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.

1. 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. 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 forum to another, model performance on different sets of subreddit labels than those selected for use in our model is not strictly comparable to our performance.

1. 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 (Word2Vec and Doc2Vec) 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.

* 1. Data

The data used in this analysis comes from the 2015 Kaggle reddit competition. 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. 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 as follows.

|  |  |
| --- | --- |
| Label | N |
| NFL | 305,566 |
| News | 214,614 |
| Movies | 174,176 |
| PCMasterRace | 170,494 |
| Relationships | 139,720 |

Prior to training the classification models, the data was split into training, validatation 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.

* 1. 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.

* 1. Feature Extraction

The first approach we took in model building was to consider three different types of feature extraction methods. The first method involves using a bag of words, or bag of n-grams at different orders.

The distributed meaning approaches utilize vector-space representations of the text learned through bulk unsupervised training of a neural network. We 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 representation of documents are learned directly and simultaneously with the vector-space representation of words.

* + 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.

* + 1. Bag-of-word-vectors (memory free)

In the average word embedding approach, a Word2Vec model is trained across a range of window context sizes and dimensionality of the vector-space. 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 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 weights learned across 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 how frequently they appear across all documents. The downside to using tf-idf weights in this particular domain and problem context 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.

* + 1. 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.

* 1. Classification Models

Once a feature representation has been generated 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.

* + 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.

* + 1. 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.

* + 1. Linear SVM

The third modeling approach utilized a SVM with linear kernel. This model was used on both the n-gram feature vector, at different orders. In addition, the vector representations of the text were feed into this classifier. Multiple iterations of this model were completed with different loss function (hinge vs squared hinge), regularization (l1 vs l2) and levels of regularization, C, which was found through a line search method.

* + 1. 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.

1. Results

Below we present results for each of the classifiers and feature extraction methods we tested. Overall, the best performance came from the linear kernel SVM trained on the n-gram features.

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

* 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.

* 1. Bag-of-words classifiers

[subsection 4.1 stub]

* 1. 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 distance of 5 words to 15 words 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 length 1200.

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 2 percentage points in the unweighted average precision across all classes.

NEED TO ACTUALLY DISCUSS RESULTS HERE

The parameter tuning for these models included a line search across possible values of lambda 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 lambda 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.



* 1. 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

1. Discussion

[section 5 stub]

* 1. Feature Extraction Discussion

[section 5.1 stub]

* 1. Classifier Discussion

[section 5.2 stub]

1. Further Research

[section 6 stub]

References

**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