An ELMo-inspired approach to SemDeep-5's Word-in-Context task

Alan Ansell
Department of Computer Science
University of Waikato, New Zealand

Felipe Bravo-Marquez
Department of Computer Science
University of Chile & IMFD

Bernhard Pfahringer
Department of Computer Science
University of Waikato, New Zealand

Abstract

This paper describes a submission to the Word-in-Context competition for the IJCAI 2019 SemDeep-5 workshop. The task is to determine whether a given focus word is used in the same or different senses in two contexts. We took an ELMo-inspired approach similar to the baseline model in the task description paper, where contextualized representations are obtained for the focus words and a classification is made according to the degree of similarity between these representations. Our model had a few simple differences, notably joint training of the forward and backward LSTMs, a different choice of states for the contextualized representations and a new similarity measure for them. These changes yielded a 3.5% improvement on the ELMo baseline.

1 Introduction

Traditional word embedding systems such as word2vec (Mikolov et al., 2013) assign each word a single fixed embedding. One weakness of these systems is that they are not well suited for representing words which have multiple meanings, as their embeddings are forced to occupy a point in the vector space which corresponds to some combination of these meanings. Much attention has been given to developing systems which can assign a word an embedding specific to the sense in which it is used in a given context (Huang et al., 2012; Neelakantan et al., 2014; Chen et al., 2014; Iacobacci et al., 2015; Li and Jurafsky, 2015; Peters et al., 2018, among others). Such a system could be thought of as yielding "sense embeddings" rather than word embeddings. Some sense embedding systems have shown advantages over traditional word embeddings, performing better on contextual word similarity tasks (Neelakantan et al., 2014; Chen et al., 2014, etc.) and relational similarity tasks (Iacobacci et al., 2015). One of the greatest successes has been the ELMo system (Peters et al., 2018) whose contextual embeddings were used to obtain state-of-the-art results on six NLP tasks.

The Word-in-Context (WiC) dataset (Pilehvar and Camacho-Collados, 2019) provides an opportunity to evaluate sense embedding systems by testing their ability to discriminate between finely-grained meanings of a word. Each instance in the dataset consists of two sentences which both contain a certain "focus" word. The instances must be classified according to whether the focus word is used in the same sense in the two sentences or not.

There are two main approaches to producing sense-specific embeddings. The first is to learn a number of embeddings for each word which correspond to its discrete senses, known as multi-prototype embeddings (Huang et al., 2012; Neelakantan et al., 2014; Chen et al., 2014; Iacobacci et al., 2015; Li and Jurafsky, 2015). The second is to dynamically create a unique embedding for a word for every context it appears in, which attempts to capture the particular shade of meaning the word has in that context. Notable examples of these "contextualized word embedding" systems include context2vec (Melamud et al., 2016) and ELMo (Peters et al., 2018).

The ELMo_1 baseline system in the task description paper (Pilehvar and Camacho-Collados, 2019) and our system can both be thought of as having three components: the LSTM-based (Hochreiter and Schmidhuber, 1997) language model; the "contextualization" component, in which contextualized embed-

dings for the focus words are obtained using the language model; and the "classification" component, where some similarity measure between the two contextualized embeddings is calculated and a positive classification is made if it is above a threshold learned on the training set.

In ELMo$_1$, the language model is as described in (Peters et al., 2018), the contextualized embeddings are the hidden states of the first LSTM layer at the focus word's position, and the similarity measure is cosine similarity. In sections 2.1, 2.2 and 2.3, we will describe how these three components operate in our system.

2 System Description

2.1 Language Model

The system uses a bidirectional LSTM-based language model. Instead of predicting the next or previous word given a left or right side context, the model predicts a missing word given both a left and right context - in this sense it is similar to context2vec. During training, two LSTM layers are run over a complete input sentence in both directions. The forward and backward directions are independent until the output layer, when they are used jointly as inputs to the softmax layer. Specifically the outputs of the second of two LSTM layers over a sentence of n words give a sequence of forward representations $\overrightarrow{\boldsymbol{u}}_1, \overrightarrow{\boldsymbol{u}}_2, ..., \overrightarrow{\boldsymbol{u}}_n \in \mathbb{R}^d$ and a sequence of backward representations $\overleftarrow{\boldsymbol{u}}_1, \overleftarrow{\boldsymbol{u}}_2, ..., \overleftarrow{\boldsymbol{u}}_n \in \mathbb{R}^d$. For each non-edge position $2 \leq i \leq n-1$ in the sentence, we define a vector $\boldsymbol{x}_i$ as $[\overrightarrow{\boldsymbol{u}}_{i-1} \overleftarrow{\boldsymbol{u}}_{i+1}]$, the concatenation of the forward representation in the preceding position and the backward representation in the following position. $\boldsymbol{x}_i$ is fed into a softmax layer over the vocabulary:

