Quantifying the Effect of In-Domain Distributed Word Representations: A Study of Privacy Policies

Vinayshekhar Bannihatti Kumar[1], Abhilasha Ravichander[1], Peter Story[2], and Norman Sadeh[2]

[1]Language Technologies Institute, Carnegie Mellon University
[2]Institute for Software Research, Carnegie Mellon University
{*vbkumar,aravicha,pstory,ns1i*}@*andrew.cmu.edu*

Abstract

Privacy policies are documents that describe what data is collected by a website or an app and how that data is handled. Privacy policies are often long and difficult to understand. Recently people have started to turn to Natural Language Processing (NLP) to automatically extract statements from the text of these policies. This article reports on a study to evaluate the benefits of using word embeddings in this endeavor. Specifically, we use 150,000 privacy policies to build word vectors in an unsupervised manner. This includes evaluating the benefits of privacy specific word embeddings. Evaluation is conducted on the OPP-115 corpus of privacy policy annotations. By building privacy-specific embeddings we hope to accelerate research at the intersection of privacy policies and language technologies.

1 Introduction

Privacy policies tend to be long and complex documents that are often difficult to understand. They are the primary mechanism to inform users about the collection and handling of their data. Over the past several years, there has been a growing interest in *automating* the understanding of privacy policies using Machine Learning and Natural Language Processing techniques (e.g., (Sadeh et al., December 2013; Wilson et al., 2016; Sathyendra et al., 2017b; Liu et al., 2016b)). These techniques rely on a small amount of supervised training data. Unfortunately, as is the case in many other domains, supervised training data requires expert annotation and is expensive to obtain, limiting the size of available corpora. On the other hand, because privacy policies are ubiquitous, unlabeled privacy policy data is plentiful and easy to obtain. In this work, we examine leveraging this unlabeled data to train word embeddings for the privacy domain. We demonstrate that these word embeddings yield meaningful improvements in performance on the popular OPP-115 (Wilson et al., 2016) benchmark for segment labeling in policies. Performance improvements are observed across a diverse set of data practices found in the OPP-115 corpus. For evaluation we use the same test set which was provided by Wilson et al. (2018).

The contributions of our paper can be summarized as follows.

1. We investigate the utility of in-domain word embeddings, and find that they indeed help over generic word embeddings to obtain better segment-labeling performance in the privacy domain. We observe meaningful improvements on the OPP-115 corpus. We measure an average macro F1 of 0.803 on the test set.
2. We empirically investigate the relationship between dimensionality of the word embeddings and segment labeling performance. We look at the performance across a diverse set of important data-practice categories when the dimensionality of the word embeddings changes.
3. We investigate the number of policies that are required to train expressive word embeddings. We have access to over 300,000 privacy policies which we scraped from the Google Play Store. We investigate the amount of policies which are needed to get good results on the OPP-115 dataset. Henceforth, we call this the policy corpus. It is to be noted that we did not need all 300,000 privacy policies as the performance saturates fairly quickly. Incidentally, we would have needed significantly more computational resources to train word embeddings on all 300,000 policies. This is not to say that word embeddings trained on all 300,000 policies might not have helped in the context of different, possibly more subtle data practices than those considered in the OPP-115 dataset.
4. We present a qualitative analysis of representations of words in vector space in the privacy domain. We want to understand how in-domain word embeddings differ from the generic word embeddings which are domain independent.

2 Related Work

Word embeddings in NLP

Word embeddings have a rich history in the Natural Language Processing community, and have proven to be useful in a wide variety of tasks. Work on neural models for formulating word representations was reported as early as 2003 by Bengio et al. (2003). They used a simple feed forward neural network to capture the language model, thereby building word embeddings. Mikolov et al. (2013) introduced Word2Vec, a fast and scalabale way of computing word vectors on large corpora by using sub-sampling and negative sampling techniques. Pennington, Socher, and Manning (2014) introduced GloVe embeddings which captured both the local context and the global context of sentences. FastText (Bojanowski et al., 2016) is a way to train a model to use the sub-word information instead of considering words as discrete tokens. It thus captures local context and is robust to perturbations. Sub-word information helps us capture better local context, and thus we decide to use FastText for training our word embeddings.

