# Text Classification of Reddit Posts

**Jacqueline Gutman**
Center for Data Science
New York University
jacqueline.gutman@nyu.edu

**Richard Nam**
Center for Urban Science and Progress
New York University
rn1041@nyu.edu

## 1 Problem Setting

The World Wide Web provides its users with a plethora of resources for discussing and gaining information on many topics. Many of these discussions take place in online forums such as Reddit, where users can submit questions to domain specific communities. Reddit provides its users with access to over 10,000 communities (sub-reddits), with a unique monthly user base close to 200 million. As with many other web forums, Reddit relies on volunteer administrators to moderate questions and answers. Due to the large volume of Reddit posts, an automated method for text classification is needed. In this paper, we present an approach for feature extraction and text classification of posts originating from a limited and diverse set of subreddits. This approach can be implemented to generate suggested forums in which to place a post, and automatically flag moderators on posts that appear to be best suited for a different subreddit, alleviating the need for administrators to manually digest and judge the relevancy of all newly submitted posts in the forum they moderate.

## 2 Related Work

Text classification accuracy on Reddit posts is severely limited by the topic cohesion and appropriateness of the ground truth labels used in any supervised classification algorithm—any set of training data is likely to contain a substantial number of posts which are irrelevant to the subreddit they have been posted under. This label noise in the training data makes it particularly difficult to compare evaluation metrics from this task to performance on similar text classification tasks in other domains. Previous text classification on Reddit posts using 2.5 million posts over 12 subreddits demonstrated worse performance for more complex models involving Latent Dirichlet Allocation and sentiment analysis as compared to simpler unigram bag-of-words models (Giel, NeCamp, & Kader, 2014). The best performance seen in this previous work was a balanced precision, recall, and F1 score of .66 on the test set. However, because the cohesion and quality of posts vary widely from one subreddit forum to another, model performance on different sets of subreddits than those selected for use in our analysis is not strictly comparable to our model performance.

## 3 Methods

In developing the model, we experimented with varying approaches at two distinct phases of model building. First, we tested several distinct methods of feature extraction to represent the text data in either a high-dimensional/sparse or low-dimensional/dense vector space. In the second phase, these vector representations were provided as input features to a number of supervised learning classification algorithms.

In the feature extraction phase, we begin by taking a bag-of-words (BOW) approach at different order n-grams. We also describe an alternative approach where word and document embeddings were learned through a single layer neural net—using Mikolov's Word2Vec (2013) and Le's Paragraph Vector (2014) models—and then provided to the classifier.

In the supervised classification phase, we test the accuracy of a regularized logistic regression, a support vector machine (SVM) with linear kernel and a boosted decision tree. We use a Naïve Bayes classifier as the baseline model, learned from bag-of-words n-gram features.

### 3.1 Data

The data used in this analysis comes from the 2015 Kaggle Reddit competition (Kaggle, 2015). The dataset contains over 1.7 billion posts from the month of May, 2015. Features available in the original dataset include subreddit labels (used as the classification label), the text of the post, as well as metadata about the post, including its

timestamp and the number of up-votes and down-votes the post received.

The original dataset was subsetted to span five subreddit categories. These five categories were chosen to make the task more tractable by selecting five distinct categories where domain specific vocabularies would be less likely to overlap. Of this subset, only post with positive scores were kept, with the assumption that posts with positive scores (i.e. posts which received more up-votes than down-votes) were less likely to exhibit significant label noise that would undermine the interpretability of the results.

The final dataset used in the analysis includes 1,004,560 samples from five sub-reddit categories. The counts by label are as follows:

*Table 1 Frequency distribution of all classes*

| Label | N |
|---|---|
| NFL | 305,556 |
| News | 214,614 |
| Movies | 174,176 |
| PCMasterRace | 170,494 |
| Relationships | 139,720 |

Prior to training the classification models, the data was split into training, validation and test sets. In the bag-of-words n-gram models, the validation and test sets were constrained to be balanced equally over the five labels, and weights were used during training to provide the model with a uniform prior distribution over the label set. In the classification models trained on word and document embeddings, the validation and test data consisted of unbalanced stratified samples of the input, and the classifiers learned the prior distribution directly from the unweighted training data.

