Short Text Classification of Political Reddit Posts

Stanford CS224N {Custom} Project

Shirley Cheng
Department of Computer Science
Stanford University
scheng3@stanford.edu

Abstract

Text classification plays a crucial role in natural language processing, enabling applications such as text retrieval, processing, and recommendation systems. This project focuses on the task of classifying the ideological leanings of comments on Reddit, a domain characterized by short and often sparse text in which traditional text classification methods struggle. Leveraging the Word2Vec model, I train word embeddings on the Reddit Corpus. I utilize word embeddings as features in word vectors and experiment with different classification approaches, including: a Support Vector Machine (SVM) classifier, Convolutional Neural Network (CNN) classifier, Long Short Term Memory (LSTM), and CNN-LSTM. I find that a hybrid CNN-LSTM architecture achieves the best performance, at .662 accuracy rate.

1 Key Information to include

- Mentor: Nelson Liu

2 Introduction

Text classification is the task of assigning labels to textual data based on its content. It is relevant to many other natural language processing tasks, including text retrieval, processing, and sentiment analysis. Traditional text classification involved techniques which extract features from text through methods such as TF-IDF, bag of words, and n-grams. However, in the context of short text classification, these techniques struggle due to the sparsity and dimensionality issues, as well as the inability to capture context dependencies.

Recent advances in deep learning and neural networks have been applied to text classification tasks to address these problems. These approaches leverage the power of distributed representations and hierarchical feature learning, leading to significant improvements in performance. For example, word embedding methods such as Word2Vec use distributed representations by mapping dense vector representations in a continuous vector space where semantically similar words are mapped to proximate points. Thus,these embeddings address both sparseness and dimensionality issues. Additionally, they are also capture contextual and semantic similarities between words, providing a more nuanced feature set for classification models.

I examine the problem of short text classification in the context of political ideology. I address the task of classifying ideological leanings of Reddit posts and comments in political subreddits. With rising political polarization, political ideology on Reddit is an important domain to examine the applications of NLP to social sciences. Reddit provides a rich corpus to study political conversations, as subreddit communities are formed along shared political beliefs.

There are several problems associated with the task of classifying Reddit posts and comments by political ideology. Notably, many Reddit posts and comments are sparse and short, often consisting of just a few words or sentences. This limited amount of text can make it difficult for classification models to extract meaningful features and accurately classify the content. Moreover, Reddit users

often employ informal language and may employ domain specific language, such as internet memes, which can be difficult for classification models to interpret correctly.

In addressing this classfication task, I base my approach on deep neural networks methods. I derive domain specific word embeddings by training word2vec model on data from political subreddits. Using word embeddings as features, I experiment with different classification techniques and neural architectures, including Support Vector Machine(SVM), Convolutional Neural Network(CNN), Long Short-Term Memory (LSTM), and CNN-LSTM.

3 Related Work

Mikolov et al. introduced the Word2Vec model in 2013 in which the key idea is to learn distributed representations of words in a continuous vector space. Word2Vec models are typically trained on large corpora of text data, and during training, they learn to map words to vectors in such a way that similar words are represented by vectors that are close together in the vector space. Mikolov et. al were able to train word embeddings on Google News dataset with a vocabulary of 3 million words with embeddings of 300 dimensions(2013).

Zhang and Han showed that using Word2Vec in conjunction with SVM can be used for short text classification tasks in the context of classifying subject category of microblogs (2019). The SVM aims to find a hyperplane that best separates the data into different classes. Zhang and Han trained a SVM classifier on mean word embeddings and obtain a baseline accuracy of .896.

Onan further explored how word embeddings can be used as input features into different deep neural architecture for the task of sentiment analysis of Twitter product reviews(2020). Onan proposes a CNN-LSTM architecture, which yielded a .835 accuracy rate when trained on padded Word2Vec embeddings. The author finds that a combined CNN-LSTM approach outperforms naive CNN and naive LSTM approach. These architectures leverage the strengths of both CNNs and LSTMs, allowing them to capture both local and long-range dependencies in text data.

For the task of classifying political leanings of Reddit posts and comments, I replicate Zhang and Han's approach of training an SVM classifier on mean word embeddings (2019). I also replicate Onan's approach of building a hybrid CNN-LSTM classifier(2020). Reddit is an important domain to replicate these techniques on as Reddit posts and comments may face even more extreme sparseness than conventional classification tasks. For example, it is not uncommon for a Reddit comment to just be "lol nice." Therefore, it is relevant to investigate how well current short text classification methods perform in this domain.

