Hw1: Presidential Speech Classification with Word Embeddings

Sam Showalter
University of California, Irvine (showalte)
Kaggle: Sam Showalter
showalte@uci.edu

Abstract

Text classification in model-restrictive settings is heavily reliant on predictive features to ensure adequate performance. In turn, this analysis explores several methods of text tokenization, feature engineering, and dimensionality reduction. Starting with simple dataset exploration, our experiments extend to attempts at encoding semantic meaning into the naive bag-of-words representations with Truncated Singular Value Decomposition and Word2Vec embeddings. Moreover, this experimentation was also applied to the model itself. Since the input features from the speeches are high-dimensional, different regularization methods were applied to improve learning and generalization. After each ablation, performance on a held-out validation set improved, and the best configuration from the experiment was passed to the next and built upon progressively. In addition, we further improved our classification performance by pre-training an embedding model on unlabeled data with Word2Vec, correlating words based on their meaning. Though alone these word embeddings did not exceed the supervised performance, when integrated the original set of supervised features performance improved slightly. However, the embeddings did exhibit a few failure modes where context hurt; these issues were be tracked back to the structure of the unlabeled data.

1 Introduction

Text classification can be accomplished with a variety of implementations. Among the simplest and most popular is the bag-of-words framework (Zhang et al., 2010). In this setting, a vocabulary is formed by tokenizing the input text and then creating a unique index for every distinct token. Then, one-hot vectors of each token are generated and a document's feature vector is represented as the sum or the logical-OR of its token vectors.

In general, feature vectors for NLP tend to be high-dimensional. To combat this curse of dimensionality in modeling, the vocabulary can be shrunk by several methods, including removing non-alphanumeric characters, expanding contractions, lemmatization, and removing stopwords. Even with these reductions, the resulting feature vectors tend to remain quite large. Further explicit reductions can come from recognizing that tokens that occur extremely rarely likely carry little predictive value. However, There is no guarantee this is true.

In subsequent sections, we outline our approach to feature engineering, supervised learning, and incorporating unlabeled data. Our main experimental contributions can be defined as follows:

1. We extensively explore feature engineering pipelines, including tokenization creation and resolution methods.
2. We explore dimensionality reduction techniques with and without the use of unlabeled data with Truncated SVD and Word2Vec.
3. We regularize our model, Logistic Regression, in several different ways and apply different gradient based solvers to the optimization.

2 Supervised Learning

In this section, we discuss methods of improving classification performance without making use of unlabeled data. The following sections are separated by feature- and model-based exploration.

2.1 Feature Engineering

To begin our analysis, a few small tests on our input data's vocabulary size were conducted. As expected, the tokens found with a simple regular expression (*regex*) separator exhibited an exponential decrease in frequency from most common to least. Fortunately, we were able to drop the differential between the most common token and the least by an order of magnitude simply be removing stopwords, as shown in Figure 1.

While this simple experiment gave us insight into the distribution of the token frequencies, we

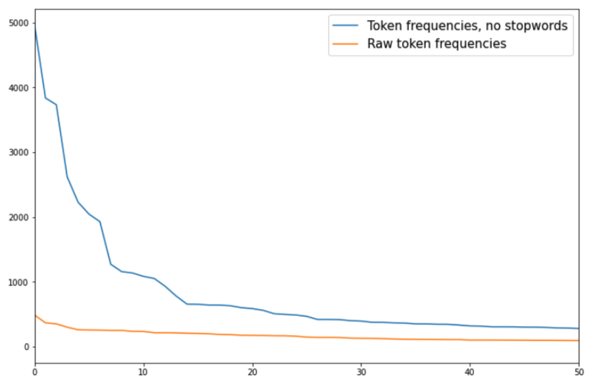

Figure 1: Token frequencies in training data before and after stopwords removed from the dataset

know little of vocabulary size. As mentioned in the introduction, several techniques exist to reduce vocabulary size and preserve meaning. A subset of these were applied to the input data, and the results are displayed in Figure 2. After regex separation, several additional filters are applied to the training data, where it can be seen that many tokens occur only once.

