A Study of Methods for the Generation of Domain-Aware Word Embeddings

Dominic Seyler
dseyler2@illinois.edu
University of Illinois at Urbana-Champaign
Department of Computer Science
Urbana, IL, USA

ChengXiang Zhai
czhai@illinois.edu
University of Illinois at Urbana-Champaign
Department of Computer Science
Urbana, IL, USA

ABSTRACT

Word embeddings are essential components for many text data applications. In most work, "out-of-the-box" embeddings trained on general text corpora are used, but they can be less effective when applied to domain-specific settings. Thus, how to create "domain-aware" word embeddings is an interesting open research question. In this paper, we study three methods for creating domain-aware word embeddings based on both general and domain-specific text corpora, including concatenation of embedding vectors, weighted fusion of text data, and interpolation of aligned embedding vectors. Even though the investigated strategies are tailored for domain-specific tasks, they are general enough to be applied to any domain and are not specific to a single task. Experimental results show that all three methods can work well, however, the interpolation method consistently works best.

KEYWORDS

domain adaptation, text representation, empirical study

ACM Reference Format:
Dominic Seyler and ChengXiang Zhai. 2020. A Study of Methods for the Generation of Domain-Aware Word Embeddings. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3397271.3401287

1 INTRODUCTION

Text representation is at the foundation of a wide range of text data applications (e.g., retrieval, clustering, categorization, summarization, and predictive modeling). When text representations are optimized for the task at hand, it naturally follows that they improve the downstream performance. "Out-of-the-box" word embeddings are disadvantaged by the fact that they are trained on general text corpora and are thus non-optimal when applied in domain-specific settings. Therefore, it is crucial to create word embeddings in a "domain-aware" manner, which is challenging due to the resource poorness of many specialized domains.

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401287

Since specialized domains usually do not have the amount of data required to train high-quality word embeddings, it is crucial to leverage the publicly-available, large-scale general data sources. This is due to the fact that even though a domain dataset might have good coverage of domain-specific terms, it will lack the breath of coverage that the general-purpose embeddings have. The general-purpose embeddings will have more robust vector representations of common words, since their word count is much higher and therefore words have been trained in more contexts. However, they are unable to capture the domain-specific semantic associations. How to best combine domain and general text data in this setting is a question we investigate in this work. We present and study three representative strategies for creating domain-aware word embeddings based on both general and domain-specific text corpora, including weighted fusion of text data, concatenation of embedding vectors, and interpolation of aligned embedding vectors.

The weighted fusion method combines the general and domain text data before training of the word embeddings to emphasise words and contexts that are only common in the domain dataset through repetition (the amount of repetition is controlled via a hyperparameter). The concatenation method combines the vectors of general and domain embeddings post-training, allowing the model to learn which dimensions are better suited for the task at hand. The embedding interpolation method is a novel technique inspired by Jelinek-Mercer smoothing [7] that interpolates trained general-purpose and domain embeddings using a weighting (i.e., smoothing) parameter. However, a direct application of the smoothing idea does not work because the two embedding vector spaces are not "compatible" with each other, i.e., the dimensions are not aligned. We address this challenge by learning a transformation matrix to transform the vectors in one space to those in the other so as to make them compatible and then interpolate them, allowing the combined vector reflecting the semantic associations in both datasets. All the three methods are general and can be applied to any task in any domain.

We systematically evaluate the proposed three methods using a common text application task, i.e., text classification, on a cybersecurity domain dataset. Experimental results show that all three methods are effective and domain-aware embeddings consistently outperforms generic embeddings, confirming that the common practice of direct use of generic embeddings is non-optimal. We find the interpolation method to be the most robust and effective among all three methods.

2 METHODOLOGY

As part of our goal to create domain-aware word embeddings, we want to leverage general (e.g., Wikipedia) and domain-specific unstructured textual data and study how to best combine them. This can be done at the corpus level (i.e., pre-training), model level (i.e., intra-training) and vector level (i.e., post-training). In the following we study one corpus level and two vector level methods.