### 3.2 Evaluation Metrics

Given the nature of our multiclass problem, the performance of the models included in our analysis can be evaluated with a number of different evaluation metrics, some of which may provide competing objectives. For hyperparameter tuning and cross-validation, we used overall accuracy on the validation set as the objective function to maximize. On the final models, we considered within-class precision, recall and F1 scores, as well as overall model accuracy. For each model, we also computed both macro- (across classes) and micro- (across samples) averages of precision, recall and F1 scores. Typically, model selection decisions were not significantly impacted by choice of metric as the best-performing models tended to outperform other models on any choice of metrics.

Because the models tended to be biased towards the highest recall and lowest precision on both the most frequent and least frequent classes (NFL and relationships), differences in performance that were a function of testing on balanced vs. unbalanced data or using macro- vs. micro- averages tended to average out and ultimately did not affect model selection or alter performance by more than 1.5 percentage points.

### 3.3 Feature Extraction

The first approach we took in model building was to consider three primary types of feature extraction methods. In the first method, features were extracted using a bag of words approach, with n-grams of various orders.

The distributed meaning approaches utilize vector-space representations of the text learned through bulk unsupervised training of a neural network. We first consider a bag-of-word-vectors approach, in which word vectors are combined in an order-independent way that ignores longer-range context dependencies between words outside of the limited context neighborhood of words used in training the word embeddings. Next, we consider a document embedding approach, in which the vector-space representations of documents are learned directly and simultaneously with the vector-space representation of words. Finally, we briefly consider a latent semantic analysis approach for comparison to the word and document embeddings.

#### 3.3.1 Bag-of-words (counts)

In the BOW approach, word counts were collected separately over training, validation, and test sets, and words with count below a frequency threshold of 10 were removed. We considered models with maximum n-gram size up to 4, and for each of these n-gram sizes all lower-order n-grams were included in the count matrix as well. This created a very sparse, extremely high-dimensional vector—the tri-gram model had just under 500 thousand features.

#### 3.3.2 Bag-of-word-vectors (memory free)

In the averaged word embedding approach, a Word2Vec model is trained across a range of window context sizes and dimensionality of the vector-space using the gensim implementation of Word2Vec in Python (Rehurek & Sojka, 2010). Stopwords in the English set of stopwords are removed and words appearing with frequency of

less than 10 counts are ignored. The validation for these hyperparameters is chosen as part of the cross-validation of the classification model. Once these embeddings are learned for all words in the trimmed vocabulary across the entire corpus of documents, the embeddings for all words within a document, ignoring words excluded from the Word2Vec model vocabulary, are combined by averaging the word vectors.

We experimented with both unweighted and weighted averaging schemes, where the weights utilized are derived from term-frequency inverse document-frequency (tf-idf) matrix of weights learned across all classes on the entire corpus of documents. The goal of these weights is to upweight the vectors corresponding to words that are more uniquely identifying or relevant to that particular document while down-weighting words relative to their global frequency across all documents in the corpus.

#### 3.3.3 Document vectors (DM + DBOW)

In the document embedding approach, we adopt the Paragraph Vector approach of Le and Mikolov (2014). With this approach, we train a neural network to learn a set of hidden weights in a document matrix *D* simultaneously with learning the hidden weights in a word matrix *W*. By learning the document vectors directly, we can capture a kind of memory for information in a document that falls outside of the context window of the current word. For greater stability and consistency of use, we learn each document vector with both a Distributed Memory Paragraph Vector model, and a Distributed Bag of Words model, where the model learns the document vectors ignoring the order of context words. The final document embeddings are the concatenation of these two vector-space representations, so that when we specify a model of dimensionality *d*, the length of the final document vectors learned is actually 2*d*.

Because training the document vectors proceeds in an unsupervised fashion, we were faced with an implementation choice that depends on the way such a model might be deployed and whether bulk-training of unlabeled documents prior to classification is possible. If the set of documents we wish to classify is a fixed, closed set, we can train the Paragraph Vector model directly on these documents in order to generate the best dense vector-space representation possible without knowing the text labels. If, however, the model needs to be able to make dynamic predictions at runtime for the most likely subreddit label for a particular post, then we need to infer the document embeddings for the validation and test set from the pre-trained document embeddings learned on the training data. We compare performance on both pre-trained and inferred document embeddings in order to estimate the cost of making these classifications on the fly.