Word embeddings and privacy policies

Harkous et al. (2018a) have previously used FastText on privacy policies to get word embeddings. However, these embeddings are not publicly available, nor were ablation studies performed to determine their utility. In addition, they were used for question category identification, rather than segment labeling as in (Wilson et al., 2016). Wilson et al. (2016) have done a thorough job in creating the OPP-115 dataset and also creating good baselines for the same. In this work, we fill this gap by training word embeddings from 150,000 privacy policies and demonstrating their utility over generic word embeddings. We show that our neural models with the help of our privacy embeddings outperform previous benchmarks on the OPP-115 dataset. Liu et al. (2018) have tried using pre-trained word embeddings with deep models to classify the OPP-115 data. However, on classes with lower numbers of positive examples, their performance seems to drop. In order to address this issue and to be more consistent with common machine learning practices, we do two things. First, we train our in-domain word embeddings to check if we will get better results on these smaller population classes. Second, we split the data into 5 folds. We use 4 folds of data to train our machine learning model. We call this the train set. We used the other fold for fine tuning our model. We call this the dev set. We choose the hyperparameters of our model using the dev set. We evaluate our model on the test set which was mentioned earlier. This way of separating the dataset helps ensure that we are not fitting your model to get good performance on the test set and helps us to be more robust to unseen data. We report our average F1 on all the folds of the dev set along with standard deviation. We also report our average F1 on the test set for our best performing model. Sathyendra et al. (2017a) used Word2Vec and trained privacy specific embeddings for an extrinsic task of question-answering. However, the authors do not report evaluating the performance of word embeddings. In contrast, our goal in the work reported herein is to shed light on and quantify the benefits of word embeddings in the privacy policy domain.

3 Overall Approach

Figure 1 describes our approach to building the word embeddings and then using them to perform the segment classification task on the OPP-115 corpus. We give a brief description of each of the modules of our system.

FastText

Fast text was introduced by Bojanowski et al. (2016). We leverage this algorithm to learn word embeddings for the words found in privacy policies scraped from the Google Play Store. We then do transfer learning by using these embeddings in the context of website privacy policies. It should be noted that these two domains have some meaningful overlap, with many privacy policies written to jointly address data practices associated with both mobile apps and websites operated by the same entity.

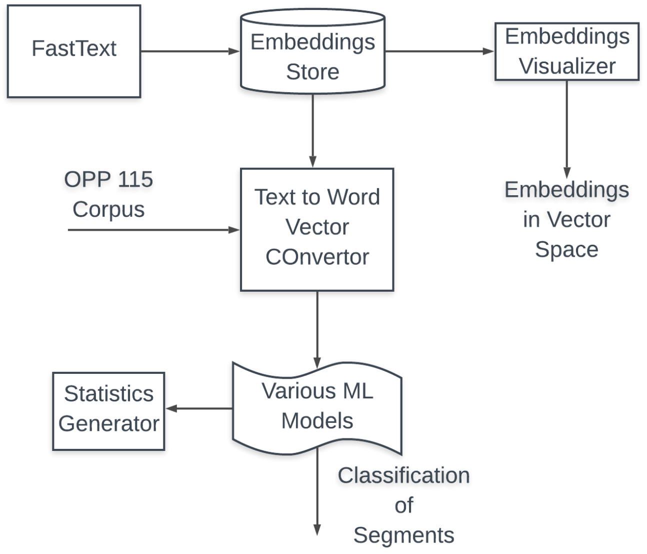

Figure 1: High Level Architecture of our approach

Embeddings Store

We store our learned embeddings for future access. We created separate word embedding stores for the different kinds of embeddings we built. We experimented with the dimensionality and the amount of data required to train good word embeddings. These results are discussed in the ”Results and discussions” section.