2.1 Interpolation of Aligned Embeddings

Let E^{domain} and $E^{general}$ be domain and general word vector spaces. We define the smoothed embedding space E^{smooth} as in Equation 1. Here, we introduce a smoothing parameter λ, that weighs the influence of the general and domain word embeddings. We make λ a parameter of our model, which we evaluate in our experiments.

$$E^{smooth} = (1-\lambda)E^{domain} + \lambda E^{general} \tag{1}$$

Since E^{domain} and $E^{general}$ are trained on different corpora, the underlying vector spaces need to be aligned before they can be added. Therefore, we learn a transformation matrix W that estimates the transformational property described in Equation 2.

$$E^{general}W = E^{domain} \tag{2}$$

We learn W by solving the optimization problem in Equation 3 using stochastic gradient decent, by aligning identical words from the different embedding spaces, indexed by i.

$$\min_W \sum_i ||WE_i^{general} - E_i^{domain}||^2 \tag{3}$$

Incorporating W in Equation 1, we arrive at our final embedding function in Equation 4. Using W we can transform the general vectors into the domain-specific space[1]. By adding them together, the model can enrich the domain embeddings with signals from the general embeddings that might not be captured in the domain training data. We argue that due to the small size of the domain corpus, the resulting word embeddings are "over-fit" and can benefit from generalization, introduced by our smoothing method.

$$E^{smooth} = (1-\lambda)E^{domain} + \lambda WE^{general} \tag{4}$$

2.2 Concatenation of Embeddings

General and domain embeddings can also be combined through concatenation. Here, the intuition is that the model will learn to prioritize certain embedding dimensions using the training data. For the domain and general word embeddings E^{domain} and $E^{general}$, the resulting word embedding E^{concat} is described in Equation 5, where $||$ is the concatenation operation. Concatenation is done only for words that appear in both embedding spaces.

$$E^{concat} = E^{domain}||E^{general} \tag{5}$$

2.3 Weighted Fusion of Training Data

When general and domain textual data are available, it is possible to combine both before training of the word embeddings. To increase the prominence of the (small) domain in the (much-larger) general dataset, we choose to duplicate the domain data during training. For this we introduce a parameter N, which reflects the number of repeats of the domain data during training. N has the following properties: if $N = 0$, the resulting word embeddings are trained using no domain data, if $N \to \infty$, then the resulting word embeddings are trained on domain data.

[1]We found that transforming the domain-specific vectors into the general embedding space was less effective.

3 EXPERIMENTAL SETUP

Dataset: *MalwareTextDB* [10] contains information about malware and vulnerabilities from articles that are published on the web. We focus on the task of classifying relevant sentences for inferring malware actions and capabilities (binary sentence classification). We employ the second version of the dataset, that was released as part of the 2018 International Workshop on Semantic Evaluation [14]. However, in our experiments we found large discrepancies between the performances of the development and test datasets, which lead us to conclude that these were constructed in a biased way. Therefore, we randomly re-sample the entire dataset into training (80%, 10,334 sentences), development (10%, 1,292 sentences) and testing (10%, 1,292 sentences). In our experiments we report F_1 score in accordance with the evaluation metrics of the workshop.

Word embeddings: We use the word2vec algorithm [11] to train our all word embeddings. Our domain embeddings are trained on a corpus of cybersecurity related documents, which we crawl from the web. Our crawler is restricted to 500 domains of cybersecurity companies [1] and keeps documents that are have a high similarity with the documents in the training corpus. After removing duplicates, the final collection contains 220,088 unique documents, with over 160 million words. For our general embeddings, we make use of a Wikipedia dump with over 2.5 billion words.

Classifier: Since our goal is to compare different word embedding models (optimizing task performance is not our main goal), we decided to make use of a simple neural classification method. We implement the standard Convolutional Neural Network model from Kim [8]. All models are trained on the training set, tuned on the development set and evaluated on the test set. We report the results of the model that performed best on the development set.