#### 3.3.4 Latent Semantic Analysis

Finally, we used latent semantic analysis (LSA) to generate dense, low-dimensional features from the text by applying singular value decomposition to the tf-idf matrices for the training, validation, and test documents. Reduced dimensions of length 100, 200, and 300 were used to generate the LSA embeddings.

### 3.4 Classification Models

After generating features from the text, one of four families of classification models were used to generate the predicted class labels. These include the baseline Naïve Bayes classifier, logistic regression, SVM with linear kernel, and Adaboost with a decision tree as its weak classifier component. Classifiers were implemented using scikit-learn **(Pedregosa, et al., 2011)**.

#### 3.4.1 Baseline (Naïve Bayes)

The baseline Naïve Bayes model was trained and tested on a uni-, bi- and tri- gram representation of the text. N-grams with counts of less than 10 were dropped from the model. To test predictions on a balanced test set, a uniform prior distribution of the class labels was specified.

#### 3.4.2 MaxEnt (Logistic Regression)

The second classification model we considered was a regularized multinomial logistic regression with L2 penalty (i.e. a MaxEnt discriminative classifier), trained on both the n-gram bag of words and the text embeddings. The regularization hyperparameter, lambda, was tuned using a line search method.

#### 3.4.3 Linear SVM

The third modeling approach utilized a SVM with linear kernel. This model was used on all types of feature representations, including the n-gram feature vectors of different orders, the word and document embeddings from Word2Vec and Doc2Vec, and the LSA embeddings. Multiple instantiations of this model were explored on the validation set with different loss functions (hinge vs squared hinge), type of regularization (L1 vs

L2) and strength of regularization (C), which was found through a line search method. At each validation step, the SVM was fit using stochastic gradient descent over a maximum of 5 passes through the training data, and hyperparameter selection was done by maximizing validation set accuracy.

#### 3.4.4 Adaboost

The final model was an Adaboost classifier that allowed the model to capture non-linearities in the feature space spanned by the data. A decision tree at different depths was used as the weak learner, with depths ranging from decision tree stumps to trees of depth 50. This classifier was trained only on the n-gram features (uni-, bi- and trigrams) and not used on the low-dimensional document embeddings.

## 4 Results

After the feature generation and classification methods were implemented the best classifier from each approach was selected. To identify the best performing model from each model type, the overall accuracy of was used. While a global performance metric such as this provides an easy, intuitive way to describe performance, the within-class metrics (recall, precision, F1) provide additional insight into the particular class labels each model predicted particularly well (or particularly poorly). Below we present the average recall, precision and F1 score, across all class labels for each model with the highest validation set accuracy of its type. Overall, the best performance came from the linear kernel SVM trained on the n-gram features.

*Table 2 Comparison of model performance*

| Model | Recall | Precision | F1 | Accuracy |
|---|---|---|---|---|
| SVM + *BOW* | 0.77 | 0.77 | 0.77 | 0.773 |
| MaxEnt (LR) + *BOW* | 0.78 | 0.76 | 0.76 | 0.760 |
| SVM + *D2V* | 0.69 | 0.68 | 0.68 | 0.684 |
| SVM + *W2V (Unweighted)* | 0.65 | 0.68 | 0.66 | 0.653 |
| Naïve Bayes | 0.68 | 0.65 | 0.65 | 0.649 |
| Adaboost + *BOW* | 0.70 | 0.64 | 0.65 | 0.641 |
| SVM + *W2V (Weighted)* | 0.64 | 0.64 | 0.64 | 0.639 |
| SVM + *LSA* | 0.62 | 0.66 | 0.62 | 0.621 |

### 4.1 Baseline performance

The baseline Naïve Bayes classifier predicted Reddit post over the test dataset with global accuracy of 64.9 percent. The mean precision over the five labels was 0.68, with the classes *NFL*, *PCMasterRace* and *Relationships* showing the highest precision of all classes. The baseline classifier tended to grossly over-predict *Movies*, the third-most frequent class, resulting in good recall but very poor precision (0.48) for movies. The confusion matrix for the baseline model is shown below.