Text to Word Vector Converter

We have to convert the text representation of the OPP-115 corpus to real valued representation so that our neural models can start using them. But, if we have a large vocabulary it becomes difficult for networks to fit the data. So we restrict our vocabulary size to the top 50,000 words in the vocabulary. We also preprocess the data to convert all the words to lower case. We use all the pre-processing techniques used by Liu et al. (2018) to get privacy policy segments.

Neural Models

We try several neural architecture models. Following Faruqui et al. (2014) and Mikolov et al. (2013) we decided to use a simple feed-forward network by averaging these embeddings. We take the word vectors corresponding to every word in the segment and average them to form the input layer to our Multi Layer Perceptron. Not surprisingly, this model generally gives very good results on the OPP-115 dataset as it acts as a bag of words model. This also shows that our word embeddings are of high quality. We also report on performance with deep convolutional models in the experiment section.

Statistics Generator

OPP-115 (Wilson et al., 2016) is a small dataset and it is easy to get varying F1 scores on different runs of the neural network. The F1 score also varies when the fold of the data being tested on changes. In this paper, we want to set a standard when using this corpus. We take the view that one should report the mean F1 and standard deviation for each of the folds tested on. We used 5-fold validation. So this component of our system generated the statistics after training the network.

Embeddings Visualizer

It is a well documented fact that words which appear in similar contexts must be close to each other in the vector space (Mikolov et al., 2013). Our word embeddings are of 300 dimensions. In order to visualize these embeddings we have to convert them to 2D vectors and plot them. We use Principal Component Analysis (PCA) in order to do this. We examine the word vectors on a 2D plot. We report on and discuss differences between the plots of generic embeddings like GloVe (Pennington, Socher, and Manning, 2014) and our in-domain word embeddings.

4 Distribution of data practices in the OPP-115 dataset

The OPP-115 data-set is a multi-label classification problem which has skewed counts on some of the classes. For a detailed understanding of these privacy policies we ask the reader to refer to: Wilson et al. (2016). We provide a distribution of the classes in Figure 2.

Some of the classes in this dataset have fairly low counts of positive examples, making it difficult for machine learning models to learn the classification. For instance, in previous publications, the authors had found it hard to identify segments associated with the ”Data retention” class, as this class only has a few positive instances in the corpus (Wilson et al., 2016; Liu et al., 2018). In contrast, our results suggest that we are able to get significantly better F1 scores on these types of classes.

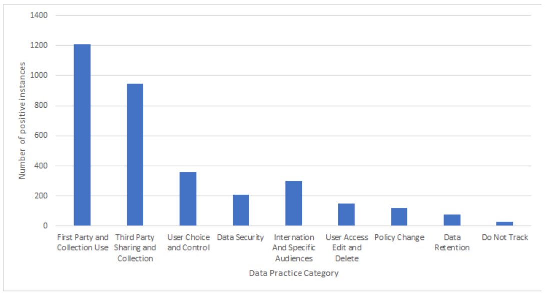

Figure 2: The counts of the number of positive examples in the OPP-115 data-set for each of the categories.

Category	Avg F1	Std Dev
First Party Collection/Use	0.783	0.02
Third Party Sharing/Collection	0.753	0.02
User Choice Control	0.641	0.06
Data Security	0.795	0.02
Intl and Specific Audiences	0.88	0.02
Access, Edit and Delete	0.716	0.03
Policy Change	0.88	0.06
Data Retention	0.48	0.01
Do Not Track	0.949	0.06

Table 1: Logistic Regression model's result on the dev set across different data-practices in the OPP-115 corpus.

5 Experiments

We performed several experiments before arriving at our final word embeddings. Each of our experiment was run on five folds of data. We report the mean and standard deviation of the F1 scores for the validation set. We also report the macro F1 score.

Logistic Regression

We used our word embeddings, averaged them and used that as the input to the logistic regression model. The results of the model are shown in Table 1.

We see that our baseline model has some improvement in performance over the results reported by Wilson et al. (2016). These results suggest that our word embeddings in the privacy domain yield important performance improvements. We get a macro F1 of 0.76 compared to macro F1 of 0.67 for the results reported in Wilson et al. (2016).