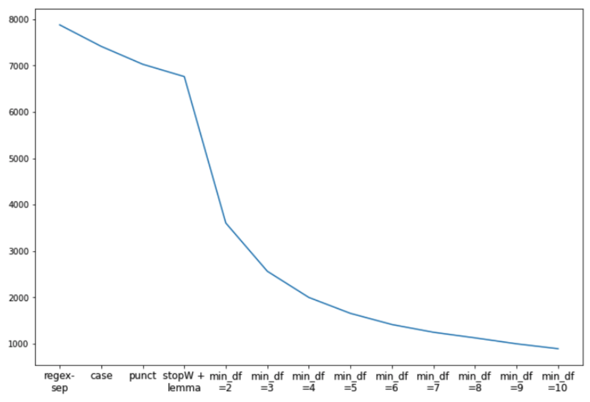

Figure 2: Vocabulary size as successive filters applied to input corpus. From left to right, the strings are regex separated, then case and punctuation is resolved, with final filters trimming rare tokens

To test different theories about the impact of feature engineering on classification, we ran a grid search of several different preprocessing methods on a standard implementation of Logistic Regression (lbfgs solver with L2 loss). Specifically, we leveraged two different tokenization strategies: a conservative regex tokenization from Scikit-Learn's `CountVectorizer` and a more sophisticated method from Python's `NLTK Toolkit`. In addition, we also resolved the case of characters and removed punctuation. Finally, lemmatization, or deconjugating words, was applied, as was the removal of stopwords. Once these were completed, vectors were generated from the tokens with two methods: the traditional bag-of-words (Luhn, 1957) count vectorization, and tf-idf vectorization. Tf-idf, or term-frequency inverse document frequency (Jones, 1972), extends on the bag of words notion by upweighting terms that do not appear in many documents and vice versa. That is, if a term occurs rarely across documents, it should be upweighted as it may prove more predictive.

Table 1: Feature Engineering Grid Search

		Token Filters			
Token Engine	Tok. Sep.	None	Case+ Punct.	Lem.	Stop Word
CVect	nltk	0.399	**0.435**	**0.435**	0.401
CVect	reg	0.396	0.413	0.413	0.399
Tf-Idf	nltk	0.382	0.382	0.382	0.386
Tf-Idf	reg	0.360	0.374	0.374	0.381

Displayed in Table 1, the removal of stopwords tended to, surprisingly, hinder performance. Tf-idf vectorization also performed worse in all cases, running counter to expectations. These two phenomena may be due to the fact that the filler words a candidate uses are somewhat unique to their speaking style and therefore predictive. Less surprisingly, nltk's more sophisticated tokenization engine outperformed its regex counterpart, perhaps due to how it handles intelligently separates contractions (breaking `can't` into `ca` + `n't`). This grid search led to a boost of 2.7% and 2.1% over the validation and testing baselines, respectively.

2.2 Dimensionality Reduction

While one method of dimensionality reduction in NLP focuses on trimming out words with little predictive value, the results from feature engineering encouraged us to intregrate all tokens if possible. Therefore, we leveraged Truncated SVD, a matrix decomposition method amenable to sparse data, to generate more semantic features without needing explicitly remove tokens. Below, we ran Truncated SVD across the features generated previously for an increasing number of components. With only 2500 components, roughly a third of the original vocabulary, we were able to exceed our previous performance on the held-out validation set (denoted with a red dotted line). Normalizing token counts before SVD had a negative effect on performance.

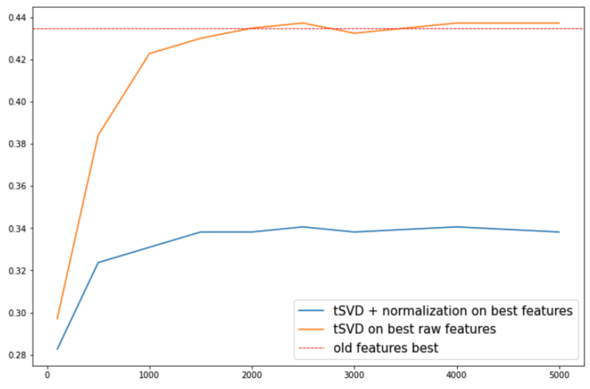

Figure 3: Classification performance applied to best feature selection after Truncated SVD applied with varying numbers of components

2.3 Model Tuning

With a smaller set of features and boosted performance on the baseline model, we turned our attention to model tuning directly. Concerned with the effect of dimensionality on the Logistic Regression, we explored regularization with L1 and L2 loss. For comparison, we also ran a trial where no regularization was applied.

