Proceedings of the Federated Conference on Computer Science and Information Systems pp. 273–276
DOI: 10.15439/2019F120
ISSN 2300-5963 ACSIS, Vol. 18

Towards semantic-rich word embeddings

Grzegorz Beringer, Mateusz Jabłoński, Piotr Januszewski,
Andrzej Sobecki, Julian Szymański
Faculty of Electronic Telecommunications and Informatics
Gdańsk University of Technology, Gdańsk, Poland
Email: julian.szymanski@eti.pg.edu.pl

***Abstract*—In recent years, word embeddings have been shown to improve the performance in NLP tasks such as syntactic parsing or sentiment analysis. While useful, they are problematic in representing ambiguous words with multiple meanings, since they keep a single representation for each word in the vocabulary. Constructing separate embeddings for meanings of ambiguous words could be useful for solving the Word Sense Disambiguation (WSD) task.**

In this work, we present how a word embeddings average-based method can be used to produce semantic-rich meaning embeddings. We also open-source a WSD dataset that was created for the purpose of evaluating methods presented in this research.

I. Introduction

Word embedding methods, that map the vocabulary words to low-dimensional continuous space, have been widely applied to various natural language processing (NLP) problems. They are commonly used as the input representation of words, replacing high-dimensional one-hot encodings, and have been shown to improve the performance in tasks such as syntactic parsing[1] and sentiment analysis[2].

In word embedding methods, such as word2vec[3] or GloVe[4], each word in the vocabulary has exactly one representation. While it is enough for most words, it is problematic for ambiguous words, which can contain more than one meaning. For example, consider the following examples with the word *tree*, extracted from Wikipedia articles:

(a) *The olive, known by the botanical name Olea europaea, meaning "European olive", is a species of small* ***tree*** *in the family Oleaceae [...]*

(b) *Many theories of syntax and grammar illustrate sentence structure using phrase* ***trees****, which provide schematics of how the words in a sentence are grouped and relate to each other.*

(c) *Upon completion of listing all files and directories found,* ***tree*** *returns the total number of files and directories listed.*

All three sentences mention the word *tree* (or *trees*), but the meaning differs based on context - (a) means tree as a forest plant, (b) tree as a parse tree, (c) tree as a command in Unix systems.

For many applications, such as improving relevance of search engines, anaphora resolution or coherence, identifying which meaning is used, based on context, is important. This task is called Word Sense Disambiguation (WSD) and is an open problem in NLP domain. Word embeddings cannot be applied to WSD out-of-the-box, since they cannot differentiate between multiple meanings of an ambiguous word.

In this work, we propose a method to create semantically rich embeddings for each *keyword* (ambiguous word together with meaning, e.g. *tree (structure)*, *pool (computer science)*), by averaging embeddings of the ambiguous word and words describing its meaning. We evaluate this approach on a WSD task, gathered from Wikipedia articles (III). Finally, we discuss our results and propose future work (V-A).

II. Related work

There have been many methods of creating semantically meaningful word representations. Global matrix factorization methods, such as latent semantic analysis (LSA)[5], use matrix factorization to perform dimensionality reduction on a large term-frequency matrix, that captures statistical information about the corpus. As the result, we receive word and document embeddings, which are parametrized by the number of topics we want to extract from the documents, and which can be used to find similarities between different words and documents.

Other approach to creating word embeddings is to take only local context into account, without using global statistics. Example of this is word2vec[3], where a shallow neural network is trained to either predict context words based on the current word (skig-gram), or predict current word based on context words (continuous bag-of-words). Continuous representations of words are then extracted from the hidden layer of the trained network. FastText[6] improves upon skip-gram method, by representing each word as a bag of character n-grams, which provides more flexibility and has an added benefit of the ability to compute word representations for words unseen during training.

Global Vectors (GloVe)[4] combine both global matrix factorization and local context window methods, by training word vectors on co-occurence matrix, so that their differences predict co-occurence ratios.

Word embeddings can be also extracted from a trained language model[7]. Recently, methods like ELMo[8] or BERT[9] were shown to achieve great results in many NLP tasks. They produce deep contextualized word embeddings by using internal states of a trained language model pretrained on large corpus of text. Since models used are bidirectional (LSTM for ELMo, Transformer for BERT), the word embedding is conditioned on its left and right context, achieving flexible vector representations that could be used to disambiguate words.