Feed-forward network

Feed-forward neural networks are powerful machine learning models which perform really well on classification tasks. We used a feed-forward neural network as shown in Figure 3. The input layer to this network was the averaged embeddings of all the words in the sentence.

More formally, given a list of words as a segment, we perform the following:

$$\text{wordVectors} = getWordVectors(w_1, w_2, w_3 ... w_n)$$

$$\text{averageEmbeddings} = \sum_{i \in n} vector_i \text{ / n}$$

Here $w_1, w_2 ... w_n$ are the words of a sentence. We convert these text of words into their vector representation to get $vector_i$. We then average these vectors to get *averageEmbeddings* .

These average embeddings were then used as the input layer to a two layer network. Each layer of the network had 64 Rectified Linear Unit (Relu) (Nair and Hinton, 2010) units. It is very easy to overfit on the training data as the size of the OPP-115 corpus is not too big. In order to circumvent this issue, we use dropouts (Srivastava et al., 2014), with a drop rate of 0.3 in the first layer and 0.2 in the second layer. In the final layer we used a softmax to predict if the segment belonged to a certain class or not. Henceforth, we call this model CBOW(Continuous Bag Of Words). The network architecture is shown in Figure 3:

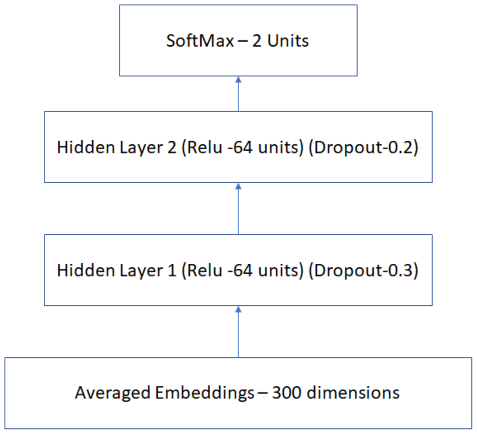
SoftMax – 2 Units
Hidden Layer 2 (Relu -64 units) (Dropout-0.2)
Hidden Layer 1 (Relu -64 units) (Dropout-0.3)
Averaged Embeddings – 300 dimensions

Figure 3: Neural models which gave the best performance

Category	Avg F1	Std Dev	Test-F1
First Party Collection/Use	0.814	0.01	0.801
Third Party Sharing/Collection	0.791	0.02	0.79
User Choice Control	0.692	0.07	0.712
Data Security	0.838	0.01	0.837
Intl and Specific Audiences	0.898	0.03	0.871
Access, Edit and Delete	0.757	0.04	0.823
Policy Change	0.917	0.08	0.875
Data Retention	0.55	0.05	0.58
Do Not Track	0.949	0.07	0.941

Table 2: Feed-forward Network's result on the dev and test set across different data-practices in the OPP-115 corpus.

This approach of text classification is generally considered to be a strong baseline by the NLP community (e.g., Faruqui et al. (2014) and Mikolov et al. (2013)).

Our CBOW Model gave us the best results. We report our accuracies in Table 2. It can be seen that our results are the state of the art on this dataset. Harkous et al. (2018b) show a higher F1 than ours. But these results are reported on a small set of user queries which by their nature are different from the segment classification task described in Wilson et al. (2016).

We compare our results with the F1 score of Wilson et al. (2016) and Liu et al. (2018).

We get a macro F1 of 0.803 when compared to their macro F1 of 0.667 on the classes which we have chosen to evaluate our model. We also compare our results with Liu et al. (2016a). We see that we get better results than Liu et al. (2018) on the average F1 score. The difference is significant when we use a neural model for classification on classes with lower number of positive examples. We made consistent observation with Liu et al. (2018) when using GloVe vectors. We found it hard to predict classes with lower counts of positive examples. However, our in-domain word embeddings help circumvent this issue by providing meaningful improvement in results. We get a macro F1 of 0.58 on this class. This further suggests that in-domain word embeddings are needed to get improvement in results.

