# Hw1: Presidential Speech Classification with Word Embeddings

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

## Abstract

Text classification in model-restrictive settings is heavily reliant on predictive features to ensure adequate performance. In this case, our task is to predict a presidential candidate by looking at a small text snippet of varying length taken from a transcription of one their speeches. This analysis explores several methods of text tokenization, feature engineering, dimensionality reduction, and model loss tuning. Starting with simple dataset exploration, our experiments extended 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 presidential speeches are very high-dimensional, different regularization methods and gradient solvers 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. Overall, we achieved a boost of roughly 2.5% over the baseline. In addition, we further improved our classification performance by pretraining 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 that could 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 generated as the sum or the logical-OR of the tokens within it.

In general, feature vectors generated in this way tend to be high-dimensional. To combat this curse of dimensionality in modeling, the vocabulary can be shrunk by several methods, including resolving the case of all characters, removing non-alphanumeric character, expanding contractions, lemmatization, and removing words that tend to carry little semantic meaning (*stopwords*). Even with these reductions, which at times may be substantial, the resulting feature vectors tend to still be quite large. Further explicit reductions can come from recognizing that tokens that occur extremely rarely across a corpus likely carry little predictive value and may be ignored. This practice should be conducted with caution; while tokens that occur very rarely (e.g. once) may not be predictive, tokens that occur slightly more often (e.g. in 1% of documents) may be very predictive. The appropriate frequency threshold for ignoring rare tokens varies by application and may need to be discovered empirically. However, once a suitable feature engineering protocol has been defined, the only other tuning possible is incorporating unlabeled data and ablations with training the model itself.

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 as well as apply different gradient based solvers for the optimization.

## 2 Supervised Learning

In this section, we discuss methods of improving classification performance without making use ofunlabeled 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 was 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.

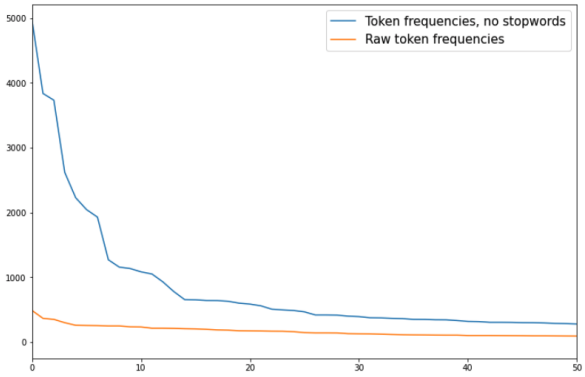


Figure 1: Token frequencies in training data before and after stopwords removed from the dataset

While this simple experiment gave us insight into the distribution of the token frequencies, we still needed to better understand the influence different preprocessing operations had on vocabulary size. As mentioned in the introduction, several techniques exist to reduce vocabulary size while largely preserving the semantic meaning of the input. 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, with the steepest drop in cardinality occurring from the limitation of minimum document frequency to two. This implies that many tokens only occur within a single document and likely only once in the training dataset.

To test different theories about the impact of feature engineering, 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 as additional ablations. 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. That is, if a term occurs rarely across documents, it should be upweighted as it may prove more predictive.

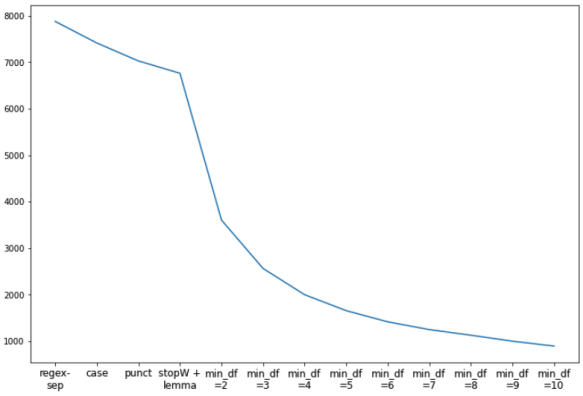


Figure 2: Vocabulary size as successive filters applied to input corpus. From left to right, the strings are regex separated, then case and punctuation is resolved, with final filters trimming rare tokens

