Acronym Expander at SDU@AAAI-21: an Acronym Disambiguation Module

João L. M. Pereira,[1] Helena Galhardas, [1] Dennis Shasha [2]
[1] INESC-ID and Instituto Superior Técnico, Universidade de Lisboa
[2] Courant Institute, NYU
joaoplmpereira@tecnico.ulisboa.pt, helena.galhardas@tecnico.ulisboa.pt, shasha@cs.nyu.edu

Abstract

In order to properly determine which of several possible meanings an acronym A in sentence s has, any system that aims to find the correct meaning for A must understand the context of s.

This paper describes the techniques we use for that problem for the SDU@AAAI benchmark in which context was provided in the form of sentences in which acronym A is present and defined.

As a capsule summary of our results, Support Vector Machines with Doc2Vec techniques achieves a higher Macro F1-Measure score than Cosine similarity with Classic Context Vector techniques. Although these techniques usually work better with documents (i.e., many sentences rather than the one sentence offered in this benchmark), they achieved scores of Macro F1-Measure 86-89%.

While these results were 5.65% worse than the best in the benchmark experiment, the high speed of our approach (max 0.6 seconds on average per sentence on a virtual machine allocated with 4 CPU cores and 32GB of RAM in a shared server) and the possibility that our methods are complementary to those of other groups may lead to high performance hybrid systems.

Introduction

The proper expansion of an acronym depends on context. For example, "HD" can mean Harmonic Distortion in a signal context, High Definition in a video context, and Huntington 's Disease in a medical context. Thus, any system that hopes to help readers understand the intended meaning of an undefined acronym in a sentence must expand that acronym using its context.

An acronym expander system comprises the following steps: (i) Extraction of both acronym and (when present) its expansion within a text. For example, if a given text has "HD (High Definition)" then HD would be the acronym and High Definition would be the expansion. We call this *in-expansion* because it can be done for a particular article on its own. (ii) In the case that an acronym is not expanded in a text, *out-expansion* chooses an expansion from a previously large parsed corpus (training corpus) like Wikipedia[1]. The choice of which of several possible expansions to choose is based on some notion of article domain similarity between the text with a non-expanded acronym A and the articles containing expansions for A. We participated in the SDU@AAAI benchmark presented in Veyseh et al. (2020) that tests out-expansion only (i.e., *acronym disambiguation*).

Our system includes two techniques for out-expansion: Cosine similarity of Classic Context Vectors (Abdalgader and Skabar 2012; Prokofyev et al. 2013; Li, Ji, and Yan 2015) and Doc2Vec (Le and Mikolov 2014) whose outputs are used as features for Support Vector Machines (SVMs) to create a new out-expansion technique. Moreover, we used Wikipedia articles to enrich the training data for these techniques. Our results show that Doc2Vec together with Support Vector Machines (SVMs) gives the best prediction results when using Wikipedia data. Without extra data, context vector works best.

Related Work

To our knowledge, systems that expand abbreviations and/or acronyms use a pre-defined dictionary of acronym-expansions (Gooch 2012; ABBREX[2]) as opposed to trying to discover the proper expansion based on context.

Ciosici and Assent (2018) proposed an abbreviation/acronym expansion system architecture that performs out-expansion. Unfortunately, their demo paper does not provide enough technical details and their code is proprietary.

The remaining part of this section describes previous work on out-expansion.

Li, Ji, and Yan (2015) proposed two approaches to out-expansion based on word embeddings from Word2Vec (Mikolov et al. 2013a) to address the out-expansion problem. Their best approach, called Surrounding Based Embedding (SBE), combines the Word2Vec embeddings of the words surrounding the acronym or the expansion. Sim-

[1]https://www.wikipedia.org/
[2]http://abbrex.com/

ilarly to SBE, Ciosici, Sommer, and Assent (2019) proposed Unsupervised Acronym Disambiguation (UAD) that replaces each expansion occurrence in the text collection by a normalized token and retrains the Word2Vec google news model (Mikolov et al. 2013a) on that collection. The resulting model produces an embedding for each normalized token, i.e., an expansion embedding.

Thakker, Barot, and Bagul (2017) creates document vector embeddings using Doc2Vec for each document. For each set of documents D containing an expansion for an acronym A, the system trains a Doc2Vec model on D which is used to infer the embedding for an input document i containing an undefined acronym A.