Hyperparameters: Besides the parameters required by our classification frameworks, we investigate the effect of hyperparameters introduced by our models. *Lambda (λ)*: This parameter regulates the interpolation between domain and general embeddings. In our experiments we explore different settings for λ within the interval [0,1], in 0.1 increments. *Domain duplication factor (N)*: Relates to our weighted fusion of training data strategy. We explore different settings ranging from no domain data ($N = 0$) to a dataset dominated by domain data ($N = 100$). *Embedding dimension (dim)*: There is no general way of determining the optimal dimension of the vector space of word embeddings. Thus, we explore different settings for $dim \in \{100, 200, 300\}$.

Baselines: We compare our models to two embedding baselines to examine our main hypothesis that some combination of both is better than each of them individually. *GENERAL* is the widely-used case of standard embeddings that are trained on a large general-purpose corpus. In our case, these are embeddings trained on a large Wikipedia corpus. *DOMAIN* are embeddings trained on a small domain corpus. We train cybersecurity embeddings from our webcrawl, mentioned above.

dim	Dev	Test	Model
300	0.5693	0.5209	DOMAIN
	0.5411	0.4773	GENERAL
	0.5952	0.5331	CONCAT
	0.6087	0.5429	FUSION (N=15)
	0.6498	**0.5778**	INTERPOL ($\lambda = 0.5$)
200	0.6340	0.6013	DOMAIN
	0.5411	0.5168	GENERAL
	0.5912	0.5164	CONCAT
	0.6131	**0.6069**	FUSION (N=50)
	0.6655	0.6053	INTERPOL ($\lambda = 0.2$)
100	0.5934	0.5685	DOMAIN
	0.5271	0.5090	GENERAL
	0.5688	0.5220	CONCAT
	0.6156	0.5607	FUSION (N=100)
	0.6446	**0.6239**	INTERPOL ($\lambda = 0.4$)

Table 1: F_1 score for different embedding dimensions.

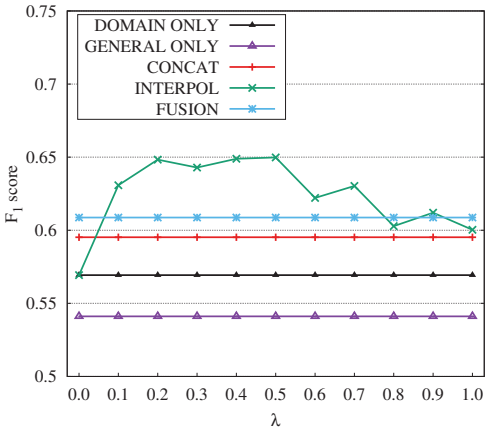

Figure 1: Influence of Lambda Parameter (dim=300).

4 EVALUATION

4.1 Comparison of Models

We compare the baselines DOMAIN and GENERAL to the three proposed methods CONCAT, INTERPOL and FUSION. We present the results in Table 1. Results are shown for different embedding dimensions along with the performance on development (Dev) and test (Test) sets and the model name. For the INTERPOL and FUSION we only show the best performing model on the development set, with its lambda parameter (λ) and domain duplication factor (N), respectively. The best performance for each dimension is marked in bold, and the best performing model on the dataset is underlined.

We find that the INTERPOL method consistently outperforms all baselines and other models on the development set. It is only slightly outperformed on the test set for dim=200 by FUSION. Based on the test performance it seems that 100 dimensions works best. Concatenation does not always work well, as it is often outperformed by the DOMAIN baseline. Considering the baselines, it seems that DOMAIN performs better than GENERAL in the majority of the cases, as can be observed for all dimensions. Overall, we conclude that combining domain and general data is generally beneficial. As concatenation is not always able to outperform the baselines, interpolation should be the preferred method due to it's flexibility introduced by the λ parameter, which we investigate next.