Iacobacci et al.[10] were the first to try to use word embeddings for Word Sense Disambiguation. They consider four different strategies for integrating a pre-trained word embeddings as context representation in a supervised WSD system: concatenation, average, fractional and exponential decay of the vectors of the words surrounding a target word. Peters et al.[11] create word representations that differ from traditional word embeddings in that each token is assigned a representation that is a function of the entire input sentence. They use vectors derived from a bidirectional LSTM that is trained with a coupled language model objective on a large text corpus.

The most usual baseline for WSD task is the Most Frequent Sense[12] (MFS) heuristic, which selects for each target word the most frequent sense in the training data. Recent growth of sequence learning techniques using artificial neural networks contributed to WSD research: Raganato et al.[13] propose a series of end-to-end neural architectures directly tailored to the task, from bidirectional Long Short-Term Memory (LSTM) to encoder-decoder models. Melamud et al.[14] also use bidirectional LSTM it their work. They use large plain text corpora to learn a neural model that embeds entire sentential contexts and target words in the same low-dimensional space, which is optimized to reflect inter-dependencies between targets and their entire sentential context as a whole.

III. Dataset

For the purpose of constructing semantic-rich word embeddings, we manually gathered usage examples for 6 ambiguous words, 4 to 7 meanings each (28 meanings in total). Ambiguous word together with its meaning constitues a *keyword*, which we use as a separate class when identyfing the closest meaning given some context.

We chose ambiguous words based on the number and variety of meanings it had. Meanings themselves were chosen to cover a range of topics (e.g. *tree (forest)*, *tree (family)*, *trees (folk band)*, *tree (command)*). We also tried to look for meanings that are semantically related and can occur in similar context (and in turn be difficult for the model to differentiate between), e.g. *tree (structure)*, *tree (parse)*, *tree (decision)* or *nails (new wave band)*, *nails (hardcore punk band)*. Lastly, we added some keywords, that we suspected to be really underrepresented in the word embedding of the ambiguous word, e.g. *Mars* as the pop singer Bruno Mars (*mars (bruno singer)*) or *pool* as the computer science term (*pool (computer science)*).

Usage examples for keywords were gathered mostly from Wikipedia, using *What links here* utility, which lists all Wikipedia pages that link to a specific article. We used these links to search for usages of our keywords in context. We found that *What links here* utility has some limitations. Many articles linked to the keyword do not use that keyword in text at all or just list it in "See also" section, which does not provide good context around the keyword for the model to improve on. Moreover, some keywords do not have enough usage examples that can be found on Wikipedia alone. In such cases, other websites were used to find proper usage examples.

The dataset is split into training and test set, with 5 training and 10 test examples for each keyword. Each example is stored in plain text, with the ambiguous word marked with "*" on both sides. For simplicity, only one word is marked in each text, even if more ambiguous word usages can be found. In case we wanted to mark another word in the same text, we could just add the same example twice, with different words marked each time.

The correct keyword for each example, together with a path to file and a link, where the original text was taken from, are stored in CSV files: *train.csv* for training set, *test.csv* for test set (columns: path,keyword,link). Keywords themselves, together with links to their Wikipedia articles, are stored in *keywords.csv*.

Dataset, together with the code to execute experiments from this paper, can be found on our GitHub repository[1].

IV. Our method

Keyword is a sequence of words that is composed of the ambigous word and words describing its specific meaning, e.g. *tree (forest)* that represents tree as a plant (ambiguous word: *tree*, meaning: *forest*) and *tree (structure)* which represents tree as a mathematical structure (ambiguous word: *tree*, meaning: *structure*).

To get the embedding of the keyword, we average embeddings of all the words in the keyword:

$$\boldsymbol{k} = e(w_1, w_2, ..., w_N) = \frac{1}{N}\sum_{i=1}^{N} e(w_i) \tag{1}$$

where $e(\cdot)$ is the embedding function used and $w_1, w_2, ..., w_N$ is a sequence of N words that, in this case, constitutes a keyword.