Charbonnier and Wartena (2018) proposed an out-expansion approach based on Word2Vec embeddings weighted by Term Frequency-Inverse Document Frequency (TF-IDF) scores to find out-expansions for acronyms in scientific article captions.

More recently, Pouran Ben Veyseh et al. (2020) compare previous works in a new dataset (i.e., the Acronym Disambiguation dataset used in SDU@AAAI competition). The authors also propose a new model called Graph-based Acronym Disambiguation (GAD). GAD uses word and sentence representations obtained from Bidirectional Long Short-Term Memory (BiLSTM) neural network. Those representations are complemented by using syntactic structure from a dependency tree graph to model far but important dependencies between words using a Graph Convolutional Neural networks (GCN) (Kipf and Welling 2017). Finally, a two layer feedforward neural network classifier is used to guess the expansion.

A related line of work explored the expansion of acronyms in enterprise texts (Feng et al. 2009; Li et al. 2018). For instance, in Li et al. (2018), enterprise textual documents are used as training data as well as Wikipedia articles and a set of features like statistics based on word frequencies, words co-occurrences, and TF-IDF. Other works explored acronym disambiguation in biomedical domains (Pustejovsky et al. 2001; Pakhomov, Pedersen, and Chute 2005; Yu et al. 2006; Stevenson et al. 2009; Moon, Pakhomov, and Melton 2012; Moon, McInnes, and Melton 2015; Wu et al. 2015; 2017).

Less directly related, but insightful, is the literature on Word Sense Disambiguation (WSD) (Navigli 2009; Moro and Navigli 2015) because that work also must make use of the context around a token (in our case, an acronym; in the word sense literature, a word).

Out-Expansion Strategy

Our out-expansion strategy consists of: (i) a *Representator* to map an input sentence to a document representation that holds contextual information and (ii) an *Out-Expansion Predictor* to choose a context-appropriate out-expansion for each acronym found in the input sentence.

Representator

Representors summarize text (documents or sentences) in order to capture information signals about their semantics.

For the competition, we tested two representator techniques: Classic Context Vector and Doc2Vec.

Classic Context Vector The context vector technique is an unsupervised method used as a baseline in Word Sense Disambiguation problems (Abdalgader and Skabar 2012) and also in acronym disambiguation problems (Prokofyev et al. 2013)(Li, Ji, and Yan 2015). We denote it as classic to distinguish it from variants or other techniques that also provide vectors to contexts.

A *Context vector* represents a term (e.g, an acronym or expansion) by a vector based on the words that co-occur with the term in each document of the corpus containing that term. Thus, a context vector is a sparse vector where each position corresponds to a word in any document in the corpus, if the word is in a document that contains the term, then the vector position has some positive value, otherwise the value is zero. In the classic approach, the value at each vector position corresponds to the number of co-occurrences of the term and the co-occurring words in all the documents of the corpus.

In acronym disambiguation, the acronym in a particular sentence yields a context vector (which we call the "target context vector") which contains the words occurring in that sentence and their number of occurrences.

Each possible expansion for the acronym will have a context vector as well ("potential context vector"). Classic context vector chooses the expansion associated with the potential context vector that is most similar to the target context vector. The simplest similarity metric is cosine similarity. Figure 1 presents an example of a context vector for *Portable Document Format* expansion using two documents. For instance, words "the" and "file" occur one time in each document and so the positions reserved to these two words in the vector contains value 2 while the others contain 1.

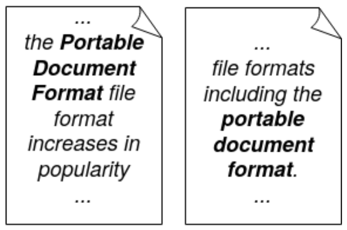
Doc2Vec *Doc2Vec* (Le and Mikolov 2014) is a document embedding and an unsupervised learning technique that adds the capability of automatically learning document (or paragraph) vectors to Word2Vec (Mikolov et al. 2013a). Given a list of words (e.g., a text document) as input, the output of Doc2Vec is a dense vector of real numbers (i.e., an embedding).

Just as Word2Vec assigns a vector to a word, Doc2Vec assigns a vector of N dimensions called a document vector to a document (or in the case of this benchmark to a sentence).