4 Approach

4.1 Word2Vec Model

Word2Vec consists of two main architectures: Continuous Bag of Words (CBOW) and Skip-gram. CBOW aims to predict the target word given its context words, while Skip-gram predicts context words given a target word. Typically, CBOW is efficient and works well with frequent words in the dataset, making it suitable for tasks where word order is less important. Skip-gram, on the other hand, predicts the context words given a target word, capturing detailed semantic relationships and contextual nuances. I train two different sets of word embeddings by using both architectures. Because Reddit users may be more likely to use internet slang, I derive domain specific word embeddings by training Word2Vec on political Reddit posts and comments. I obtain posts and comments from the following subreddits

"2016$Elections$,"$Ask\_politics$","$Republican$","$centrist$","$democraticparty$","$democrats$"

I utilized Convokit to load PushShift.io Reddit Corpus which contains Reddit posts and comments up until October 2018. After subsetting for political subreddits, I yielded 4955615 training examples. I preprocessed the data by stemming, tokenizing, and removing common stop words. I leveraged Gensim's implementation of Word2Vec model to derive word embeddings with 100 dimensions.

To visualize the word embeddings trained through my model, I use Principal Component Analysis for dimensionality reduction.

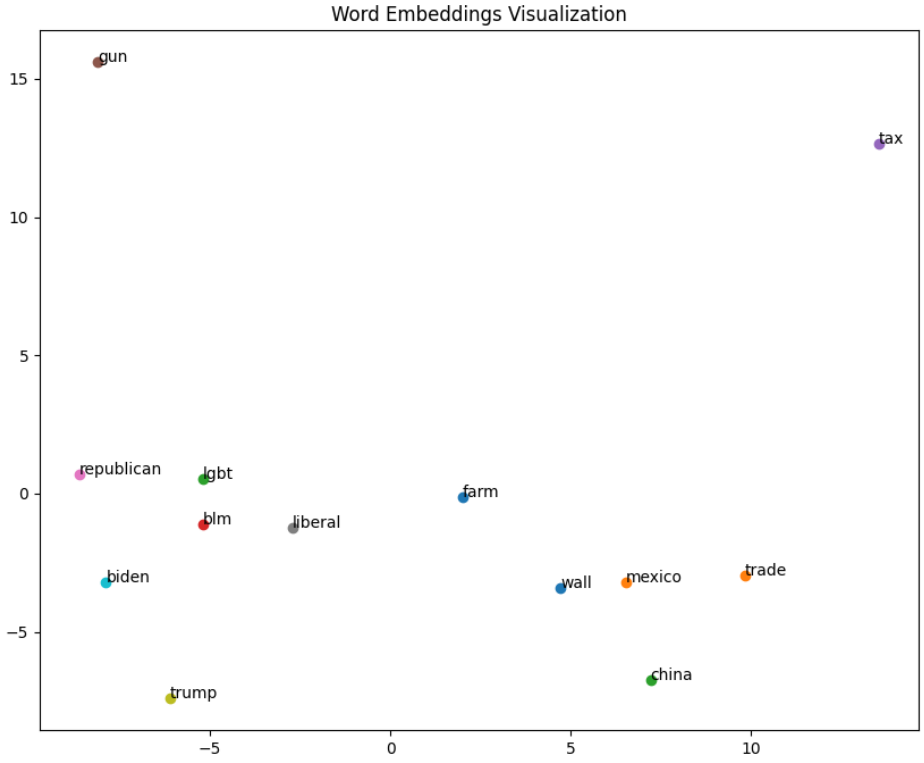


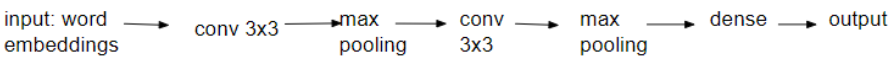
Figure 1: Word embeddings trained using CBOW

4.2 Classification Models

I build four different classification models: SVM, CNN, LSTM, and CNN-LSTM.

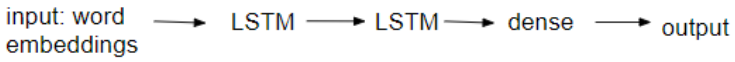
SVM aims to find the hyperplane that best separates the data points of different classes. The hyperplane is a decision boundary that maximizes the margin, which is the distance between the hyperplane and the nearest data points of each class error. I utilize linear kernels in optimizing the hyperplane between the two classes of data.

