Language Modeling Teaches You More Syntax than Translation Does: Lessons Learned Through Auxiliary Task Analysis

Kelly W. Zhang[1]
kz918@nyu.edu

Samuel R. Bowman[1,2,3]
bowman@nyu.edu

[1]Dept. of Computer Science
New York University
60 Fifth Avenue
New York, NY 10011

[2]Center for Data Science
New York University
60 Fifth Avenue
New York, NY 10011

[3]Dept. of Linguistics
New York University
10 Washington Place
New York, NY 10003

1 Introduction

Recently, researchers have found that deep LSTMs (Hochreiter and Schmidhuber, 1997) trained on tasks like machine translation learn substantial syntactic and semantic information about their input sentences, including part-of-speech (Belinkov et al., 2017a,b; Blevins et al., 2018). These findings begin to shed light on why pretrained representations, like ELMo and CoVe, are so beneficial for neural language understanding models (Peters et al., 2018; McCann et al., 2017). We still, though, do not yet have a clear understanding of how the choice of pretraining objective affects the type of linguistic information that models learn. With this in mind, we compare four objectives—language modeling, translation, skip-thought, and autoencoding—on their ability to induce syntactic and part-of-speech information, holding constant the quantity and genre of the training data, as well as the LSTM architecture.

2 Methodology

We control for the data domain by exclusively training on datasets from WMT 2016 (Bojar et al., 2016). We train models on all tasks using the parallel En-De corpus, which allows us to make fair comparisons across all tasks. We also augment the parallel data with a large monolingual corpus from WMT to examine how the performance of the unsupervised tasks scales with more data.

We analyze representations learned by language models and by the *encoders* of sequence-to-sequence models.[1] Following Belinkov et al. (2017a), after pretraining, we fix the LSTM model parameters and use the hidden states to train auxiliary classifiers on several probing tasks. We use two syntactic evaluation tasks: part-of-speech (POS) tagging on Penn Treebank WSJ (Marcus et al., 1993) and Combinatorial Categorical Grammar (CCG) supertagging on CCG Bank (Hockenmaier and Steedman, 2007). CCG supertagging allows us to measure the degree to which models learn syntactic structure above the word. We also measure how much LSTMs simply memorize input sequences with a word identity prediction task.

3 Results

Comparing Pretraining Tasks For all pretraining dataset sizes, bidirectional language model (BiLM) and translation encoder representations outperform skip-thought and autoencoder representations on both POS and CCG tagging. Translation encoders, however, slightly underperform BiLMs, even when both models are trained on the same amount of data. Furthermore, BiLMs trained on the smallest amount of data (1 million sentences) outperform models trained on all other tasks using larger dataset sizes (5 million sentences for translation, and 63 million sentences for skip-thought and autoencoding). Especially since BiLMs do not require aligned data to train, the superior performance of BiLM representations on these tasks suggests that BiLMs (like ELMo) are better than translation encoders (like CoVe) for transfer learning of syntactic information.

Untrained Baseline Surprisingly, we find that the untrained LSTM baseline—frozen after random initialization—performs quite well on syntactic tagging tasks (a few percentage points behind BiLMs) when using all auxiliary data; however, decreasing the amount of *classifier* training data leads to a significantly greater drop in the untrained encoder performance compared to trained encoders. We hypothesize that the classifiers can recover neighboring word identity information—

[1]All our encoders are 2-layer, bidirectional LSTMs (500-D in each direction)—except for our large *forward* language models, which are 1000-D and unidirectional.