Category	Avg F1	Std Dev
First Party Collection/Use	0.81	0.03
Third Party Sharing/Collection	0.767	0.04
User Choice Control	0.656	0.09
Data Security	0.75	0.06
Intl and Specific Audiences	0.85	0.05
Access, Edit and Delete	0.658	0.04
Policy Change	0.872	0.04
Data Retention	0.378	0.05
Do Not Track	0.90	0.1

Table 3: Deep convolutional model's result on the dev set across different data-practices in the OPP-115 corpus.

Deep Convolutional Model

We also tried deep convolutional models by convolving over the word embeddings. We used two sets of convolutional layers , with 200 filters each, but with strides of 3 and 5. We used Relu non-linearity and Maxpooling operation before being fed to the next set of these CONV-RELU-MaxPool Layers. The detailed architecture for our model is shown in Figure 4.

Although this model generally does better over the baselines described by Wilson et al. (2016), in some of the categories, it fails to capture the meaning when the number of positive examples are very small. The average macro F1 for this task was 0.74.

The results of our convolutional model is shown in Table 3. It can be seen that our model is very close to the CBOW model, but does not outperform it as the number of training examples is fairly low in the OPP-115 corpus. The simple feed forward network is able to capture the relationship better as the number of parameters in this model is orders of magnitude lower than in the deeper model. This is consistent with the general observation that as the model parameters increase, one needs more data to get better results.

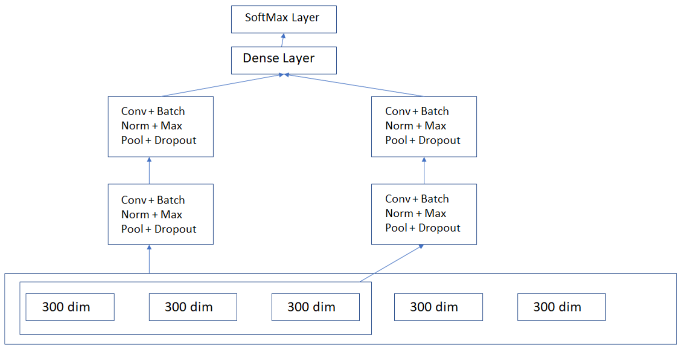
SoftMax Layer
Dense Layer
Conv + Batch Norm + Max Pool + Dropout
Conv + Batch Norm + Max Pool + Dropout
Conv + Batch Norm + Max Pool + Dropout
Conv + Batch Norm + Max Pool + Dropout
300 dim
300 dim
300 dim
300 dim
300 dim

Figure 4: Deep conv model for segment classification

6 Deep Contextualized Word Embeddings

We also explored the use of contextualized word embeddings, using BERT (Devlin et al., 2018), which at the time of writing is generally considered the state of the art model for such embeddings. We follow Devlin et al.'s recommendation and only train the model for 3 epochs.

Category	F1
First Party Collection/Use	0.86
Third Party Sharing/Collection	0.83
User Choice Control	0.74
Data Security	0.78
Intl and Specific Audiences	0.87
Access, Edit and Delete	0.8
Policy Change	0.83
Data Retention	0.36
Do Not Track	0.89

Table 4: BERT model's test set results across different data-practices in the OPP-115 corpus.

From Table 4 we see that we get good results on classes that have a higher number of positive instances. BERT does not do as well on classes with a lower number of positive instances. This is consistent with the observations across a lot of NLP tasks such as Question Answering or Natural Language Inference, where authors have reported significantly better results when BERT was used for classification tasks. It is to be noted that we don't do any hyper-parameter tuning with BERT, we just take an off-the-shelf model and train on the OPP-115 corpus for 3 epochs.

7 Results and Discussions

In this section we try and answer research questions that are at the intersection of privacy policies and language technologies.

Do in-domain word embeddings help?