Table 2: Model Tuning: Logistic Regression

	Logistic Reg. Solver				
Reg. Pen.	lbfgs	lib-linear	newton	saga	sag
-	0.399	-	0.401	**0.454**	0.447
L1	-	0.423	-	0.428	-
L2	0.436	0.432	0.448	0.447	**0.450**

In general, L1 loss, which induces sparsity, did not perform as well as L2 (weight decay) regularization. Moreover, the non-regularized model performed well on the validation set but hurt performance on the testing set, it was not as robust as its regularized counterpart.

3 Semi-supervised Learning with Word Embeddings

With our supervised model and data tuned, this section explores methods of incorporating unlabeled data to boost performance. In particular, we explore Word2Vec embeddings to encode semantic meaning between tokens.

3.1 Word2Vec for Text Classification

Word2Vec (Mikolov et al., 2013) is a vectorization method that seeks to embed words that co-occur together, assuming word proximity implies shared meaning (Lilleberg et al., 2015). In our implementation we use a window of 5, meaning the two words to the left and right of a target word are considered during the embedding process. More interesting, this method implicitly downsamples more frequent words, effectively handling the influence of stopwords. As shown in Figure 4, a Principal Component Analysis (PCA) decomposition a selection of word embeddings, candidates tend to be co-located together, as do words grouped by topic, like conflict or health.

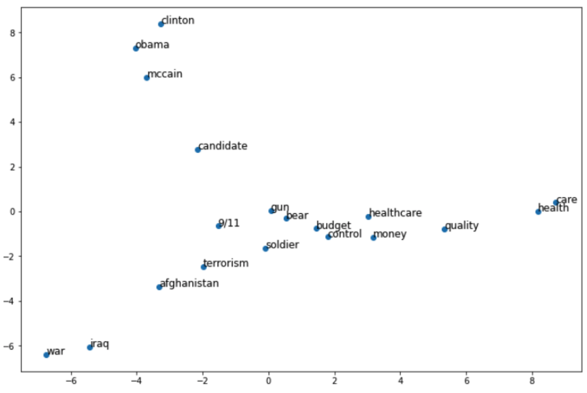

Figure 4: Visualization of Word2Vec embeddings with PCA with a selection of common political terms. Embeddings appear to capture semantic meaning

We chose to utilize word embedding models over the alternatives because we felt it would better generalize similarity in meaning between tokens and boost performance. Other implementations considered include dataset augmentation with unlabeled data. Though a promising option, we were concerned the approach would cause the trained model to diverge under its own, imperfect predictions, leading to poor performance.

3.2 Experimental Results and Discussion

Given a large corpus of 43,000+ unlabeled speech snippets, we then trained several Word2Vec modelswith different fractions of the dataset. The embedded dimensionality of these words was set to 300 (a common standard), reducing the original dimensionality by an order of magnitude. To generalize this context to documents, the word embeddings in the document are averaged, an order agnostic but somewhat performant approach (Turney and Pantel, 2010). Taken alone, these document embeddings were not able to come close to even the initial baseline performance, but increasing the document fraction of the Word2Vec model did improve performance in almost all cases, peaking at 0.34 when all unlabeled data is used.

We think one potential reason the Word2Vec embeddings alone did not meet previous performance is due to its averaging of word embeddings to make document vectors. Since documents are composed of token sequences of varying length, it is typically not useful to sum the embeddings. Averaging these embeddings overcomes this issue, but carries the cost of diluting document meaning across many tokens. Since many presidential candidates speak on similar topics, there may only be subtle differences in the composite embeddings. This is verified by looking at the embeddings of documents taken from the primary speeches of Obama and Clinton, the two most common classes in the dataset, displayed in Figure 5.

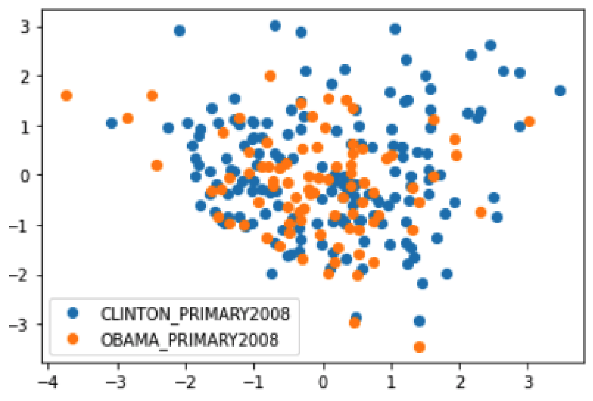

Figure 5: Visualization of political candidate embedding profiles as defined by their speech embeddings. Little separation can be cleaned, likely because order is not considered in embedding creation and candidates discuss similar topics