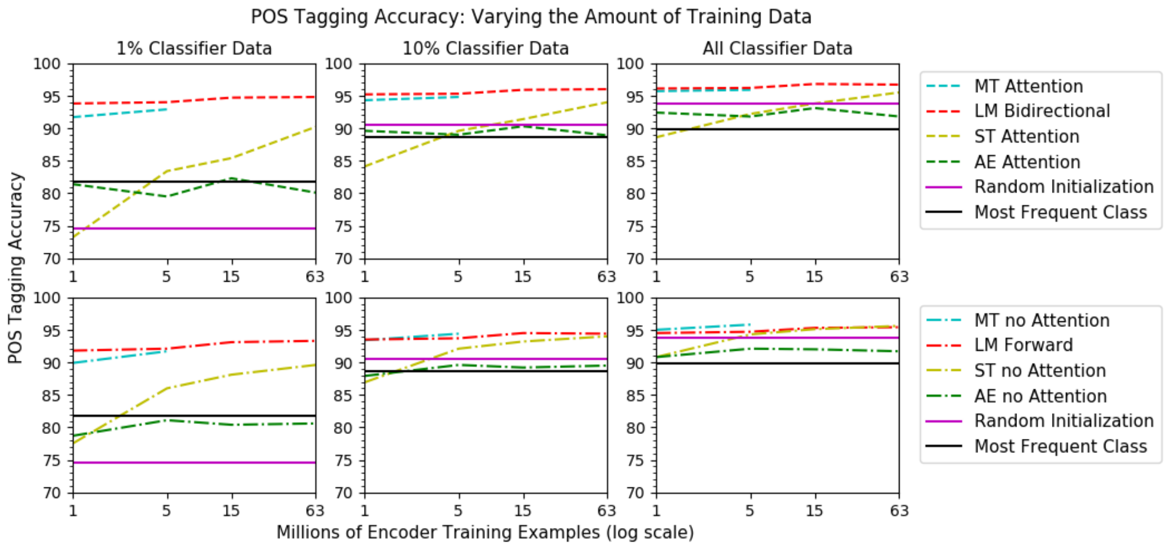

Figure 1: POS accuracies when training on different amounts of encoder and classifier data. We show results for the best performing layer of each model. The most frequent class baseline is word-conditional.

even from untrained LSTMs representations—and thus perform well on tagging tasks by memorizing word configurations and their associated tags from the training data. We test this hypothesis directly with the word identity task.

Word Identity For this task, we train classifiers to take LSTM hidden states and predict the identities of the words from different time steps. For example, for the sentence "I love NLP ." and a time step shift of -2, we would train the classifier to take the hidden state for "NLP" and predict the word "I". While trained encoders outperform untrained ones on both POS and CCG tagging, we find that all trained LSTMs *underperform* untrained ones on word identity prediction. This finding confirms that trained encoders genuinely capture substantial syntactic features, beyond mere word identity, that the auxiliary classifiers can use.

Effect of Depth Belinkov et al. (2017a) find that, for translation models, the first layer consistently outperforms the second on POS tagging. We find that this pattern holds for all our models, except BiLMs, where the first and second layers perform equivalently. This pattern occurs even in untrained models, which suggests that POS information is stored on the lower layer not necessarily because the training tasks encourage this, but due to properties of the deep LSTM architecture. For CCG supertagging though, the second layer performs better than the first in some cases (first layer performs best for untrained LSTMs). Which layer performs best appears to be independent of absolute performance on the supertagging task.

On word identity prediction, we find that for both trained and untrained models, the first layer outperforms the second layer when predicting the identity of the *immediate* neighbors of a word. However, the second layer tends to outperform the first at predicting the identity of more distant neighboring words. As is the case for convolutional neural networks, our findings suggest that depth in recurrent neural networks has the effect of increasing the "receptive field" and allows the upper layers to have representations that capture a larger context. These results reflect the findings of Blevins et al. (2018) that for trained models, upper levels of LSTMs encode more abstract syntactic information, since more abstract information generally requires larger context information.

4 Conclusion

By controlling for the genre and quantity of the training data, we make fair comparisons between several data-rich training tasks in their ability to induce syntactic information. Our results suggest that for transfer learning, bidirectional language models like ELMo (Peters et al., 2018) capture more useful features than translation encoders—and that this holds even on genres for which data is not abundant. Our work also highlights the interesting behavior of untrained LSTMs, which show an ability to preserve the contents of their inputs better than trained models.

References

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James R. Glass. 2017a. What do Neural Machine Translation Models Learn about Morphology? *ACL*.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017b. Evaluating Layers of Representation in Neural Machine Translation on Part-of-Speech and Semantic Tagging Tasks. *IJCNLP*.

Terra Blevins, Omer Levy, and Luke Zettlemoyer. 2018. Deep RNNs Learn Hierarchical Syntax. *ACL*.

Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 Conference on Machine Translation (WMT16). *ACL*.

Sepp Hochreiter and Jüergen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*.

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

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in Translation: Contextualized Word Vectors. *NIPS*.

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