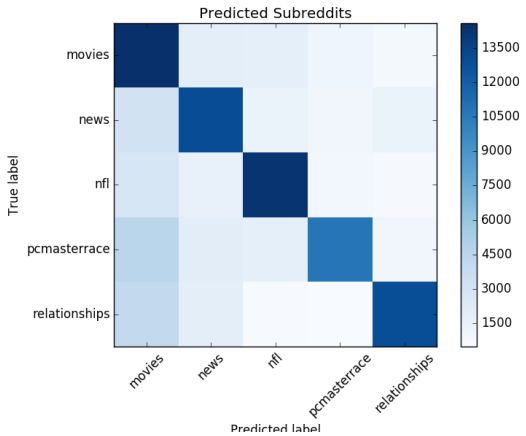


*Figure 1 Naive Bayes classifier trained on n-gram counts*

### 4.2 Bag-of-words classifiers

#### 4.2.1 Multinomial logistic regression

The best performing multinomial logistic regression classifier was trained with uni-, bi- and trigram feature representations of the text. Overall accuracy for this model was 76.7%, where the highest within-class F1 score came from the *relationships* label, at 0.84. The lowest within-class F1 score came from the *news* label at 0.72. Generally, the precision and recall score for each class did not deviate much from its respective F1 score with the exception of the *NFL* class label, which had an F1 score of 0.74, a recall score of 0.65 and precision score of 0.87 (the highest precision of any class), suggesting this classifier tended to overpredict the most frequent class label seen in training.

#### 4.2.2 Linear SVM

The best performing SVM model was trained with uni-, bi- and trigram feature representations of the text, using a linear kernel, hinge loss, and L2 penalty. No promising results were seen from the models fitted with L1 penalty or squared hinge loss, so these parameters were dropped.

The final SVM n-gram model scored the highest in overall accuracy, at 77.3%, of all

models considered. The *relationship*s class label attained the highest F1 value (.84), along with both the highest precision and recall within-class scores (0.84 and 0.83 respectively). In contrast with the logistic regression, the SVM classifier did not overpredict the *NFL* class, achieving a good balance between precision (0.76) and recall (0.80) from the F1 score (0.78). This balance between precision and recall is seen in all the classes under this model, with the largest deviation of precision/recall from its respective F1 score being 0.03. The table summarizes the average precision, recall, and F1 score for the SVM with varying maximum n-gram size of the feature vectors (i.e. the bigram model includes first and second order n-grams)

*Table 3 SVM within-Class Metrics*

| Model | Precision | Recall | F1 |
|---|---|---|---|
| Uni-gram | 0.76 | 0.76 | 0.76 |
| Bi-gram | 0.77 | 0.77 | 0.77 |
| Tri-gram | 0.77 | 0.77 | 0.77 |
| Quad-gram | 0.77 | 0.78 | 0.77 |

The L2 regularization in the SVM model had a major effect on model scores. In the plot below, note the harsher regularization on the L2 penalty resulted in more accurate models. Though the range in which log lambda (C) can produce a consistent model score is relatively large, this supports the idea that C within a certain range will provide a stable model accuracy.

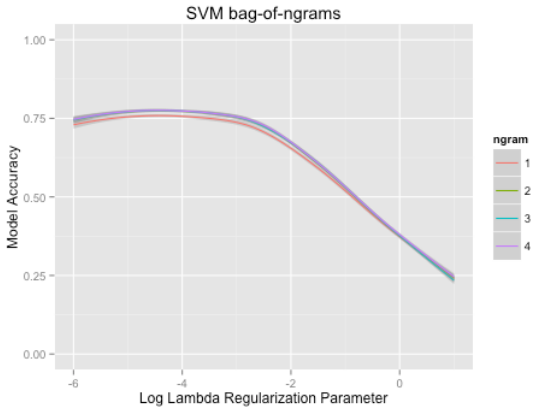


*Figure 2 Performance of SVM on all n-gram models as a function of regularization*