The training problem consists of finding the best set of embedding values for each word and document (i.e., Doc2Vec model parameters) that, given a document, predicts the set of words in that document. For example, consider a document consisting on a list containing the countries in Figure 2. If the document is known to the Doc2Vec model (i.e., it was included in the training data) then we have a document embedding available, otherwise, a document embedding d is computed by finding the best values that maximize the prediction of the country names given d.

In contrast to Word2Vec which averages word vectors to represent a particular document, Doc2Vec creates a trained vector for each document in the corpus (Dai, Olah, and Le 2015). By comparing those document vectors through co-

Documents containing the *Portable Document Format* expansion


...
the **Portable Document Format** file format increases in popularity
...
...
file formats including the **portable document format**.
...

Potential Context Vector for the *Portable Document Format* expansion

Words	the	file	format	increases	in	popularity	formats	including
Count	2	2	1	1	1	1	1	1

Figure 1: Classic context vector example for *portable document format* expansion.

sine similarity, Doc2Vec can infer semantically similar documents.

Out-Expansion Predictor

Out-expansion predictors select an out-expansion for a given acronym A in an input sentence ins. For this purpose, a predictor considers each sentence containing a valid expansion E for A. Sentences are characterized by the representator output explained in the previous section.

In the case of the Classic Context Vector, we compare the input sentence context vector with the vector resulting from summing the context vectors of the sentences for E. In this classical approach, because we have only a context vector per expansion E, we use cosine similarity to evaluate similarity.

We consider the use of machine learning classifiers as alternatives to cosine similarity when more than one training sample is possible for an expansion (i.e., label). This is the case of Doc2Vec whose embeddings represent a set of words (e.g., document or sentence) so we will have as many samples per expansion as set of words (e.g., sentences) where it occurs. However, for Classic Context Vector, it is not possible to use such machine learning approach because we have a context vector per expansion and so only one sample per expansion. Specifically, for the competition we used *Support Vector Machines (SVMs)* where non-binary classification was performed by a ”one-vs-all” approach where a binary SVM classifier predicts with a certain probability if a sample belongs to a particular class. The class with highest probability is selected. We used the LibLinear (Fan et al. 2008) implementation included in sckit-learn toolkit (Pedregosa et al. 2011).

SDU@AAAI Benchmark of Out-expansion Techniques

This section describes the SDU@AAAI benchmark used to evaluate the out-expansion techniques described in the previous section.

Datasets

The datasets that we use in this benchmark are:

SciAD contains sentences from human annotated scientific articles extracted from ArXiv [3]. Each sentence contains an acronym to disambiguate. This is the dataset provided for the SDU@AAAI Acronym Disambiguation (AD) competition and it was proposed in (Pouran Ben Veyseh et al. 2020). There are three data splits: (i) Train with 50,033 sentences, (ii) Dev with 6,188 sentences, and (iii) Test with 6,217 sentences where acronym expansion is unknown. This dataset also contains a dictionary with acronyms and their possible expansions.

Wikipedia contains all English articles of Wikipedia.org taken from the Wikipedia dump of March 1, 2020[4]. We used the WikiExtractor[5] software to obtain the articles in plain text, and we used the Schwartz and Hearst (2003) algorithm to extract acronyms and expansions from each Wikipedia article.

Data Preparation

We process the datasets by removing punctuation and normalizing tokens in order to create a better textual representation. That is, we perform the following operations on each dataset:

SciAD We remove non alphanumeric tokens, punctuation characters, and stop-words. Then, we transform each token to its stem, e.g. expander, expanding, and expanded all map to expand. We use the Porter Stemmer algorithm from the Natural Language Toolkit (NLTK) (Bird, Klein, and Loper 2009) for that purpose.

Wikipedia Because expansions in Wikipedia may be written in different formats and with plurals, we normalize the expansions found against the dictionary of acronym

