CS 5830 - Project 5

Chetan Birthare, Zion Steiner

INTRODUCTION

After millennia of development, modern human language is capable of communicating complex information in precise terms. Thousands of words with unique meanings allow for the construction of an infinite number of documents. Documents of similar type often share discernible feature patterns, such as document length, word frequencies, or semantic structure. Deriving these features allows for the classification and grouping of documents. This is very useful for applications where document-indexing is valuable, such as automatic library organization, code indexing, or even document forgery detection.

One machine learning model suitable for text classification is naive Bayes. Given a list of document feature sets, naive Bayes works by calculating the conditional probability of a document belonging to class Y given the occurrence of document feature values X1, …, Xn. The goal of this project is to use this technique to classify which subdomain a document comes from within the popular content-sharing site Reddit.com.

Reddit is a multi-interest forum divided into communities called "subreddits." Each subreddit is focused on a particular kind of content, such as legal advice, comic books, or cat pictures. Using common subreddit post idiosyncrasies identified by naive Bayes, we were interested in predicting which subreddit a given document is likely to belong. After trying two text feature engineering techniques, we show that naive Bayes can achieve a cross-validated F1 score of 90.3% on a dataset containing the post titles of five subreddits.

DATASET

The PRAW Python wrapper for Reddit was used to collect subreddit post titles. Because some subreddits overlap in terms of content, posts were only collected from five thematically disjunct subreddits. These are:

- r/AskReddit for asking questions about anything
- r/WritingPrompts for providing a story prompt for other users to write about
- r/todayilearned about interesting facts
- r/worldnews for world news
- r/UnethicalLifeProTips for useful life advice

The 13,000 top post titles were collected from each subreddit for analysis. Figs 1, 2, and 3 show the characteristics of words included in these titles.

ANALYSIS

To eliminate useless words (data), known as stop words, such as "the", "a", "an", "in" etc, we used Natural Language Toolkit's nltk.corpus to import a predefined set of stop words. Apart from that, we removed subreddit-specific acronyms from all titles to remove in-text labeling. For

example, posts from r/todayilearned commonly contain the acronym “TIL”, which is a self-identifying feature. While training a model on documents that include these acronyms would be more suited for practical use, we are more interested in the effectiveness of the model on non-self-identifying documents.

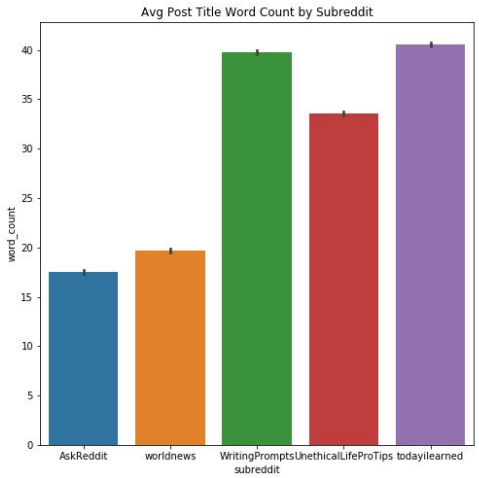

Fig 1. r/TIL has the highest average word count

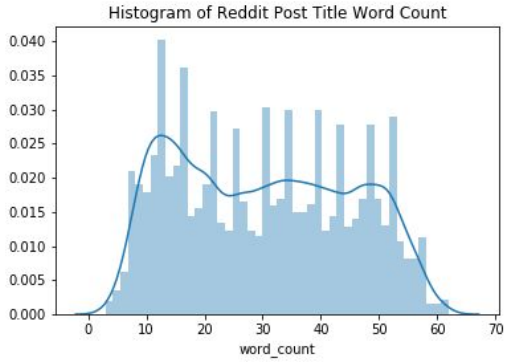

Fig 2. Word count is distributed heavier at the tails than in the center.