### 4.2.3 Adaboost

The last model to use an n-gram representation of the data was the boosted decision tree, or Adaboost. In this model, increasing the tree depth of the component learners generally improved overall accuracy. A weak probabilistic decision tree classifier at depth 14 was used in the final model, achieving 64.1% overall accuracy. The *relationships* within-class F1 score had the highest value among classes at 0.76. One weakness found in this model was the high deviation between precision, recall and the respective F1 score. For example, the *NFL* within-class precision and recall scores deviate from its F1 score as much as 0.24 (recall). This trend is seen in all classes under this model, but the model especially underpredicts *NFL* (the most frequent class), and overpredicts *Movies* and *Relationships* (the third most frequent and least frequent classes respectively). The table below presents the precision and recall scores for each class.

*Table 4 Adaboost Precision, Recall, and F1 scores*

| Class Label | Precision | Recall | F1 |
|---|---|---|---|
| *Movies* | 0.83 | 0.51 | 0.63 |
| *News* | 0.55 | 0.63 | 0.59 |
| *NFL* | 0.48 | 0.85 | 0.61 |
| *PCMasterRace* | 0.79 | 0.54 | 0.64 |
| *Relationships* | 0.86 | 0.68 | 0.76 |

In the plot below the score of the Adaboost vs weak classifier tree depth in presented. Note the sharp increase in model score when incrementally increasing the tree depth from one. While the loss in model accuracy decreased less quickly when the tree is deeper. The analysis found the model to perform the best at depth 14.

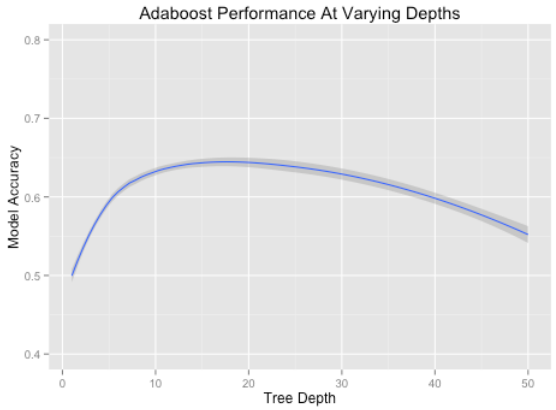


*Figure 3 Performance of Adaboost trained on n-grams as a function of tree depth*

## 4.3 Document embedding SVMs

We compared the model performance of SVM classifiers on the standard Doc2Vec model embeddings to classification performance on test

documents not provided to the neural network in the feature generation phase. Inferring not only the class label but also the feature representation from training data comes at a steep cost, and in our experiments using the inferred document vectors typically hurt model performance by anywhere from 10 to 15 percentage points. Even with this handicap, however, model accuracy remained in the range of 53 to 60 percent, below the performance of our baseline model, but well above the level of chance.

In pre-training the document embeddings, context window sizes were tested ranging from a 5 word to 15 word maximum distance between the predicted word and the context words on each side. Dimensionality of the embedded vector-space ranged from vectors of length 100 to vectors of length 600. Because of the concatenation of DM + DBOW vectors, the document embeddings ranged in size from length 200 to 1200.

The best performing Doc2Vec model was the DM + DBOW model with 500 (1000 total) dimensions, and a context window size of 5 or 10 (this was not a critical parameter in determining model performance). This model achieved 68.4% accuracy, with balanced precision and recall across all classes. Performance was strongest on the *NFL* and *Relationships* classes, although there was slight overprediction of *NFL*, with *Movies* and *PCMasterRace* occasionally, but not frequently, being mistaken for the *NFL* class. The confusion matrix for the Doc2Vec model using an SVM classifier is shown below.