Examining this phenomena further, we find that among the 100 most common phrases tied to the tokens `obama` and `clinton`, half were shared. Much of what these candidates discuss is interrelated, as expected in a presidential race. With that said, there are still several defining features the embedding model discerned. Notably, the model tied the prases `tony`, `foreigner`, and `president` strongly to Obama but not Clinton. `tony` refers to Tony Rezko, an Obama fundraiser who was sentenced to prison, while the other features refer to comments about his citizenship and his ascent to president. By contrast, for Hillary Clinton, words such as `bubba` allude to her campaign to win the vote of white men in the southern United States.

There is considerable overlap in what the two candidates discussed, as described by their document embeddings mapped down to two dimensions. Thus, to differentiate candidates more effectively, these 300-dimensional embeddings were concatenated with the SVD features of the previous analysis. By incorporating semi-supervised embeddings with labeled data, we explored performance changes with our tuned model.

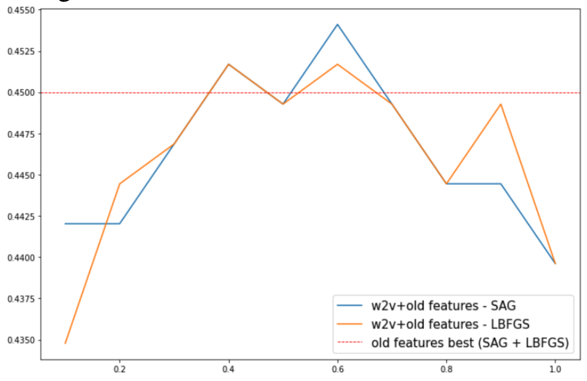

Figure 6: Performance with concatenated Word2Vec document embeddings and different fractions of unlabeled documents and SAG, LBFGS gradient solvers

Visualized in Figure 6, Word2Vec models that were trained on 60%of the unlabeled data proved optimal for downstream classification. It is surprising to see that performance decreased as the fraction of unlabeled data utilized exceeded 60%, and may be due to randomness or spurious interactions between our original and embedded data representations. Regardless, the performance on our validation set was again improved, though unfortunately this boost in performance did not translate to the validation set, though it exceeded the baseline.

We can also explore performance more closely with a few examples. In one document, the phrase `when Bill asks what time it is` was first predicted to be said by Obama, as the word `Bill` (Bill Clinton) was conflated with legislation. The Word2Vec model disambiguated this issue and correctly predicted Hillary Clinton, who was speaking of her husband. Unfortunately, this extra context was not always helpful. The phrase `the comatose stance, "Re-elect Obama"`, stated as criticism of Democrats by Newt Gingrich, was misinterpreted by the additional context that Democrats supported Obama in his bid for re-election in 2012. Instead of predicting Gingrich, the model now associated the phrase with Hillary Clinton, a vocal proponent of the Democratic party.

4 Conclusion

Through a sequence of grid search explorations, model tuning, and contextual word embeddings, we were able to improve our classifier beyond the existing baseline. We feel that this can be credited to our consideration of data tokenization, feature engineering, and principled methods for dimensionality reduction. However, the additional context we incorporated was not universally beneficial. In some cases, additional context actually further confused the canonical meaning, usually in cases where flowery or idiomatic language was utilized. However, with context the model improved overall.

5 Statement of Collaboration

For this project I utilized a variety of resources. These include articles read from google scholar, API documentation for different NLP packages, and office hours with Dr. Singh. Beyond these resources, I completed this project in isolation and the content within this report is my own.

References

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

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*.

Joseph Lilleberg, Yun Zhu, and Yanqing Zhang. 2015. Support vector machines and word2vec for text classification with semantic features. In *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*, pages 136–140. IEEE.

Hans Peter Luhn. 1957. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Yin Zhang, Rong Jin, and Zhi-Hua Zhou. 2010. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52.

A Future Research

Some potential avenues for future research within the stated constraints of the problem include alternative word embedding strategies such as GloVE (Pennington et al., 2014). In addition, designing a method to incorporate word order would likely improve the expressivity of our features. This could be as simple as incorporating bigrams or training a sequence (Hochreiter and Schmidhuber, 1997) or self-attention (Vaswani et al., 2017) model to keep a memory statistic of the sequence information. External sources of data such as a pretrained embedding model on a larger corpus could be useful as well and fine tuned for our specific objective.