Example for keyword *tree (forest)*:

$$\boldsymbol{k}_{tree(forest)} = e(tree, forest) = \frac{e(tree) + e(forest)}{2} \tag{2}$$

Context is a sequence of words, extracted from some text, which contains an ambiguous word and words surrounding it in text. It is parametrized by **context length** l, which specifies how many words from both sides of the ambiguous word are taken into consideration.

Context embedding $\boldsymbol{c}$ is also achieved by taking an average of word embeddings (Equation 1). In this case, $N = 2l + 1$ and $w_1, w_2, ..., w_N$ is the context with ambiguous word inside. For some cases $N < 2l + 1$, since the ambiguous word may occur at the beginning or end of text example and full context cannot be collected. In this case, we just average the reduced context.

The approach is to use keyword and context embeddings to find the closest keyword given some context, using cosine distance as a similarity metric.

[1] https://github.com/gberinger/automatic-wiki-links

Table I
RESULTS ACHIEVED ON THE TEST SET FOR OUR METHOD. COSINE DISTANCE IS MEASURED BETWEEN THE CORRECT KEYWORD AND CONTEXT EMBEDDINGS.

Metric	Our Model
Top-1 accuracy	67%
Top-2 accuracy	85%
Top-3 accuracy	93%

In other words, given an input text and marked ambiguous word within, we extract the context and compute its embedding $\boldsymbol{c}$. The keyword, whose embedding is closest to $\boldsymbol{c}$ w.r.t. cosine distance, is chosen as the ambiguous word's meaning.

V. RESULTS OF THE EXPERIMENTS

In our experiments we evaluate how semantic-rich keyword embeddings, perform for the dataset we collected (III), for the our approach. We use a pretrained embedding model from spaCy - *en_vectors_web_lg*, which contains 300-dimensional word vectors trained on Common Crawl with GloVe[2].

We compare results on the test set with top-k metrics ($k \in 1, 2, 3$), where we check, if the correct keyword is in closest k keywords given a specific context describe it. We focus mostly on top-1 accuracy, since we are interested if the word is correctly disambiguated.

Due to the high impact of training data order on test results, we take the average score of 30 runs (each with a random order of training data) for each optimization experiment. We evaluate the performance of the proposed model with different context lengths, to see how it affects top-k accuracies (Fig. 1).

We can see, that the model does relatively well. Top-3 accuracy is about 85-90%, which is probably caused by a low number of meanings for each ambiguous word. Top-1 accuracy for shorter context lengths can go as high as 65% but decreases with longer contexts. As suspected (V-A), this is most likely due to the fact, that the average of many word embeddings may make some contexts similar to each other, therefore making it harder to distinguish between some meanings.

The best result w.r.t. top-1 accuracy was achieved with $l = 3$, which is why we choose this context length as a starting point for next experiments.

Performance on the test set can be seen in Table I. All metrics improved due to the optimization process of moving correct keywords closer to (and incorrect keywords away from) contexts found in the training set. High top-2 and top-3 accuracies suggest, that the correct keyword is usually relatively close to the context describing it.

It is important to note, that the performance might worsen, if we expand the keyword vocabulary to large-scale experiments, where we have much more possible keywords than 28.

A. Discussion and future work

We are aware that our method has some limitations. First of all, it may be impossible to achieve the optimal solution, as we can only optimize keyword embeddings, leaving context embeddings fixed in the multidimensional space. Therefore, it is possible that contexts for specific keyword overlap on contexts for other keyword.

Secondly, the average context embedding may be ambiguous, with a high possibility of two different context being mapped to a similar point in space, especially for longer context lengths. In future work we plan to experiment with different sequence embedding techniques, that might be better suited for this purpose than a flat average.

Finally, we run experiments for a very small number of ambiguous words and meanings. Our method could have problems with a bigger dataset, since it would be much more difficult to separate different keywords.

Constructing semantic-rich embeddings for ambiguous words, by taking the average of embeddings of the ambiguous word and words describing its meaning, and then comparing it with the average embedding of context words describing given keyword, proved to be a surprisingly good approach for the task of disambiguation on the dataset of 28 keywords we collected (III). Our method achieved 67% top-1, 85% top-2 and 93% top-3 accuracy for context length $l = 3$. Longer context lengths were shown to decrease the accuracy, since the average of many word embeddings may result in similar embeddings for different contexts.