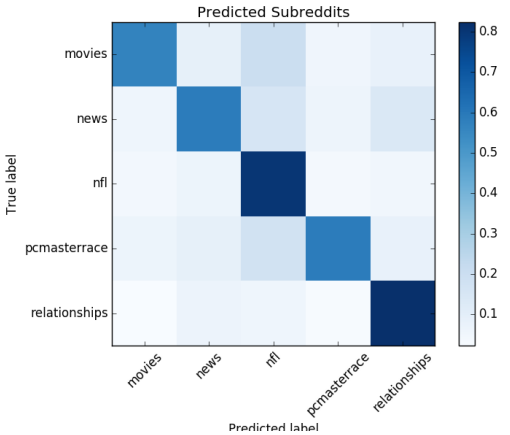


*Figure 4 Confusion matrix for DM + DBOW with SVM*

Overall, hyperparameter tuning of the context window size and dimensionality of the document embeddings had somewhat limited effect on the model performance, with grid search over 46 values of the parameter space yielding performance within a range of 5 percentage points difference in accuracy on the validation set between the worst and best performing models tested, and a range of only 2 percentage points in the unweighted average precision across all classes. There was a weakly evident trend of improved performance with increasing dimensionality of the document embeddings (with performance leveling off or waning beyond about 500 dimensions), and a weakly evident trend of reduced performance with increasing context window size as the window expanded beyond about 6-8 words to either side of the target word. These trends are shown below in Figures 5 and 6.

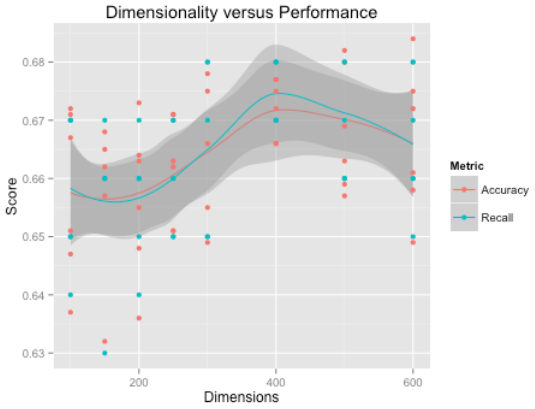


*Figure 5 Fitted trend of dimensionality versus Accuracy and Recall in the Doc2Vec model*

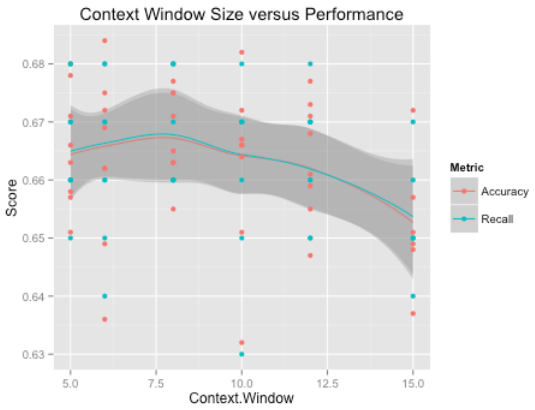


*Figure 6 Fitted trend of context window size versus Accuracy and Recall in the Doc2Vec model*

Performance of this model was significantly worse than the performance of the SVM and MaxEnt n-gram models, increasing the misclassification error rate by about 8 percentage points. This model, however, has the advantage of capturing a significant proportion of the information needed for the text classification task in a significantly reduced number of dimensions. While the n-gram models required about 400 to 500 thousand dimensions to attain the best level of accuracy, the Doc2Vec model with an SVM

classifier required only in the range of 300 to 1 thousand dimensions to achieve results that, if not exactly comparable, may still be acceptable for the purposes of this particular classification task. Because the Doc2Vec model represents the meaning of a particular Reddit post in a distributed way, it is less likely to place all the weight on a few key terms that are highly predictive for the classification task. This may weaken performance, but it may also suggest a more robust model, as the specific terminology and buzzwords associated with each candidate class shift over time.