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, several findings surprised us. In particular, the removal of stopwords tended to hinder performance or at least not improve it substantially. Tf-idf vectorization, often thought as an improvement to standard counts, actually performed worse in all cases. These two phenomena may be due to the fact that the filler words a can-

didate uses are somewhat unique to their speaking style and therefore predictive. Less surprisingly, nltk's more sophisticated tokenization engine outperformed the more conservative regex counterpart, perhaps slightly due to how it handles contractions (breaking `can't` into `ca` + `n't` to separate the contraction from the root word). The resolution of case, punctuation, and lemmatization also tended to lead to improvements, though at times this boost was not substantial, perhaps due to the fundamential difficulty of the task and the capacity of our model. Fortunately, this grid search did identify a new method of featurizing the input corpus that led to a boost of 2.7% and 2.1% over the validation and testing baselines, respectively. We used the preprocessing protocol that achieved the highest score with the lowest dimensionality for out following analyses.

### 2.2 Dimensionality Reduction

While one method of dimensionality reduction in NLP focuses on trimming out words with little predictive value, the results of the feature engineering study led us to believe this would not be as intuitive as previously thought. 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 set, we were able to exceed our previous performance on the held-out validation set (denoted with red dotted line). For a separate trial, features were normalized before SVD, drastically hurting performance. We posit this is due to the sparsity of the data, where normalization is skewed by the immense presence of zero-terms.

### 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 many regularization strategies including L1 and L2 loss. For comparison, we also ran a trial where no regularization was applied.

In general, L1 loss appeared to be a inferior regularization of the input data, surprising since inducing sparsity on a high-dimensional dataset would improve the predictive power of the model (Turney and Pantel, 2010). Moreover, the non-regularized

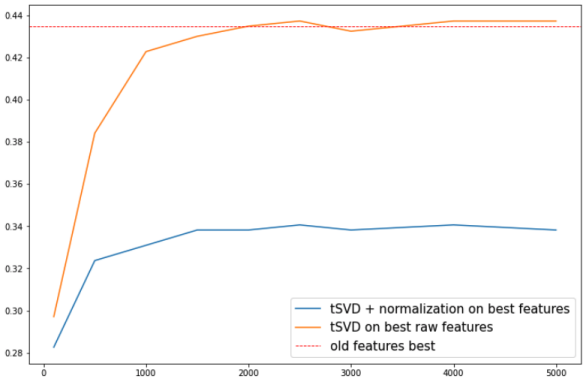


Figure 3: Classification performance applied to best feature selection after Truncated SVD applied with varying numbers of components

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** |

model performed well on the validation set but hurt performance on the testing set, implying the non-regularized optima was not as robust as its regularized counterparts. Affirming this notion, the best L2 loss configuration boosted our performance on our testing submission.

## 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 linear model implementation what 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 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 Princi-

pal Component Analysis (PCA) decomposition of a selection of word embeddings, candidates tend to be co-located together, as do words connoted with conflict or health.

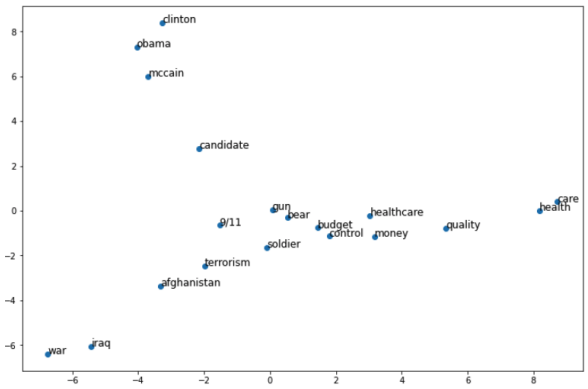


Figure 4: Visualization of Word2Vec embeddings with PCA with a selection of common political terms. Embeddings appear to capture semantic meaning

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 included dataset augmentation with unlabeled data. First, a model is trained with labeled data, then it is iteratively used to label unlabeled instances. If these predictions meet certain confidence criteria, then the sample is added to the training set and the process continues until performance ceases to improve. Though a promising option, we were concerned the approach would cause the trained model to diverge under its own, highly imperfect predictions, leading to poorer performance overall. Empricially validating this assumption is one promising avenue for future research.

