et al., 2018), which, using ELMo representations, achieved state of the art performance for this task. As shown in Table 2, the LSTM based models demonstrate the best performance with a 0.2% and 1.0% improvement over the Transformer and CNN models, respectively. Whether the explicit recurrence structure modeled with the biLSTM in the RSP is important for parsing is explored in Sec. 5.3.

4.4 Named entity recognition

The CoNLL 2003 NER task (Sang and Meulder, 2003) provides entity annotations for approximately 20K sentences from the Reuters RCV1 news corpus. Our model is a re-implementation of the state-of-the-art system in Peters et al. (2018) with a character based CNN word representation, two biLSTM layers and a conditional random field (CRF) loss (Lafferty et al., 2001). For this task, the 2-layer LSTM performs the best, with averaged F_1 0.4% - 0.8% higher then the other biLMs averaged across five random seeds.
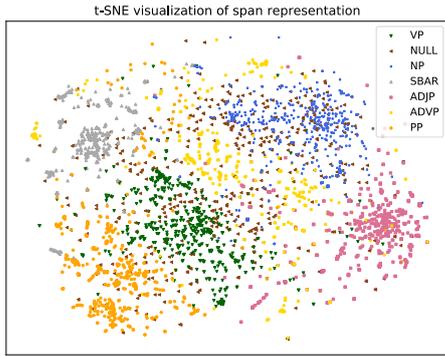
Figure 2: t-SNE visualization of 3K random chunks and 500 unlabeled spans ("NULL") from the CoNLL 2000 chunking dataset.

5 Properties of contextual vectors

In this section, we examine the intrinsic properties of contextual vectors learned with biLMs, focusing on those that are independent of the architecture details. In particular, we seek to understand how familiar types of linguistic information such as syntactic or coreferent relationships are represented throughout the depth of the network. Our experiments show that deep biLMs learn representations that vary with network depth, from morphology in the word embedding layer, to local syntax in the lowest contextual layers, to semantic relationships such as coreference in the upper layers.

We gain intuition and motivate our analysis by first considering the inter-sentence contextual similarity of words and phrases (Sec. 5.1). Then, we show that, in contrast to traditional word vectors, the biLM word embeddings capture little semantic information (Sec. 5.2) that is instead represented in the contextual layers (Sec. 5.3). Our analysis moves beyond single tokens by showing that a simple span representation based on the context vectors captures elements of phrasal syntax.

5.1 Contextual similarity

Nearest neighbors using cosine similarity are a popular way to visualize the relationships encoded in word vectors and we can apply a similar method to context vectors. As the biLMs use context vectors to pass information between layers in the network, this allows us to visualize how information is represented throughout the network.

Intra-sentence similarity Fig. 1 shows the intra-sentence contextual similarity between all pairs of words in single sentence using the 4-

layer LSTM.[7] From the figure, we make several observations. First, the lower layer (left) captures mostly local information, while the top layer (right) represents longer range relationships. Second, at the lowest layer the biLM tends to place words from the same syntactic constituents in similar parts of the vector space. For example, the words in the noun phrase "the new international space station" are clustered together, similar to "can not" and "The Russian government".

In addition, we can see how the biLM is implicitly learning other linguistic information in the upper layer. For example, all of the verbs ("says", "can", "afford", "maintain", "meet") have high similarity suggesting the biLM is capturing part-of-speech information. We can also see some hints that the model is implicitly learning to perform coreference resolution by considering the high contextual similarity of "it" to "government", the head of "it"'s antecedent span. Section 5.3 provides empirical support for these observations.

Span representations The observation that the biLM's context vectors abruptly change at syntactic boundaries suggests we can also use them to form representations of spans, or consecutive token sequences. To do so, given a span of S tokens from indices s_0 to s_1, we compute a span representation $\mathbf{s}_{(s_0,s_1),i}$ at layer i by concatenating the first and last context vectors with the element wise product and difference of the first and last vectors:

$$\mathbf{s}_{(s_0,s_1),i} = [\mathbf{h}_{s_0,i}; \mathbf{h}_{s_1,i}; \mathbf{h}_{s_0,i} \odot \mathbf{h}_{s_1,i}; \mathbf{h}_{s_0,i} - \mathbf{h}_{s_1,i}].$$

Figure 2 shows a t-SNE (Maaten and Hinton, 2008) visualization of span representations of 3,000 labeled chunks and 500 spans not labeled as chunks from the CoNLL 2000 chunking dataset (Sang and Buchholz, 2000), from the first layer of the 4-layer LSTM. As we can see, the spans are clustered by chunk type confirming our intuition that the span representations capture elements of syntax. Sec. 5.3 evaluates whether we can use these span representations for constituency parsing.