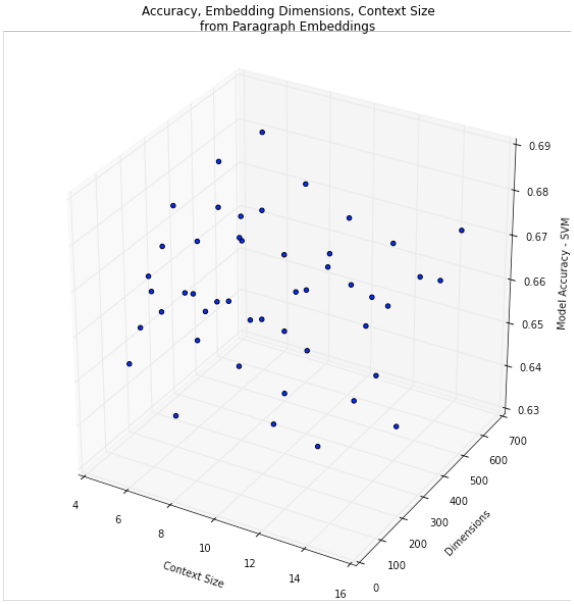


*Figure 7 Performance over grid search space of context size and dimensions in document embedding SVM classifier*

The parameter tuning for these models included a line search across possible values of C for the L2 regularization penalty used in fitting the model. However, because stochastic gradient descent with warm start initialization was used, it was possible to note whether the model had converged by considering successive fits with the same values of C yielded stable accuracy metrics. The maximum number of epochs for the SGD algorithm was set to 5, a sufficient number to reach convergence in the sparse, high-dimensional vector space spanned by the SVM in fitting the n-gram features. With the document embeddings, however, it was less apparent that the model had always converged and achieved a stable measure of accuracy after 5 epochs. While this provided good performance in the n-gram classifiers, the classifiers learned on the document embeddings may require training for a greater number of epochs in order to ensure convergence and potentially improve their performance on the test set.

### 4.4 Average word embedding SVMs

As expected, the model performance of the SVM trained on the averaged word vectors underperformed the SVM trained on the document embeddings. The best performing average word embedding model was achieved using an unweighted average (under the trimmed vocabulary) of all words in a document, using a context size of 5 and dimensionality of 300. Due to runtime constraints of averaging long vectors for all words in all documents, we did not test the averaged Word2Vec model with word embeddings longer than 300 dimensions, but as in the Doc2Vec model, these parameters had minimal effect on the accuracy achieved in the classification task. At all levels of dimensionality, and at all levels of context size, the unweighted average of the word vectors outperforms the tf-idf weighted average.

Here, we chose to average rather than sum the weight vectors in order to normalize for the length of the Reddit post. However, the length of a post may be a highly predictive feature in this problem context, and therefore, using a summation function over all the word embeddings in a document may provide more information to the downstream classifier than simply taking an unweighted or weighted mean.

While tf-idf weights can typically be very helpful in text classification, the downside to using tf-idf weights in this particular domain and problem setting is that the documents that comprise the corpus are relatively short, and it is not clear that words that are particularly relevant or integral to a Reddit post will necessarily be mentioned multiple times in a single document. Additionally, because the weights are computed across classes, these weights may ultimately downweight word embeddings corresponding to words that appear frequently across a large number of documents but only within a single class.

### 4.5 Latent Semantic Analysis SVMs

The reduced dimension embeddings produced by the LSA were provided to the linear kernel SVM in the same manner as the word and document embeddings. The embeddings of length 200 achieved the best performance of the LSA models, but the worst performance out of all candidate models, with an accuracy of 62.1% and a balanced precision rate of 64.4%.

# 5 Discussion

## 5.1 Feature Extraction Discussion

Feature extraction in the analysis involved two techniques. The first was an n-gram bag-of-words method. In this case, a document was represented by a sparse vector of n-gram counts. The second technique involved generating reduced dimension document embeddings through averaging the word embeddings generated by a Word2Vec model, or by concatenating the DBOW and DM vectors, each generated by a Doc2Vec model.

In the bag-of-words models, utilizing higher order n-grams resulted in diminishing marginal returns on model accuracy. In the case of the Naïve Bayes, MaxEnt, SVM and Adaboost, the best performing model (mean accuracy) used a maximum n-gram size of trigrams.

In the example of the SVM, maximum n-gram size of one, two, three and four produce model scores of 0.756, 0.771, 0.773, and 0.770, respectively. With the exception of the increase between first and second order n-grams, the change in model accuracy only changes slightly between models. This behavior was also seen across the other classifiers.