4.2 Lambda Parameter

Our previous evaluation has shown that performance can be sensitive to the λ parameter. Therefore, we further study the lambda parameter's sensitivity on the development set. In Figure 1 we plot F_1 performance for different settings of λ against the other models. It is noticeable that adding even small amounts of general information can push the performance past all other models (see $\lambda = 0.1$). Values higher than 0.5 seem to hurt performance, which can also be observed in Table 1 for other dimensions. Here, we would like to point out that even though $\lambda = 0.0$ is equal to the DOMAIN model, $\lambda = 1.0$ is not equal to the GENERAL model. This is because general embeddings are transformed using the alignment matrix W, as described in Equation 4. Interestingly, we find that just transforming the general embeddings with the alignment matrix helps boosting the performance (see $\lambda = 1.0$), which can be interpreted as evidence that the learned transformation is effective.

4.3 Domain Duplication Factor

To further understand how to best combine general and domain data, we experiment with a third method: weighted fusion of training data. The domain duplication factor (N) weights the amount of domain data that is used during training. We show the results for different settings of N in Table 2. The row "% domain" shows the percentage of domain data in comparison to the general data. The best model for each dimension is in boldface. We find that the N parameter is very sensitive and no clear overall trend is observable. Based on the results from Tables 1 and 2, we conclude that the fusion of training data can outperform the other baselines under certain conditions and beat the INTERPOL model in one setting: dim=200, on the test dataset. Furthermore, tuning N is costly, as it requires re-training of the embeddings for each setting.

4.4 Qualitative Evaluation

Following our quantitative evaluation, we further perform a qualitative analysis where we inspect the properties of the resulting embedding spaces. Table 3 shows the top ten closest neighbors to the word "bug" in different embedding spaces (DOMAIN, GENERAL, INTERPOL $\lambda = 0.5$) within the cybersecurity domain. We purposefully choose the word "bug" because of its ambiguity in different domains, which is nicely reflected in the table. GENERAL captures mostly the biological meaning of bug and lists other insects (e.g., worm), properties (e.g., flappy) or syntactic variations (e.g., bugs). Surprisingly, the word "stagefright", which is the name of a well-known bug in the Android operating system, is ranked in one of the top positions. In contrast, DOMAIN lists more cybersecurity related terms, where bugs are synonyms for errors in programs. The INTERPOL method is able to introduce "stagefright" to the most similar words and more general terms are ranked higher (vulnerability, loophole). This seems to indicate that INTERPOL is able to promote more general concepts and prevent the domain embeddings from being "overly specific" (i.e., overfit) to the domain.

N	0	1	3	5	10	15	20	25	50	100
% domain	0	6.27	18.82	31.37	62.73	94.10	125.46	156.83	313.66	627.31
dim=300	0.5411	0.5685	0.5455	0.5843	0.5777	**0.6087**	0.5859	0.5483	0.5909	0.5886
dim=200	0.5411	0.5554	0.5685	0.5657	0.5411	0.5954	0.5664	0.5473	**0.6131**	0.5973
dim=100	0.5271	0.5779	0.5972	0.5936	0.5419	0.5674	0.5546	0.5734	0.5990	**0.6156**

Table 2: Influence of domain duplication factor N.

DOMAIN		GENERAL		INTERPOL	
flaw	.795	bugs	.817	flaw	.758
glitch	.732	worm	.711	vulnerability	.690
vulnerability	.719	stagefright	.702	issue	.681
issue	.686	mouse	.679	bugs	.660
bugs	.644	flappy	.670	glitch	.629
defect	.624	beetle	.654	problem	.596
loophole	.611	blob	.651	defect	.550
problem	.610	gizmo	.641	flaws	.535
weakness	.571	stink	.637	stagefright	.524
flaws	.568	critter	.633	loophole	.519

Table 3: Neighbors of "bug" in Different Embedding Spaces.

5 RELATED WORK

Domain-agnostic are the general case of word embeddings with no domain information. The earliest work to encode the context of a word in a neural language model is Bengio et al. [2]. Mikolov et al. [11] introduces *word2vec*, where a target word's vector representation is used to predict its context window (Skip-gram) and vice versa (CBOW). Pennington et al. [12] proposes to incorporate word contexts using factorization of a word co-occurence matrix. Levy and Goldberg [9] alters the context selection while training, conditioning a word's context on the word dependencies within a sentence. The drawbacks of domain-agnostic methods are: (1) word representations are general and not tailored for specialized domains; (2) the evaluation does not consider specific tasks.