CNNs, originally developed for computer vision tasks, have been adapted for text classification by treating words or character n-grams as spatial features. Convolutional layers capture local patterns in the input data, making them effective at learning hierarchical representations of text. I utilize the following CNN architecture.

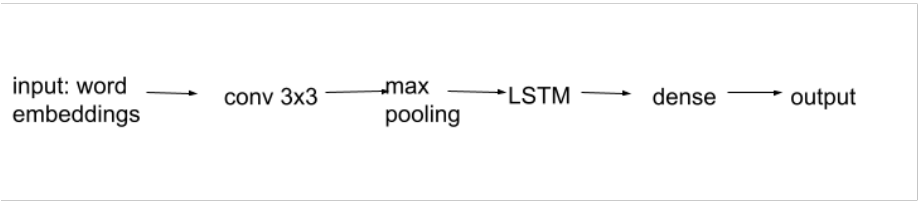


LSTMs are specialized variants of RNNs in which RNNs are designed to process sequential data and are well-suited for tasks involving variable-length inputs. However, traditional RNNs suffer from the vanishing gradient problem, which limits their ability to capture long-range dependencies in text. By incorporating mechanisms such as forget and input gates, LSTMs address the shortcomings of traditional RNNs and are capable of learning long-term dependencies in sequential data. I utilize

the following LSTM architecture. In each LSTM layer, I apply input dropout and recurrent dropout to encourage robustness. Input dropout prevents the model from relying too heavily on specific input features while recurrent dropout prevents overfitting by introducing noise into the recurrent connections.



CNN-LSTM can effectively combine the separate advantages of CNN and LSTM architectures to detect local patterns and long range dependencies. Following from Onan's work which showed the potential of CNN-LSTM hybrid architecture, I utilize the following CNN-LSTM architecture shown below. I also apply input and recurrent dropout in the LSTM layer.



5 Experiments

5.1 Data

My classification task is to classify Reddit post and comments by its political leaning. To create the training dataset for the classification problem, I load posts and comments from r\democrats and r\Republicans through Convokit. Because r\Republicans is more active than r\democrats, I randomly sample 50,000 comments and posts from each subreddit to create an evenly distributed dataset. Each training example is a tokenized and stemmed post/comment. A training example is labelled as 0 if it was posted on r\democrats and 1 if it was posted on r\Republicans. I use a .8/.2 split to create training and test data.

5.2 Evaluation method

In order to evaluate the performance of various classification methods, I compare their accuracy rates using

$$P = \frac{Number of Correct Assignments}{Total Number of Assignments}$$

Accuracy rates are a common metric for performance. Thus, utilizing accuracy rates as a performance metric allows for comparison across different baselines.

5.3 Experimental details

Utilizing the word embeddings I trained through Word2Vec, I obtain mean word embeddings for each training example which I then feed into SVM and CNN classifiers. I also derive sequential word embeddings which I feed into LSTM, and CNN-LSTM classifiers.

I leverage SKlearn's SVM classifier which is implemented using hinge loss for the problem of binary classification. I add a regularization parameter C=.1, which balances the trade-off between achieving a low error on the training data and minimizing the norm of the weights of the decision function.

I implement the CNN, LSTM, and CNN-LSTM models within a Tensorflow and Keras framework. For all models, I use a binary cross-entropy loss.

$$loss = -(y \log(p) + (1 - y) \log(1 - p))$$

I initialize all models with a learning rate of .001 and use Adam's optimizer to increase speed of convergence. Training is performed over 30 epochs with a batch size of 64.

5.4 Results

I obtain the following accuracy rates:

	SVM	CNN	LSTM	CNN-LSTM
CBOW	.632	.578	.627	.662
skipgram	.623	.586	.543	.621

All my models performed considerably worse than the baselines established by Zhang and Onan, who were able to achieve accuracy rates in the high .80s. There are a few factors that influence . First, to reduce memory usage, I trained word embeddings only on six subreddits. On the other hand, Hoffman et. al defined 605 subreddits which compose of the Reddit Politosphere(2022). Thus, my word embeddings may be low quality due to insufficient training data. Additionally, the low accuracy rates also highlight the inherent difficulty of classifying political sentiment based on a singular Reddit comment. For example, one comment appearing on r\democrats reads, "Hahaha!! Not quite, but I'll take it." There is no indication from this comment that it expresses an explicit liberal leaning. Taking into account that there are many comments that do not express explicit political leanings, achieving an accuracy rate of .662 through CNN-LSTM represents reasonable performance.