In the second method of feature extraction, vector representations of the text were used, based on word and paragraph embeddings. In both the Word2Vec approach and Doc2Vec approach, the SVM classifier was able to beat the baseline. , but not the n-gram classifier.

One interesting note is that the unbalanced, averaged Word2Vec approach outscored the balanced. This should be taken note of and may be a follow up technique for the bag-of-words n-gram models beyond the work done in this paper.

In the Doc2Vec model, hyperparameter tuning provided small deviations in model accuracy. In the end a large dimension size (1000) with a small context window (5) beat out other iterations of these parameters. More noise may be introduced into the vectors with large context sizes while, larger dimension maybe needed to incorporate information regarding on one of the five classes. In addition, large dimension sizes may be needed since the vocabularies and sentence structures may vary widely, even within classes, due to the nature of web forums such as Reddit. One potential approach to overcome this obstacle could be the application of sentence parsing were a string of text is restructured into a systematic format.

## 5.2 Classifier Discussion

In the analysis four models are tested on the classification problem. These models include a Naïve Bayes classifier, maximum entropy (logistic regression), Support Vector Machine and a boosted decision tree. Of these models the SVM with trigram bag-of-word representations had the best overall accuracy in predicting class labels. The SVM also had the most stable within-class precision and recall scores.

In contrast, the Naïve Bayes, logistic regression and Adaboost classifiers, in at least one class, had large distances between precision and recall scores. For example, in the Naïve Bayes classifier the *Movies* and *Relationships* class labels had considerably large differences between precision and recall. In the logistic regression, the *NFL* class experienced the same behavior. While the logistic regression was an improvement in overall within-class metrics, over the naïve bayes classifier, the Adaboost classifier performed the worst among all models. This non-linear approach did not beat the baseline accuracy score, even while utilizing a deep week classifier, at depth 14. In addition, the stability between precision and recall for all within-class labels was the worst among the models. Given the only slight improvement of the SVM over the logistic regression, the gains from within-class precision/recall stability (ie. distance from F1 score) the SVM may be a more generalizable model for the classification problem.

# 6 Further Research

Beyond the analysis done in this paper, addition research may add to these works by exploring different methods of feature generation. Beyond the bag-of-words n-gram approach, features could be generated based on prefixes and suffixes, along with part-of-speech (POS) and word shape. Incorporating metadata from the Reddit posts themselves, and potentially weighting the training data so as to train more heavily on highly up-voted, potentially more topically relevant posts could improve performance.

Although we explored TSNE dimension reduction and k-means clustering on the document embeddings, preliminary results suggested a lack of cohesiveness within several of the subreddits we explored. Further research might explore relative document similarity within the available forums to identify more homogenous subreddits that could potentially be more amenable to the task of text classification.

## References


Giel, A., NeCamp, J., & Kader, H. (2014). *r/Classifier - Subreddit Text Classification.* Retrieved from stanford.edu: http://cs229.stanford.edu/projects2014.html

Kaggle. (2015, October). *Reddit Comments*. Retrieved from kaggle.com: https://www.kaggle.com/c/reddit-comments-may-2015/data

Le, Q., & Mikolov, T. (2014). Distributed Representations of Sentences and Documents. In E. P. Xing,, & T. Jebara (Ed.), *Proceedings of the 31st International Conference on Machine Learning. 32.* Beijing, China: JMLR.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani , & K. Q. Weinberger (Ed.), *Advances in Neural Information Processing Systems 26 (NIPS 2013).*

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). *Journal of Machine Learning Research, 12*, 2825-2830.

Rehurek, R., & Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* (pp. 45-50). Valletta, Malta: ELRA.


## Honor Pledge

I pledge my honor that all the work described in this report is solely mine and that I have given credit to all third party resources that I have used.

- Richard Nam
- Jacqueline Gutman

## Code

The code used in these analyses is available at github.com/richardNam/koding and github.com/jgutman/koding.

Analyses were primarily constructed using the sckit-learn and gensim toolkits in Python. Plots were constructed using the matplotlib toolkit in Python and the ggplot2 toolkit in R.