Further improvements could be sought by using different keyword and context embedding schemes, e.g. weighted average or by using some sentence embedding method. Optimization method itself could be made more stable by applying decay to alpha and beta parameters and by using a validation set for early stopping. It could also be bound to cosine distance between the keyword and context - the bigger the difference, the bigger the update.

It would also be interesting to see, how the suggested approach for constructing semantic-rich embeddings would perform on a large-scale dataset. Such a dataset could be automatically collected from Wikipedia, using disambiguation pages to find ambiguous words and their meanings, and *What links here* utility, to find usage examples for each keyword.

We assume, that the performance of the our model (and thus the quality of keyword embeddings) can be improved, if we provide examples of contexts that the specific keyword appears in. This can be reached by moving keyword embeddings closer to embeddings of contexts they appear in, so for each training example, the correct keyword embedding is shift by a given factor, in the direction of the context embedding, which describes said keyword. Further optimizations can be done, by moving top-k closest keywords that are incorrect given the same context.

VI. ACKNOWLEDGEMENTS

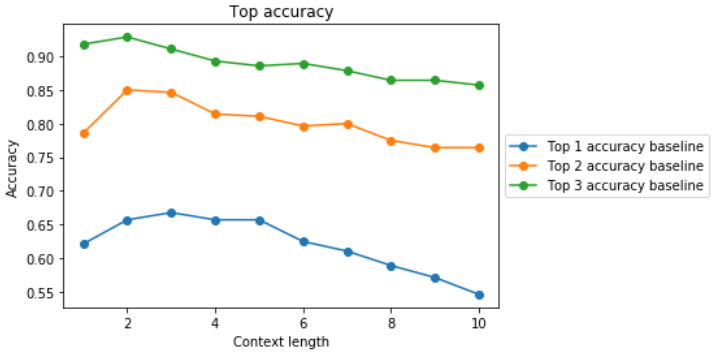
The work has been partially supported by funds of Faculty of Electronics, Telecommunications and Informatics of Gdańsk University of Technology.

REFERENCES

[1] R. Socher, J. Bauer, C. D. Manning, and N. Andrew Y., "Parsing with compositional vector grammars," in *Proceedings of the 51st Annual*

[2]https://spacy.io/models/en#section-en_vectors_web_lg

Figure 1. Top-k accuracies of the our model with different context lengths.



Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, 2013, pp. 455–465. [Online]. Available: http://aclweb.org/anthology/P13-1045

[2] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2013, pp. 1631–1642. [Online]. Available: http://aclweb.org/anthology/D13-1170

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1301.html#abs-1301-3781

[4] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *In EMNLP*, 2014.

[5] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, vol. 41, no. 6, pp. 391–407, 1990.

[6] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *arXiv preprint arXiv:1607.04606*, 2016.

[7] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Mar. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944966

[8] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," feb 2018. [Online]. Available: http://arxiv.org/abs/1802.05365

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," oct 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[10] I. Iacobacci, M. T. Pilehvar, and R. Navigli, "Embeddings for word sense disambiguation: An evaluation study," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2016, pp. 897–907. [Online]. Available: http://aclweb.org/anthology/P16-1085

[11] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "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)*. Association for Computational Linguistics, 2018, pp. 2227–2237. [Online]. Available: http://aclweb.org/anthology/N18-1202

[12] A. Raganato, J. Camacho-Collados, and R. Navigli, "Word sense disambiguation: A unified evaluation framework and empirical comparison," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, 2017, pp. 99–110. [Online]. Available: http://aclweb.org/anthology/E17-1010

[13] A. Raganato, C. Delli Bovi, and R. Navigli, "Neural sequence learning models for word sense disambiguation," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2017, pp. 1156–1167. [Online]. Available: http://aclweb.org/anthology/D17-1120

[14] O. Melamud, J. Goldberger, and I. Dagan, "context2vec: Learning generic context embedding with bidirectional lstm," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Association for Computational Linguistics, 2016, pp. 51–61. [Online]. Available: http://aclweb.org/anthology/K16-1006