$$\boldsymbol{p}^{(i)} = \text{softmax}(W\boldsymbol{x}_i)$$

The objective is to maximise the predicted probability of the observed sentences, or equivalently minimise the cross entropy loss J:

$$J = -\sum_{i=2}^{n-1} \log p_{s_i}^{(i)},$$

where s_i is the index in the vocabulary of the word which appears in position i in the sentence.

2.2 Contextualization

Let f_1 and f_2 be the position of the focus word in sentences 1 and 2 of an example in the WiC dataset. Rather than using $\overrightarrow{\boldsymbol{u}}_f$ or $\overleftarrow{\boldsymbol{u}}_f$ as contextualized word representations for the focus word f, we instead use $\boldsymbol{x}_f$, the vector which would be used to predict the focus word. $\boldsymbol{x}_f$ is a representation of the expectations we have about the focus word given the context, and so we would expect $\boldsymbol{x}_{f_1}^{(1)}$ and $\boldsymbol{x}_{f_2}^{(2)}$ to differ significantly when the focus word is being used in a different sense.

2.3 Classification

We tried several similarity measures between $\boldsymbol{x}_{f_1}^{(1)}$ and $\boldsymbol{x}_{f_2}^{(2)}$, including dot product and cosine distance. The best-performing measure in our experiments was a weighted dot product

$$d(\boldsymbol{x}_{f_1}^{(1)}, \boldsymbol{x}_{f_2}^{(2)}) = \boldsymbol{w}^\top (\boldsymbol{x}_{f_1}^{(1)} \circ \boldsymbol{x}_{f_2}^{(2)})$$

where $\circ$ denotes element-wise product and $\boldsymbol{w}$ is a trainable weight vector. Learning $\boldsymbol{w}$ was treated as a logistic regression problem on the training set with feature vector $\boldsymbol{x}_{f_1}^{(1)} \circ \boldsymbol{x}_{f_2}^{(2)}$.

2.4 Corpus, Preprocessing and Hyperparameters

The model was trained on a 2018 Wikipedia dump. All tokens were lowercased. Those which appeared at least 300 times were included in the vocabulary, and others were replaced with <UNK>, resulting in a vocabulary size of ~100,000. The corpus was split into training examples at sentence boundaries, each example containing as many sentences as possible followed by padding to reach a 50 token limit. Each example was capped with a <START> and <END> token. The examples were randomly shuffled so that each batch would contain a diverse range of texts.

Each token in the vocabulary and <START>, <END> and <UNK> were assigned a randomly initialized 256-dimensional embedding. The LSTM cells had 2048 hidden units which were projected to a 256-dimensional output. There was no sharing of weights for the LSTM cells between layers or the two directions. There was an additional residual connection which fed the raw embeddings directly into the second LSTM layer.

The weight vector $\boldsymbol{w}$ had dimension 512 (as $\boldsymbol{x}$ is the concatenation of two LSTM output vectors), and the training set contained ~5,500 examples. $\boldsymbol{w}$ was fitted using Scikit-Learn's LogisticRegression with L2 regularization with parameter C ("inverse of regularization strength") set to 0.2. This value was obtained through tuning on the development set.

3 Results and Analysis

3.1 Results

A single submission was made to the competition during the evaluation period with parameters as described above, scoring 61.2% on the test set. This result and the results of a number of other system configurations are shown in Table 1.

Another submission made in the post-evaluation period attempted to improve on the original submission by using a model trained on a corpus consisting of the Wikipedia dump combined with a corpus of books, "BookCorpus" (Zhu et al., 2015). It also used stronger regularization when learning $\boldsymbol{w}$, setting $C = 0.02$. This submission scored 62.4%.