[3]https://arxiv.org/

[4]https://dumps.wikimedia.org/enwiki

[5]http://medialab.di.unipi.it/wiki/Wikipedia_Extractor

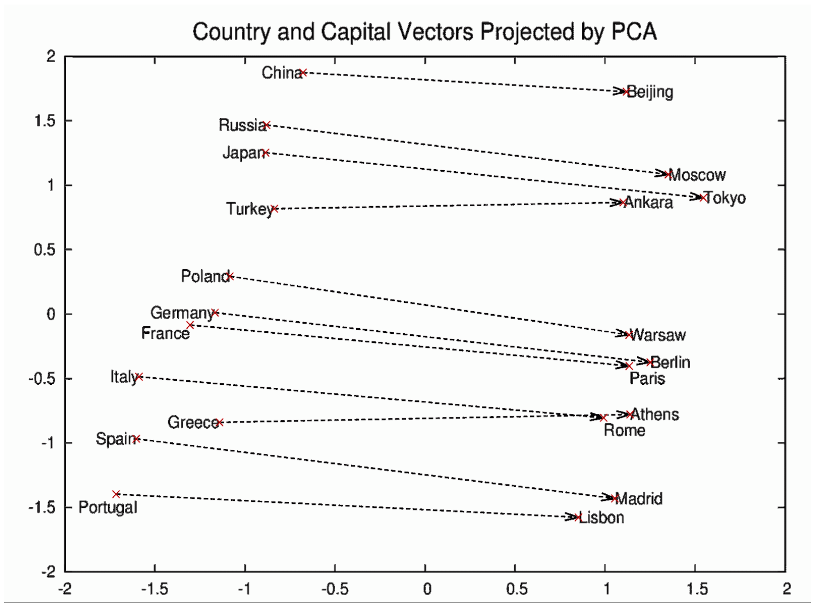


Figure 2: Countries and Capitals vectors. Modified from (Mikolov et al. 2013b).

expansions shared with SciAD. So, each expansion in the Wikipedia documents is replaced by the closest expansion in the SciAD dictionary. Distance is given by comparing the expansion in Wikipedia against a SciAD expansion, if the first 4 characters of each word are equal we consider the expansions to be equal (distance=0); distance is given by the edit-distance between both expansions, if the edit-distance is below 3 then the expansions are close enough, otherwise they are considered two distinct expansions. Wikipedia expansions not close to any expansion in the SciAD dictionary and their corresponding documents are not considered for prediction because only the expansions in the dictionary are valid for the SciAD evaluation set. Furthermore, while keeping the expansions in text, we apply the tokenizer from NLTK and remove the non alphanumeric tokens, punctuation characters, and stopwords. Finally, we transform each token to its stem as we did for the SciAD dataset.

Out-expansion Techniques

For the SDU@AAAI AD competition, we test the following out-expansion techniques: we use (i) the Cosine similarity (Cossim) with the Classic Context Vector (Li, Ji, and Yan 2015) and (ii) the outputs of Doc2Vec as features for Support Vector Machines (SVMs).

Prediction and Performance Metrics

For the SDU@AAAI competition, we use the following metrics:

Out-expansion Macro Averages: the average of the Precision, Recall and F1-Measure for each expansion. These are the official metrics used in the SDU@AAAI competition, being the Macro F1-Measure used to rank the competitors based on expansion prediction quality.

Training execution times: the execution time to create the representator model based on the training sentences and/or documents.

Average execution times per sentence: the average execution time to predict the expansions for the acronym in a sentence.

Experimental Evaluation

This section reports on the out-expansion experiments.

We run the experiments on a machine with the following specifications: Virtual Machine (VM) with 4 CPU cores from an Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz, 32GiB of RAM (Random Access Memory), and Ubuntu 18.04.3 LTS.

Out-Expansions on the AAAI Benchmark

In this section, we report the results obtained using the SciAD dataset. We used the Train set and Dev set as test data to tune the hyperparameters of Doc2Vec and SVMs. Then, we used the Train and Dev sets as training data and the Test set as evaluation. Evaluation quality measures were provided by the Codelab competition evaluation system[6]. We have submitted two combinations of out-expander predictors and representators: (i) cosine similarity (Cossim) as predictor with Classic Context Vector as representator, and (ii) Support Vector Machine (SVM) as predictor with Doc2Vec as representator.

In Table 1, we report the out-expansion macro averages for predicting expansions for acronyms in sentences: Precision (P), Recall (R), and F1-measure (F1). Macro F1-measure is the official measure for ranking competitors in

[6]https://competitions.codalab.org/competitions/26611

Acronym out-expansion technique		Dev macro avg.			Test macro avg.			Execution times	
Predictor	Representator	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)	Training (s)	Avg. per sentence (s)
Cossim	Classic Context Vector	90.00%	84.68%	87.26%	92.13%	84.16%	87.96%	1.24	0.18
SVM	Doc2Vec	90.49%	81.01%	85.48%	91.95%	80.15%	85.65%	92.79	0.09

Table 1: Out-expansion macro averages and execution times for training and average per sentence for SciAD dataset.