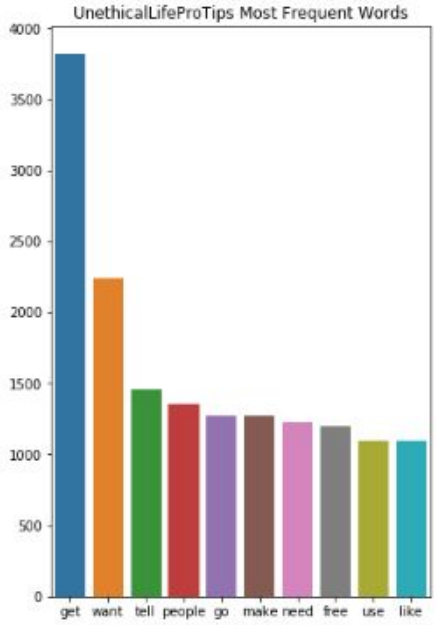

Fig 3. The most common words in r/ULPT post titles

We used two feature extraction methods for generating inputs for model prediction. The first is word-count vectorization, where each document is represented by a vector of length |unique

words in corpus| with values corresponding to the counts of the words that are mapped to the vector indices. The second method is term frequency-inverse document frequency (tfidf) matrixization. This method is similar to word count vectorization, but it adds a penalty to the value of terms corresponding to how often they occur throughout the entire corpus. This technique aims to more accurately represent the relative importance of each term in a document. These techniques were performed separately to compare the performance of models trained on each of them.

Sklearn's MultinomialNB was used to classify this dataset because of its ability to handle vectorized documents.

RESULTS

After building the feature sets, model metrics were recorded from 10 fold stratified cross-validation. Surprisingly, models trained on the word-count vectorization dataset had f1 scores 0.004 higher than those trained on the tfidf dataset. An independent two-sample t-test found these results to be non-significant.

We also tested if title length affects model f1. We did this to see if the feature extaction methods perform better or worse depending on document length. Title length among posts in the dataset ranges from 3-62, so posts were divided into six groups according to length. The results for this are seen in Fig 4.

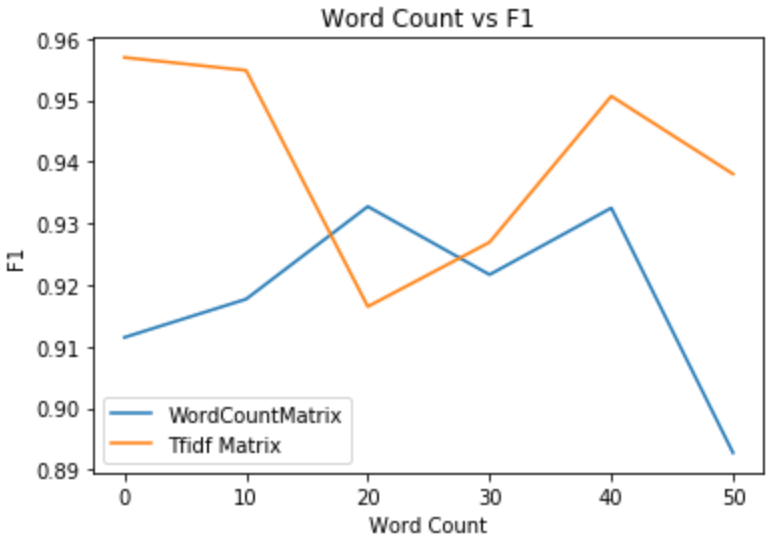

Fig 4. Tfidf does better at most title subsets

It appears that there is some variation in f1 score depending on title word count, with Tfidf having a higher score on average. This is surprising considering how word-count vectorization

had an overall higher f1 score. However, this may be insignificant considering the insignificant t-test results.

For practical application, the best model was one trained on word-count vectorized post titles with subreddit specific acronyms included. The results from each model are in Table 1.

Method	F1	Precision	Recall
Count, stopwords included	0.929	0.934	0.930
Count, stopwords removed	0.823	0.831	0.828
Tfidf, stopwords included	0.887	0.894	0.888
Tfidf, stopwords removed	0.819	0.827	0.821