Unsupervised pronominal coref We hypothesize that the contextual similarity of coreferential mentions should be similar, as in many cases it is possible to replace them with their referent. If true, we should be able to use contextual similarity to perform unsupervised coreference reso-

---
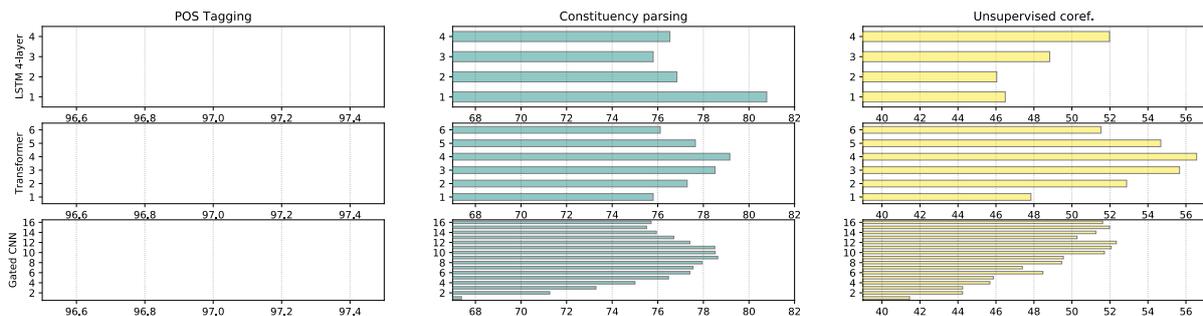
[7]See appendix for visualizations of the other models.

Figure 3: Various methods of probing the information stored in context vectors of deep biLMs. Each panel shows the results for all layers from a single biLM, with the first layer of contextual representations at the bottom and last layer at the top. From top to bottom, the figure shows results from the 4-layer LSTM, the Transformer and Gated CNN models. From left to right, the figure shows linear POS tagging accuracy (%; Sec. 5.3), linear constituency parsing (F_1; Sec. 5.3), and unsupervised pronominal coreference accuracy (%; Sec. 5.1).

Representation	Syntactic	Semantic
GloVe	77.9	**79.2**
n-gram hash	72.3	0.5
LSTM 4-layer	74.2	11.5
Transformer	**87.1**	48.8
Gated CNN	83.6	26.3

Table 3: Accuracy (%) for word vector analogies. In addition to the 300 dimension 840B GloVe vectors, the table contains results from a character n-gram hash and the context insensitive word embedding layer ($\mathbf{x}_k$) from the biLMs.
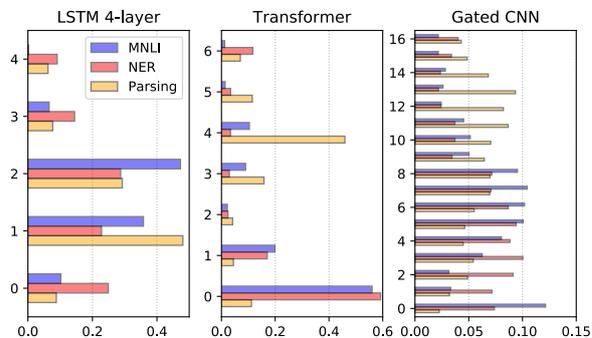

Figure 4: Normalized layer weights $\mathbf{s}$ for the tasks in Sec. 4. The vertical axis indexes the layer in the biLM, with layer 0 the word embedding $\mathbf{x}_k$.

lution. To test this, we designed an experiment as follows. To rule out trivially high mention-mention similarities due to lexical overlap, we restricted to pronominal coreference resolution. We took all sentences from the development set of the OntoNotes annotations in the CoNLL 2012 shared task (Pradhan et al., 2012) that had a third-person personal pronoun[8] and antecedent in the same sentence (904 sentences), and tested whether a system could identify the head word of the antecedent span given the pronoun location. In addition, by restricting to pronouns, systems are forced to rely on context to form their representation of the pronoun, as the surface form of the pronoun is uninformative. As an upper bound on performance, the state-of-the-art coreference model from Lee et al. (2017)[9] finds an antecedent span with the head word 64% of the time. As a lower bound on performance, a simple baseline that chooses the closest noun occurring before the pronoun has an ac-

curacy of 27%, and one that chooses the first noun in the sentence has an accuracy of 35%. If we add an additional rule and further restrict to antecedent nouns matching the pronoun in number, the accuracies increase to 41% and 47% respectively.

To use contextual representations to solve this task, we first compute the mean context vector of the smallest constituent with more then one word containing the pronoun and subtract it from the pronoun's context vector. This step is motivated by the above observation that local syntax is the dominant signal in the contextualized word vectors, and removing it improves the accuracies of our method. Then, we choose the noun with the highest contextual similarity to the adjusted context vector that occurs before the pronoun and matches it in number.