Acronym out-expansion technique		Dev macro avg.			Test macro avg.			Execution times	
Predictor	Representator	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)	Training (s)	Avg. per sentence (s)
Cossim	Classic Context Vector	88.24%	82.79%	85.43%	90.27%	83.73%	86.88%	504.52	0.61
SVM	Doc2Vec	91.54%	81.50%	86.23%	93.57%	83.77%	88.40%	7367.32	0.12

Table 2: Out-expansion macro averages and execution times for training and average per sentence for SciAD dataset with also Wikipedia as training data.

the SDU@AAAI competition. We also report the best results obtained for both the Dev set used for hyperparmeter selection and the Test set as testing data. In addition, we report the execution times for training and the average per predicted acronym in a sentence (note that each sentence contains only one acronym to expand).

We can see that Cossim with Classic Context Vector achieved the best results. In general, both techniques have slightly lower recall (less than 1%) in the Test set than in the Dev test but higher macro precisions (1-2%). Since the gains in macro precisions are higher than the losses in macro recalls for the Test set, the harmonic means of both macros (i.e., macro F1-measures) are higher in the Test set. Differences among the techniques are consistent across various test sets. Regarding execution times, Cossim with Classic Context Vector is faster in training (91s) and SVM with Doc2Vec is faster on average per sentence (0.09s). Classic Context Vector counts word occurrences at training time while Doc2Vec trains a neural network for word and document embeddings with several iterations over the training corpus (e.g., 200). Both training and sentence processing times are low given that they are executed on a regular machine (4 CPU cores and 32GB of RAM). Both techniques are lightweight solutions for this problem.

Cross-Training and Additional Data for the SDU@AAAI Competition For our next set of experiments for the SDU@AAAI competition, we increase the training data provided by the competition sets with documents obtained from Wikipedia, i.e., the Wikipedia dataset. We wanted to test whether additional data and cross-training data helps to solve this problem and which techniques can benefit from such a data increment.

Table 2 shows the macro averages on the SciAD Dev set using SciAD train and Wikipedia documents as training data; and the macro averages and execution times on the SciAD test set using the above training sets plus the Dev set as training data.

In contrast to previous results where Wikipedia data was not used, after adding Wikipedia documents to the training data, SVM with Doc2Vec obtains the best results. That combination also benefits from using Wikipedia data. The three macro averages are lower when applied to the Dev set than when applied to the Test set.

Consistent with the experimental results without Wikipedia, Cossim with Classic Context Vector is faster in training than SVM with Doc2Vec, while slower in per-sentence processing. Training both techniques is much slower with the addition of Wikipedia data, yet fast enough for a regular machine (e.g., 2 hours to train the Doc2Vec model). On average, to process a sentence, the incorporation of Wikipedia slows down Cossim with Classic Context Vector by 0.43 seconds and slows down SVM with Doc2Vec by 0.03 seconds.

Most of the excellent efforts by other research groups submitted to the competition are transformer-based models that use pretrained models like BERT (Devlin et al. 2018), ROBERTA (Liu et al. 2019), and SciBERT (Beltagy, Lo, and Cohan 2019). Those works mostly distinguish themselves on how they adapt such transformers models to out-expansion (Veyseh et al. 2020). The three leaderboard works use transformers and their macro F1-measures range from 93.19% to 94.05%. In our understanding, only three works including ours explored alternative techniques to transformers, no other work explored Doc2Vec or SVMs. Although our best technique scores are 6% less than the best in competition, we believe that our techniques are distinct enough to be complements to transformer-based techniques or may introduce a lighter/faster approach to this problem since transformer models even using GPUs (Graphics processing units) or TPUs (Tensor Processing Unit (TPU)) usually take more time to train and to process data than Doc2Vec and SVMs. Further, our approaches could work better when the context consists of entire documents rather than single sentences, which is our core use case.