In this section we compare our non contextualized word embeddings to the non contextualized generic word embeddings. We believe it is not fair to compare our word embeddings to BERT, because BERT is not an embedding, but more of a language model to represent sentences. Here we focus on evaluating the benefits of non-contextualized in-domain word embeddings over non-contextualized generic word embeddings. We use GloVe embeddings (Pennington, Socher, and Manning, 2014) as our generic embeddings. We observe from tables 2, 3, 4 and 5 that our in-domain embeddings perform better than GloVe in both models. This comparison intentionally ignores the performance of BERT, as BERT is a contextualized embedding that represent sentences rather than words.

It is also worth noting that the standard deviation of F1 scores is higher for practices with lower numbers of positive examples. In this paper, we have report average F1 scores across the different validation folds, along with their standard deviations. This is to capture the variability of the F1 score, as one might observe when looking at unseen data.

What should be the dimensionality of privacy embeddings?

We checked the performance of both the GloVe and our in-domain embeddings with different settings of the embeddings size (100 and 300). As can be seen, the 300 dimensional embeddings performed better than the 100 dimensional embeddings. The results of this experiment is provided in Table 7.

Category	Avg F1	Std Dev
First Party Collection/Use	0.772	0.02
Third Party Sharing/Collection	0.742	0.01
User Choice Control	0.62	0.07
Data Security	0.82	0.03
Intl and Specific Audiences	0.832	0.04
Access, Edit and Delete	0.721	0.04
Policy Change	0.897	0.07
Data Retention	0.32	0.08
Do Not Track	0.926	0.06

Table 5: F1 score across various categories of the OPP-115 data with CBOW and GloVe Embeddings

Category	Avg F1	Std Dev
First Party Collection/Use	0.795	0.02
Third Party Sharing/Collection	0.726	0.01
User Choice Control	0.606	0.07
Data Security	0.744	0.05
Intl and Specific Audiences	0.791	0.04
Access, Edit and Delete	0.662	0.04
Policy Change	0.828	0.04
Data Retention	0.286	0.08
Do Not Track	0.85	0.22

Table 6: F1 score across various categories of the OPP-115 data with Deep convolutional model and GloVe embeddings.

Category	GloVe-100	FT -100	GloVe-300	FT -300
First Party Collection/Use	0.725	0.801	0.772	0.814
Third Party Sharing/Collection	0.69	0.776	0.742	0.791
User Choice Control	0.528	0.68	0.620	0.692
Data Security	0.76	0.82	0.823	0.838
Intl and Specific Audiences	0.80	0.88	0.832	0.898
Access, Edit and Delete	0.61	0.76	0.721	0.757
Policy Change	0.883	0.915	0.897	0.917
Data Retention	0	0.51	0.32	0.55
Do Not Track	0.909	0.97	0.926	0.949

Table 7: F1 score across various categories of the OPP-115 using the CBOW Model and different dimensional vectors.

From Table 7, it can be observed that the higher dimensional embeddings tend to give better results over their lower dimensional counterparts. The higher order dimension takes longer to train, but since it has more dimensions it can capture relationships between words in a more expressive fashion.

Performance for different values of the embeddings are shown in from Table 7. Results presented in earlier tables are for 300 dimensional embeddings, as these embeddings performed than the 100 dimensional ones.

Training Data vs. F1 score

We now turn our attention to trying to answer the question of how many privacy policies are needed to get good embeddings for the OPP-115 dataset classification task. For this,

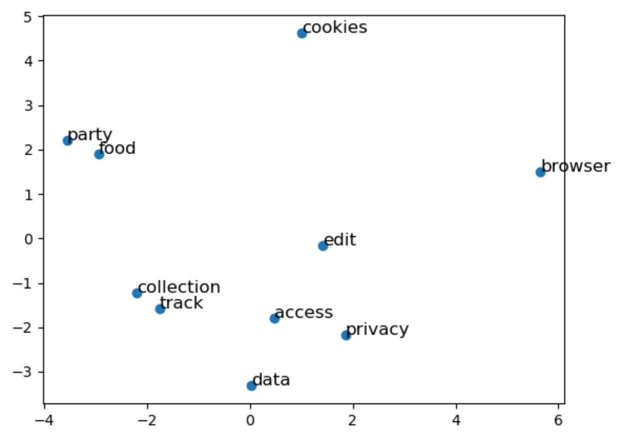

Figure 5: GloVe Word Vectors Visualization. Please note that the scale of the image is bigger than the one for FastText. Embeddings for words such as (first or third) ”party” and ”privacy”, which are closely related in the privacy domain, appear as fairly distant with GloVe.

we trained 100 dimensional FastText embeddings using different numbers of privacy policies and looked at the F1 score across various categories.