The right hand column of Fig. 3 shows the results for all layers of the biLMs. Accuracies for the

---

[8]he, him, she, her, it, them, they
[9]http://allennlp.org/models

1505

models peak between 52% and 57%, well above the baseline, with the Transformer overall having the highest accuracy. Interestingly, accuracies only drop 2-3% compared to 12-14% in the baseline if we remove the assumption of number agreement and simply consider all nouns, highlighting that the biLMs are to a large extent capturing number agreement across coreferent clusters. Finally, accuracies are highest at layers near the top of each model, showing that the upper layer representations are better at capturing longer range coreferent relationships then lower layers.

5.2 Context independent word representation

The word analogy task introduced in Mikolov et al. (2013) are commonly used as intrinsic evaluations of word vectors. Here, we use them to compare the word embedding layer from the biLMs to word vectors. The task has two types of analogies: syntactic with examples such as "bird:birds :: goat:goats", and semantic with examples such as "Athens:Greece :: Oslo:Norway". Traditional word vectors score highly on both sections. However, as shown in Table 3, the word embedding layer $\mathbf{x}_k$ from the biLMs is markedly different with syntactic accuracies on par or better then GloVe, but with very low semantic accuracies. To further highlight this distinction, we also computed a purely orthographically based word vector by hashing all character 1, 2, and 3-grams in a word into a sparse 300 dimensional vector. As expected, vectors from this method had near zero accuracy in the semantic portion, but scored well on the syntactic portion, showing that most of these analogies can be answered with morphology alone. As a result, we conclude that the word representation layer in deep biLMs is only faithfully encoding morphology with little semantics.

5.3 Probing contextual information

In this section, we quantify some of the anecdotal observations made in Sec. 5.1. To do so, we adopt a series of linear probes (Belinkov et al., 2017) with two NLP tasks to test the contextual representations in each model layer for each biLM architecture. In addition to examining single tokens, we also depart from previous work by examining to what extent the span representations capture phrasal syntax.

Our results show that all biLM architectures learn syntax, including span-based syntax; and part-of-speech information is captured at lower layers then constituent structure. When combined with the coreference accuracies in Sec. 5.1 that peak at even higher layers, this supports our claim that deep biLMs learn a hierarchy of contextual information.

POS tagging   Peters et al. (2018) showed that the contextual vectors from the first layer of the 2-layer LSTM plus a linear classifier was near state-of-the-art for part-of-speech tagging. Here, we test whether this result holds for the other architectures. The second row of Fig. 3 shows tagging accuracies for all layers of the biLMs evaluated with the Wall Street Journal portion of Penn Treebank (Marcus et al., 1993). Accuracies for all of the models are high, ranging from 97.2 to 97.4, and follow a similar trend with maximum values at lower layers (bottom layer for LSTM, second layer for Transformer, and third layer for CNN).

Constituency parsing   Here, we test whether the span representations introduced in Sec. 5.1 capture enough information to model constituent structure. Our linear model is a very simple and independently predicts the constituent type for all possible spans in a sentence using a linear classifier and the span representation. Then, a valid tree is built with a greedy decoding step that reconciles overlapping spans with an ILP, similar to Joshi et al. (2018).

The third row in Fig. 3 shows the results. Remarkably, predicting spans independently using the biLM representations alone has F_1 of near 80% for the best layers from each model. For comparison, a linear model using GloVe vectors performs very poorly, with F_1 of 18.1%. Across all architectures, the layers best suited for constituency parsing are at or above the layers with maximum POS accuracies as modeling phrasal syntactic structure requires a wider context then token level syntax. Similarity, the layers most transferable to parsing are at or below the layers with maximum pronominal coreference accuracy in all models, as constituent structure tends to be more local than coreference (Kuncoro et al., 2017).

5.4 Learned layer weights

Fig. 4 plots the softmax-normalized layer weights $\mathbf{s}$ from each biLM, learned as part of the tasks in Sec. 4. The SRL model weights are omitted as they close to constant since we had to regularize them to stabilize training. For constituency parsing, $\mathbf{s}$ mirrors the layer wise linear parsing results,

with the largest weights near or at the same layers as maximum linear parsing. For both NER and MultiNLI, the Transformer focuses heavily on the word embedding layer, $\mathbf{x}_k$, and the first contextual layer. In all cases, the maximum layer weights occur below the top layers as the most transferable contextual representations tend to occur in the middle layers, while the top layers specialize for language modeling.

6 Related work

In addition to biLM-based representations, McCann et al. (2017) learned contextualized vectors with a neural machine translation system (CoVe). However, as Peters et al. (2018) showed the biLM based representations outperformed CoVe in all considered tasks, we focus exclusively on biLMs.