Conclusions and Future Work

We have evaluated two rapid techniques for acronym disambiguation using the SDU@AAAI benchmarks. We have found that Cosine similarity with Classic Context Vector works best when no Wikipedia data is used. SVM with Doc2Vec outperforms Cosine similarity with Classic Context Vector when using Wikipedia data. Our overall results, as measured by F1-measure score, are within 5.7% of the best system in competition. By analyzing the execution times of each phase (training and evaluation of sentences), we showed that our approach is lightweight even on a standard computer.

We believe we could have improved performance if we had used data sources in addition to Wikipedia such as abstracts from articles in web repositories to make the domain closer to the SDU@AAAI competition data.

Acknowledgments

Pereira's work was supported by national funds through FCT (*Fundação para a Ciência e a Tecnologia*), under the PhD Scholarship SFRH/BD/135719/2018. Furthermore, Pereira and Galhardas' work was supported by national funds through FCT under the project UIDB/50021/2020.

Shasha's work has been partly supported by (i) the New York University Abu Dhabi Center for Interacting Urban Networks (CITIES), funded by Tamkeen under the NYUAD Research Institute Award CG001 and by the Swiss Re Institute under the Quantum Cities initiative, (ii) NYU Wireless, and (iii) U.S. National Science Foundation grants 1934388, 1840761, and 1339362.

The server virtual machine used to run the experiments was supported by BioData.pt – *Infraestrutura Portuguesa de Dados Biológicos*, project 22231/01/SAICT/2016, funded by Portugal 2020.

References

Abdalgader, K., and Skabar, A. 2012. Unsupervised Similarity-based Word Sense Disambiguation Using Context Vectors and Sentential Word Importance. *ACM Transactions on Speech and Language Processing* 9(1):2–21.

Beltagy, I.; Lo, K.; and Cohan, A. 2019. SciBERT: Pretrained Language Model for Scientific Text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*.

Bird, S.; Klein, E.; and Loper, E. 2009. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition.

Charbonnier, J., and Wartena, C. 2018. Using word embeddings for unsupervised acronym disambiguation. In *Proceedings of the Twenty-Seventh International Conference on Computational Linguistics*, 2610–2619. Santa Fe, New Mexico: Association for Computational Linguistics.

Ciosici, M. R., and Assent, I. 2018. Abbreviation expander - a web-based system for easy reading of technical documents. In *Proceedings of the Twenty-Seventh International Conference on Computational Linguistics: System Demonstrations*, 1–4. Santa Fe, New Mexico: Association for Computational Linguistics.

Ciosici, M. R.; Sommer, T.; and Assent, I. 2019. Unsupervised abbreviation disambiguation contextual disambiguation using word embeddings. arXiv preprint. arXiv:1904.00929v2 [cs.CL]. Ithaca, NY: Cornell University Library.

Dai, A. M.; Olah, C.; and Le, Q. V. 2015. Document embedding with paragraph vectors. arXiv preprint. arXiv:1507.07998 [cs.CL]. Ithaca, NY: Cornell University Library.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint. arXiv:1810.04805 [cs.CL]. Ithaca, NY: Cornell University Library.

Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; and Lin, C.-J. 2008. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research* 9:1871–1874.

Feng, S.; Xiong, Y.; Yao, C.; Zheng, L.; and Liu, W. 2009. Acronym extraction and disambiguation in large-scale organizational web pages. In *Proceedings of the Eighteenth ACM Conference on Information and Knowledge Management*, 1693–1696. New York, NY: Association for Computing Machinery.

Gooch, P. 2012. BADREX: in situ expansion and coreference of biomedical abbreviations using dynamic regular expressions. arXiv preprint. arXiv:1206.4522 [cs.CL]. Ithaca, NY: Cornell University Library.

Kipf, T. N., and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. arXiv preprint. arXiv:1609.02907 [cs.CL]. Ithaca, NY: Cornell University Library.

Le, Q. V., and Mikolov, T. 2014. Distributed representations of sentences and documents. In *Proceedings of the Thirty-First International Conference on Machine Learning*, 1188–1196.

Li, Y.; Zhao, B.; Fuxman, A.; and Tao, F. 2018. Guess Me if You Can: Acronym Disambiguation for Enterprises. In *Proceedings of the Fifty-Sixth Annual Meeting of the Association for Computational Linguistics*, 1308–1317. Melbourne, Australia: Association for Computational Linguistics.