### 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 that data. The embedded dimensionality of these words is 300, a reduction of the previous vocabulary size by an order of magnitude. To generalize these word embeddings 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.

We think one potential reason the Word2Vec embeddings alone did not boost or even meet previous performance is due to its averaging of embeddings. 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 also comes with the cost that meaning is obscured 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 Barack Obama and Hillary Clinton, the two most common classes in the dataset, displayed in Figure 5.

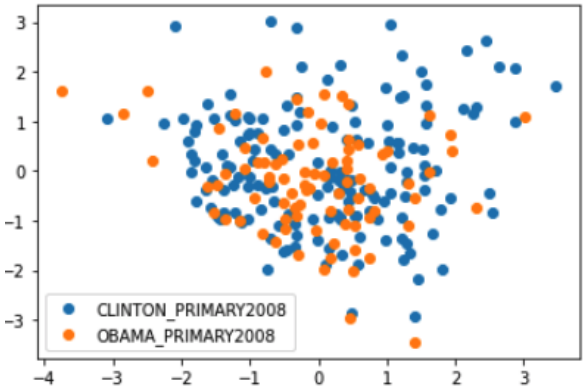


Figure 5: Visualization of political candidate embedding profiles as defined by their speech embeddings. Little separation can be cleaned, likely because order is not considered in embedding creation and candidates discuss similar topics

Examining this phenomena further, we find that among the 100 most common phrases tied to the tokens `obama` and `clinton`, only half of them were unique between them. Indeed, much of what these candidates discuss is interrelated, as would be expected in a presidential race. With that said, there are still several defining features the embedding model was able to discern. 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 others features, namely comments about his citizenship by other candidates and his ascent to president. By contrast, words such as `bubba` allude to her campaign to win the vote of white men in the southern United States.

As shown, there is considerable overlap in what the two candidates discussed, as described by their document embeddings mapped down to two dimen-

sions. Thus, to differentiate candidates more effectively, these 300-dimensional embeddings were concatenated together with the SVD features of the previous analysis. By incorporating embeddings from models with different amounts of labeled data, we explored performance changes with our tuned model and this concatenated feature set.

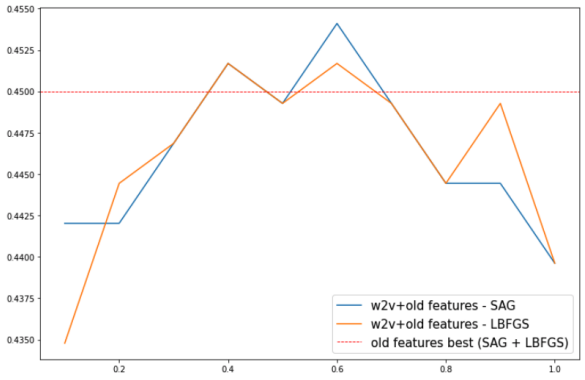


Figure 6: Performance with concatenated Word2Vec document embeddings and different fractions of unlabeled documents and SAG, LBFGS gradient solvers

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 stochasticity in training or other spurious modes of interaction between our original and embedded data representations. Regardless, The performance on our validation set was again improved, though unfortunately this boost in performance again did not translate to the validation set, though it exceeded the baseline.

One reason for this validation-test performance mismatch may be due to the fact that the validation set is 10 times smaller than the training set and 100 times smaller than the testing set. Future experimentation could benefit from a more balanced split. 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 legistlation. 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 in criticism of Democrats by Gingrich, was misinterpreted by the additional context of democrats supporting 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. Additional context does not always improve classification.

## 4 Conclusions and Further Exploration

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. Some defining traits of this project include the restriction to a Logistic Regression classifier, a highly imbalanced split between train, validation, and test data, as well as a difficult classification objective that utilizes a small snippet of a candidate's speech. Because of these defining traits, we were only able to boost performance in this classifier a few percent. With that said, if restrictions regarding models and external data were relaxed, we feel we could boost our performance far higher than the improvement noted in this report.

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) may boost performance by better capturing context between words. In addition, designing a method to incorporate word order would likely improve the expressivity of our features. This could be a 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.

## 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.