Liu et al. (2018) proposed using densely connected RNNs and layer pruning to speed up the use of context vectors for prediction. As their method is applicable to other architectures, it could also be combined with our approach.

Several prior studies have examined the learned representations in RNNs. Karpathy et al. (2015) trained a character LSTM language model on source code and showed that individual neurons in the hidden state track the beginning and end of code blocks. Linzen et al. (2016) assessed whether RNNs can learn number agreement in subject-verb dependencies. Our analysis in Sec. 5.1 showed that biLMs also learn number agreement for coreference. Kádár et al. (2017) attributed the activation patters of RNNs to input tokens and showed that a RNN language model is strongly sensitive to tokens with syntactic functions. Belinkov et al. (2017) used linear classifiers to determine whether neural machine translation systems learned morphology and POS tags. Concurrent with our work, Khandelwal et al. (2018) studied the role of context in influencing language model predictions, Gaddy et al. (2018) analyzed neural constituency parsers, Blevins et al. (2018) explored whether RNNs trained with several different objectives can learn hierarchical syntax, and Conneau et al. (2018) examined to what extent sentence representations capture linguistic features. Our intrinsic analysis is most similar to Belinkov et al. (2017); however, we probe span representations in addition to word representations, evaluate the transferability of the biLM representations to semantic tasks in addition to syntax tasks, and

consider a wider variety of neural architectures in addition to RNNs.

Other work has focused on attributing network predictions. Li et al. (2016) examined the impact of erasing portions of a network's representations on the output, Sundararajan et al. (2017) used a gradient based method to attribute predictions to inputs, and Murdoch et al. (2018) decomposed LSTMs to interpret classification predictions. In contrast to these approaches, we explore the types of contextual information encoded in the biLM internal states instead of focusing on attributing this information to words in the input sentence.

7 Conclusions and future work

We have shown that deep biLMs learn a rich hierarchy of contextual information, both at the word and span level, and that this is captured in three disparate types of network architectures. Across all architecture types, the lower biLM layers specialize in local syntactic relationships, allowing the higher layers to model longer range relationships such as coreference, and to specialize for the language modeling task at the top most layers. These results highlight the rich nature of the linguistic information captured in the biLM's representations and show that biLMs act as a general purpose feature extractor for natural language, opening the way for computer vision style feature re-use and transfer methods.

Our results also suggest avenues for future work. One open question is to what extent can the quality of biLM representations be improved by simply scaling up model size or data size? As our results have show that computationally efficient architectures also learn high quality representations, one natural direction would be exploring the very large model and data regime.

Despite their successes biLM representations are far from perfect; during training, they have access to only surface forms of words and their order, meaning deeper linguistic phenomena must be learned "tabula rasa". Infusing models with explicit syntactic structure or other linguistically motivated inductive biases may overcome some of the limitations of sequential biLMs. An alternate direction for future work combines the purely unsupervised biLM training objective with existing annotated resources in a multitask or semi-supervised manner.

References

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

T. Blevins, O. Levy, and L. Zettlemoyer. 2018. Deep RNNs Encode Soft Hierarchical Syntax. In *ACL*.

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

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

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *ACL*.

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

David Gaddy, Mitchell Stern, and Dan Klein. 2018. What's going on in neural constituency parsers? An analysis. In *NAACL*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *ACL workshop for NLP Open Source Software*.

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

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

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

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

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Vidur Joshi, Matthew Peters, and Mark Hopkins. 2018. Extending a parser to distant domains using a few dozen partially annotated examples. In *ACL*.

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

Ákos Kádár, Grzegorz Chrupala, and Afra Alishahi. 2017. Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics*, 43:761–780.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. In *ICLR workshop*.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In *ACL*.

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

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *ACL*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *ACL*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *EACL*.

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

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

Jiwei Li, Will Monroe, and Daniel Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv*, abs/1612.08220.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. In *TACL*.

Liyuan Liu, Xiang Ren, Jingbo Shang, Jian Peng, and Jiawei Han. 2018. Efficient contextualized representation: Language model pruning for sequence labeling. In *EMNLP*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

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

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

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing lstm language models. In *ICLR*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

W. James Murdoch, Peter J. Liu, and Bin Yu. 2018. Beyond word importance: Contextual decomposition to extract interactions from lstms. In *ICLR*.

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

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

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

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

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

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

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Hasim Sak, Andrew W. Senior, and Franoise Beaufays. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH*.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task chunking. In *CoNLL/LLL*.

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

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

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *ICML*.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *AAAI*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*.

Matthew D. Zeiler. 2012. ADADELTA: An adaptive learning rate method. *arXiv*, abs/1212.5701.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833.

Kelly Zhang and Samuel R. Bowman. 2018. Language modeling teaches you more syntax than translation does: Lessons learned through auxiliary task analysis. *arXiv preprint*.