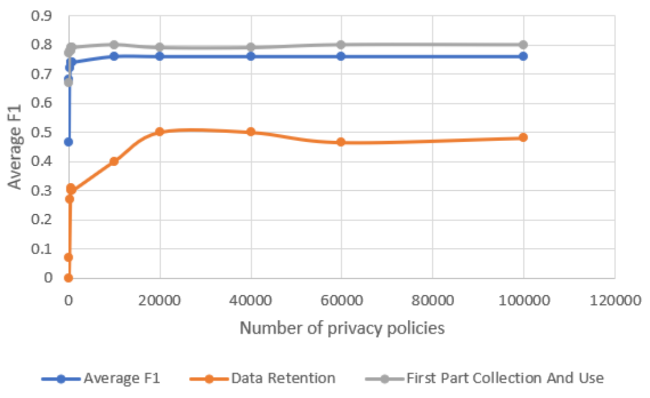

Figure 6: A scatter plot of number of privacy policies used while training word embeddings and the F1 score it achieved in the downstream task.

We observe from figure 7 that there is a plateau in the average F1 score which can be obtained after 20,000 policies. It is also interesting to note that classes which have higher positive examples in this dataset, tend to plateau quicker than the ones which have lower numbers of positive examples. For example, the “First Party Collection and Use” class has a lot more positive examples than the “Data Retention” class. We can see in figure 7 that the “Data Retention” class also takes a lot more time than the‘First Party Collection and Use” to reach the maximum F1 observed using our model.

How do the embeddings look?

After training word embeddings we would want to visualize a how the embeddings look like in the high dimensional vector space. For this, we take the 300 dimensional embeddings which we trained using our policy corpus and project them onto a 2D space using Principal Component Analysis (PCA). We compare the result of this projection with GloVe to see if our domain-specific model captures domain semantics better than GloVe.

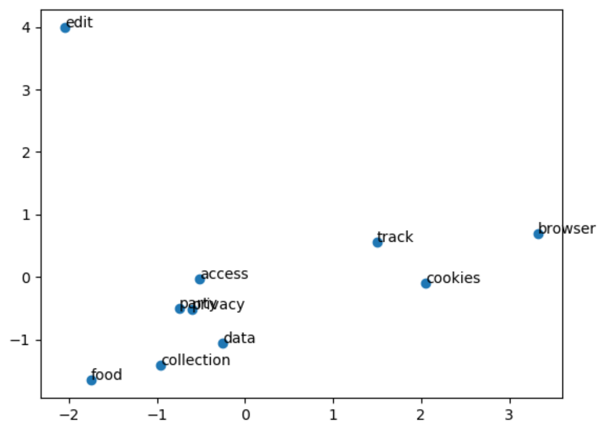

Figure 7: Fast Text Word Vectors visualization. The words ”privacy” and ”party” have almost overlapped and words such as ”track”, ”cookies” and ”browser” are also closer to one another.

It can be observed in Figure 7 that the privacy related terms are very closely clustered. In Figure 5 we see that there is no special semantics for privacy related words. The terms “food” and “party” are closely related, as that is the more common case outside of the domain of privacy policies. But in the Figure 7 “party” is more closely associated with “privacy” because of the privacy domain-specific concept of “third-party collection.” The terms “cookies,” “track,” and “browser” are three words that commonly appear together in privacy policies. We observe that these words are closely related in our in-domain vector space. It can also be observed that the terms ”data” and ”collection” are closer to each other in our in-domain vector space when compared to the more general vector space of GloVe. This is because privacy policies talk about users’ data collection most of the time.