In general, excluding the CNN model, CBOW embeddings outperformed skipgram. Because CBOW focuses on target word predictions, CBOW embeddings capture better representations of frequent words. Since political discussion often focus on certain key issues(for example, "Trump" is a very common key word), my task was able to leverage CBOW's advantage. On the other hand, Skip-gram performs better when capturing fine grained semantic relationships. However, since my task involves capturing larger political sentiments, it was not able to leverage Skip-gram's fine grained advantage.

6 Analysis

Due the large presence of comments that do not indicate explicit political ideology, my models were only able to obtain moderate accuracy rates. For example, one comment from the preprocessed dataset simply reads:

'sens'

This comment was posted r\democrat, but due to the incoherence of this comment, all four models classified it as Republican.

In contrast, the classifiers performed well on comments that expressed political sentiment. Another comment reads:

'believ', 'republican', 'parti', 'suppos', 'stand', 'candid', 'leadership'

Even though this comment is short, all four models successfully classified this to its true label, r\Republicans.

Additionally, despite the large number of incoherent comments, CNN-LSTM with CBOW word embeddings was still able to achieve reasonable performance of .662. I show that CNN-LSTM is able to effectively leverage the separate advantages of CNN and LSTM to obtain better performance. For example, both CNN and CNN-LSTM were able to correctly classify the following comment as Republican, while LSTM incorrectly classified it as Democrat.

'dunno', 'lot', 'appeal', 'ego'

As another example, both LSTM and CNN-LSTM was able to correctly classify the following comment as Democrat, while CNN incorrectly classified it as Republican.

'obtain', 'gun', 'legal', 'proof', 'gun', 'control', 'work', 'bar', 'bui', 'gun', 'legal', 'obtain'

In this case, LSTM-CNN and LSTM classifiers were both able to capture the long range dependencies of the relationship between guns and gun control.

7 Conclusion

This project investigated the application of deep neural methods to the task of short text classification in the context classifying Reddit posts and comments by political leaning. Despite problems of extreme sparseness in Reddit comments, a hybrid CNN-LSTM architecture was able to achieve reasonable performance of .662 accuracy rate. I showed that CNN-LSTM architecture is able to leverage the separate advantages of CNN and LSTM models to achieve superior performance.

In the context of Reddit posts and comments, most of the problems associated with classifying short texts remain in feature extraction. When comments have sufficient context, deep neural methods perform well. However, when comments are only a single word long, there is simply insufficient information to determine its political leaning. Therefore, a major next step for this project is to augment my data training data with more features. For example, single word comments on Reddit are typically in response to a post or a larger thread. Therefore, I can augment my training data by incorporating the surrounding thread that a comment appears in as features. Additionally, I can also improve the quality of word embeddings by scaling training to include the entire Reddit Politosphere.

All together, deep neural networks demonstrate potential even when dealing with datasets as sparse as Reddit comments. To improve text classification of political discussions, the main steps going forward involve improving input features into these deep neural networks in order to improve feature representation.

References

ConvoKit Developers. (2023). Reddit Corpus (by subreddit). Cornell University. Retrieved from https://convokit.cornell.edu/documentation/subreddit.html

Hofmann, V., Schütze, H., Pierrehumbert, J. B. (2022). The Reddit Politosphere: A Large-Scale Text and Network Resource of Online Political Discourse [Data set]. Zenodo. https://doi.org/10.5281/zenodo.58517293

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J. (2013). Distributed representations of words and phrases and their compositionality. arXiv preprint arXiv:1310.4546.

Onan, A. (2020). Sentiment analysis on product reviews based on weighted word embeddings and deep neural networks. Concurrency and Computation: Practice and Experience, e5909. https://doi.org/10.1002/cpe.5909

Zhang, R. and Han Y. , "Research on Short Text Classification Based on Word2Vec Microblog," 2020 International Conference on Computer Science and Management Technology (ICC- SMT), Shanghai, China, 2020, pp. 178-182, doi: 10.1109/ICCSMT51754.2020.00042. https://ieeexplore.ieee.org/document/944400