3.2 Analysis

Our system demonstrated significant improvement on the baseline models using relatively simple techniques. There are only a few significant differences between our system and the baseline model ELMo_1:

- Input is whole-token based rather than character based.
- Different training corpora - the ELMo version used was trained on the 1 Billion Word Benchmark.
- Forward and backward LSTMs are trained jointly.
- Contextualized embedding for a word is the vector used to predict the word rather than the output vector for the word's position in the sentence.
- Different similarity measure for contextualized embeddings.

We note that when ELMo_1's contextualization and classification methods (i.e. first layer hidden states, cosine similarity) are used with our trained language model, the test set accuracy is 54.9% compared with the 57.7% quoted in (Pilehvar and Camacho-Collados, 2019) for ELMo_1. This suggests that our system may have performed better with better language model implementation or training.

Comparing the "predictor" to the "hidden" states with cosine similarity, we see that this different choice of contextualized embedding is worth 4.2% on the test set.

While the use of weighted dot product was worth 7.2% compared to unweighted dot product on the dev set, this translated to only a 2.1% improvement on the test set, suggesting that some overfitting occurred when learning the dot product weights $\boldsymbol{w}$ despite the use of a regularization parameter fitted on the dev set. This may be because there is a greater degree of similarity between the train and dev sets than the train and test sets, as suggested in (Pilehvar and Camacho-Collados, 2019).

3.3 Limitations

There are a number of potential areas for improvements in our system:

- Since $\boldsymbol{x}_f$ is determined only by the context of the focus word, the focus word itself has no impact on the model's predictions. It seems as though it should be possible to improve performance by utilising knowledge about the focus word, but we did not manage to find a convincing method.
- The dataset contains many examples where the focus words in the two sentences share the same root but have different inflectional morphology, e.g. "break" and "breaks". This may cause some false negative classifications because the focus words having different tense or plurality is likely to result in differences in their $\boldsymbol{x}$ vectors in the dimensions relating to these features. Using the weighted dot product may alleviate this problem somewhat because it allows reduced weight to be assigned to dimensions which correspond to tense and plurality. A better solution however might be to preprocess the training corpus and all examples in the dataset to remove inflection entirely.

States	Similarity measure	Dev.	Test	Notes
Predictor	Weighted dot product	67.4	61.2	Submitted to competition
Predictor	Unweighted dot product	60.2	59.1	
Predictor	Cosine similarity	60.5	59.1	
Hidden	Cosine similarity	55.2	54.9	cf. ELMo_1 "threshold" version.
Hidden	Weighted dot product	54.1	53.1	

Table 1: Results with different system configurations. All results listed were obtained with the same training run of the language model trained on Wikipedia 2018. "Predictor" refers to the use of the $\boldsymbol{x}$ vectors used for predicting missing words, while "hidden" refers to the outputs of the first LSTM layer for both directions concatenated.

4 Conclusions and Future Work

We discovered several improved ways of using ELMo-type contextualized word embeddings to perform word sense disambiguation in the Word-in-Context task. When the forward and backward LSTMs were trained jointly, we found that it is better to use the concatenation of output states from the forward LSTM at the position before the focus and the backward LSTM at the position after the focus than it is to use hidden states from the focus position. We also found that a weighted dot product performs better than unweighted dot product or cosine similarity as a metric for determining whether two contextualized word embeddings refer to the same sense of the word or not. This suggests that some dimensions of such embeddings carry more information related to the human notion of word sense than others. Together these improvements yielded a 3.5% gain over the ELMo_1 baseline of (Pilehvar and Camacho-Collados, 2019), and there is reason to think that a better implemented language model could do even better.

A surprising aspect of our system is that it never looks at the focus word itself, only the context. Future work on this system might center on exploiting what we know about the focus word to improve performance.

5 Acknowledgements

Felipe Bravo-Marquez was funded by Millennium Institute for Foundational Research on Data.

References

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1025–1035, Doha, Qatar. Association for Computational Linguistics.

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

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 873–882, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. SensEmbed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 95–105, Beijing, China. Association for Computational Linguistics.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1722–1732, Lisbon, Portugal. Association for Computational Linguistics.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61.

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

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

of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1059–1069, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of NAACL*, Minneapolis, United States.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books.