While anecdotal, these observations suggest that the improvement in performance resulting from the use of in-domain embeddings can be attributed to these embeddings being able to better capture unique syntactic and semantic features of privacy policies.

8 Conclusion

In this paper we presented distributed word vector representations for privacy policies. We showed that in-domain embeddings can yield performance improvements on privacy policy related tasks over the use of generic embeddings such as GloVe. We reported good F1 scores across different data practice categories using our domain specific embeddings. We also showed both quantitatively and qualitatively that our in-domain word embeddings help improve performance of privacy policy segment labeling tasks on the OPP-115 corpus of privacy policies.

Acknowledgment

This study was supported in part by the NSF Frontier grant on Usable Privacy Policies (CNS-1330596). The US Government is authorized to reproduce and distribute reprints for Governmental purposes not withstanding any copyright notation. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the NSF or the US Government. This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562. The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing high performance computing resources that have contributed to the research results reported within this paper.

References

Bengio, Y.; Ducharme, R.; Vincent, P.; and Jauvin, C. 2003. A neural probabilistic language model. *Journal of machine learning research* 3(Feb):1137–1155.

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

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.*

Faruqui, M.; Dodge, J.; Jauhar, S. K.; Dyer, C.; Hovy, E.; and Smith, N. A. 2014. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166.*

Harkous, H.; Fawaz, K.; Lebret, R.; Schaub, F.; Shin, K. G.; and Aberer, K. 2018a. Polisis: Automated analysis and presentation of privacy policies using deep learning. *arXiv preprint arXiv:1802.02561.*

Harkous, H.; Fawaz, K.; Lebret, R.; Schaub, F.; Shin, K. G.; and Aberer, K. 2018b. Polisis: Automated analysis and presentation of privacy policies using deep learning. *arXiv preprint arXiv:1802.02561.*

Liu, B.; Andersen, M. S.; Schaub, F.; Almuhimedi, H.; Zhang, S.; Sadeh, N.; Acquisti, A.; and Agarwal, Y. 2016a. Follow my recommendations: A personalized privacy assistant for mobile app permissions. In *Symposium on Usable Privacy and Security.*

Liu, F.; Wilson, S.; Schaub, F.; and Sadeh, N. 2016b. Analyzing vocabulary intersections of expert annotations and topic models for data practices in privacy policies. In *2016 AAAI Fall Symposium Series.*

Liu, F.; Wilson, S.; Story, P.; Zimmeck, S.; and Sadeh, N. 2018. Towards automatic classification of privacy policy text.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 807–814.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Sadeh, N.; Acquisti, A.; Breaux, T. D.; Cranor, L. F.; Smith, N. A.; Liu, F.; Schaub, F.; and Wilson, S. December 2013. The usable privacy policy project: Combining crowdsourcing, machine learning and natural language processing to semi-automatically answer those privacy questions users care about. *Tech. report CMU-ISR-13-119, School of Computer Science, Carnegie Mellon University,Pittsburgh, PA 15213, USA.*

Sathyendra, K. M.; Ravichander, A.; Story, P. G.; Black, A. W.; and Sadeh, N. 2017a. Helping users understand privacy notices with automated query answering functionality: An exploratory study.

Sathyendra, K. M.; Wilson, S.; Schaub, F.; Zimmeck, S.; and Sadeh, N. 2017b. Identifying the provision of choices in privacy policy text. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2774–2779.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Wilson, S.; Schaub, F.; Dara, A. A.; Liu, F.; Cherivirala, S.; Leon, P. G.; Andersen, M. S.; Zimmeck, S.; Sathyendra, K. M.; Russell, N. C.; et al. 2016. The creation and analysis of a website privacy policy corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 1330–1340.

Wilson, S.; Schaub, F.; Liu, F.; Sathyendra, K. M.; Smullen, D.; Zimmeck, S.; Ramanath, R.; Story, P.; Liu, F.; Sadeh, N.; et al. 2018. Analyzing privacy policies at scale: From crowdsourcing to automated annotations. *ACM Transactions on the Web (TWEB)* 13(1):1.