Domain-specific methods have some form of domain knowledge integrated during training [6, 15] or as a post-processing step. For instance, Faruqui et al. [4] proposes to "retrofit" word embeddings by incorporating semantic lexicons (e.g., WordNet). These methods require (semi-) structured data sources, which can be thought of as distant supervision during or post-training and require significant human effort. In contrast, our method is entirely unsupervised and requires no expert knowledge.

Combination of Embeddings methods have been studied in Ghannay et al. [5]. The authors compare vector concatenation, dimensionality reduction and an antoencoder model. The study focuses on different embedding algorithms, rather than domain-specificity.

Context-sensitive models, such as BERT [3] and ELMo [13], make the word vector dependent on the word sequence that the word occurs in. While our approaches and the context-sensitive models deviate from the generic embedding representation in that the generated representation better reflects the needed representation for a particular domain and task, the improvement is from different angles. As the methods we study are independent of the model that was used to train the word vectors, it is generally possible to incorporate context sensitive representations as word vectors into our methods.

6 CONCLUSION

We investigated three general methods for creating domain-aware word embeddings based on both general and domain-specific text corpora. Overall, interpolation performs best in almost all settings, whereas concatenation is not always able to outperform the baselines. The fusion model can achieve good performance when tuned correctly, but tuning is a very costly process. The general embeddings are the least effective in almost all settings, thus we conclude that it is beneficial to create domain-aware word embeddings and the interpolation method can be recommended as a robust way to generate domain-aware embeddings. As the approach is general, it can be potentially used in many downstream applications to improve task performance, such as compromised account detection [16], stock trend prediction [17], etc.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1801652.

REFERENCES

[1] [n.d.]. Cybersecurity 500. https://cybersecurityventures.com/cybersecurity-500/
[2] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *J MACH LEARN RES* 3, Feb (2003).
[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.
[4] Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2015. Retrofitting Word Vectors to Semantic Lexicons. In *HLT-NAACL*.
[5] Sahar Ghannay, Benoît Favre, Yannick Estève, and Nathalie Camelin. 2016. Word Embedding Evaluation and Combination. In *LREC*.
[6] Saurav Ghosh, Prithwish Chakraborty, Emily Cohn, John S Brownstein, and Naren Ramakrishnan. 2016. Designing domain specific word embeddings: Applications to disease surveillance. *arXiv preprint arXiv:1603.00106* (2016).
[7] Frederick Jelinek and Robert L. Mercer. 1980. Interpolated Estimation of Markov Source Parameters from Sparse Data Pattern Recognition in Practice. *ES Gelsema and LN Kanal* (1980).
[8] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*.
[9] Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *ACL*.
[10] Swee Kiat Lim, Aldrian Obaja Muis, Wei Lu, and Chen Hui Ong. 2017. Malwaretextdb: A database for annotated malware articles. In *ACL*.
[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*.
[12] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.
[13] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.
[14] Peter Phandi, Amila Silva, and Wei Lu. 2018. Semeval-2018 Task 8: Semantic Extraction from CybersecUrity REports using Natural Language Processing (SecureNLP). In *SemEval*.
[15] Arpita Roy, Youngja Park, and SHimei Pan. 2017. Learning domain-specific word embeddings from sparse cybersecurity texts. *arXiv preprint arXiv:1709.07470* (2017).
[16] Dominic Seyler, Lunan Li, and ChengXiang Zhai. 2018. Identifying compromised accounts on social media using statistical text analysis. *arXiv preprint arXiv:1804.07247* (2018).
[17] Yiren Wang, Dominic Seyler, Shubhra Kanti Karmaker Santu, and ChengXiang Zhai. 2017. A Study of Feature Construction for Text-based Forecasting of Time Series Variables. In *CIKM*.