Li, C.; Ji, L.; and Yan, J. 2015. Acronym Disambiguation Using Word Embedding. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 4178–4179. Menlo Park, Calif.: AAAI Press.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv preprint. arXiv:1907.11692 [cs.CL]. Ithaca, NY: Cornell University Library.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013a. Efficient Estimation of Word Representations in Vector Space. arXiv preprint. arXiv:1301.3781v3 [cs.CL]. Ithaca, NY: Cornell University Library.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; and Dean, J. 2013b. Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of the Twenty-Sixth International Conference on Neural Information Processing Systems*, 3111–3119. Red Hook, NY: Curran Associates Inc.

Moon, S.; McInnes, B.; and Melton, G. B. 2015. Challenges and practical approaches with word sense disambiguation of acronyms and abbreviations in the clinical domain. *Healthcare Informatics Research* 21(1):35–42.

Moon, S.; Pakhomov, S.; and Melton, G. B. 2012. Automated disambiguation of acronyms and abbreviations in clinical texts: window and training size considerations. *AMIA Annual Symposium proceedings* 2012:1310–1319.

Moro, A., and Navigli, R. 2015. SemEval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. In *Proceedings of the Ninth International Workshop on Semantic Evaluation*, 288–297. Denver, Colorado: Association for Computational Linguistics.

Navigli, R. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys* 41(2):1–69.

Pakhomov, S.; Pedersen, T.; and Chute, C. G. 2005. Abbreviation and acronym disambiguation in clinical discourse. *AMIA Annual Symposium proceedings* 2005:589–593.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Pouran Ben Veyseh, A.; Dernoncourt, F.; Tran, Q. H.; and Nguyen, T. H. 2020. What does this acronym mean? introducing a new dataset for acronym identification and disambiguation. In *Proceedings of the Twenty-Eighth International Conference on Computational Linguistics*, 3285–3301. Barcelona, Spain (Online): International Committee on Computational Linguistics.

Prokofyev, R.; Demartini, G.; Boyarsky, A.; Ruchayskiy, O.; and Cudré-Mauroux, P. 2013. Ontology-based word sense disambiguation for scientific literature. In *Proceedings of the Thirty-Fifth European Conference on Advances in Information Retrieval*, 594–605. Berlin, Heidelberg: Springer-Verlag.

Pustejovsky, J.; Castaño, J.; Cochran, B.; Kotecki, M.; and Morrell, M. 2001. Automatic extraction of acronym-meaning pairs from MEDLINE databases. *Studies in Health Technology and Informatics* 84(Pt 1):371–375.

Schwartz, A. S., and Hearst, M. A. 2003. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Pacific Symposium on Biocomputing*, 451–462. Singapore: World Scientific Press.

Stevenson, M.; Guo, Y.; Al Amri, A.; and Gaizauskas, R. 2009. Disambiguation of Biomedical Abbreviations. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, 71–79. Boulder, Colorado: Association for Computational Linguistics.

Thakker, A.; Barot, S.; and Bagul, S. 2017. Acronym Disambiguation: A Domain Independent Approach. arXiv preprint. arXiv:1711.09271v3 [cs.CL]. Ithaca, NY: Cornell University Library.

Veyseh, A. P. B.; Dernoncourt, F.; Nguyen, T. H.; Chang, W.; and Celi, L. A. 2020. Acronym identification and disambiguation shared tasks for scientific document understanding. In *Proceedings of the AAAI-21 Workshop on Scientific Document Understanding*.

Wu, Y.; Xu, J.; Zhang, Y.; and Xu, H. 2015. Clinical abbreviation disambiguation using neural word embeddings. In *Proceedings of the Workshop on Biomedical Natural Language Processing*, 171–176. Beijing, China: Association for Computational Linguistics.

Wu, Y.; Denny, J. C.; Trent Rosenbloom, S.; Miller, R. A.; Giuse, D. A.; Wang, L.; Blanquicett, C.; Soysal, E.; Xu, J.; and Xu, H. 2017. A long journey to short abbreviations: developing an open-source framework for clinical abbreviation recognition and disambiguation (CARD). *Journal of the American Medical Informatics Association* 24(e1):e79–e86.

Yu, H.; Kim, W.; Hatzivassiloglou, V.; and Wilbur, J. 2006. A Large Scale, Corpus-Based Approach for Automatically Disambiguating Biomedical Abbreviations. *ACM Transactions on Information Systems* 24(3):380–404.