CS221: Project Progress Report

Nimisha Tandon
nimisha@stanford.edu
Shaila Balaraddi
shailaab@stanford.edu
Naman Muley
ngmuley@stanford.edu

November 15, 2019

1 Introduction

We are tackling the problem of identifying articles that could be potentially dangerous to a brand. Our approach has two steps. First, we take in a set of articles and classify them into categories. In the final application, this could be done by pulling articles from the internet daily. Once these categories are identified, in the second step, we create an impact score for the brand in question. The articles with top impact scores can be provided to the client as having high potential for impact to the brand.

This problem has a rich application of various algorithms for natural language processing. Since proposing this project, we have explored a few algorithms to create a better sense of what an impact score can be. We looked into utilizing unsupervised learning algorithms like GloVe and supervised algorithm implementations like Naive Bayes. These have given us better impact scores. Further more, it feels like we could perform some more fine tuning to come up with better models to bring a richer understanding of the *impact score*

2 Methodology

2.1 Classification

In making progress for the project, we decided to give more weight to the impact score analysis section, since that's the more novel component of the solution approach. Our classification currently still uses a simple Stochastic Gradient Descent classifier. We found it's accuracy to be nearly 82 percent and thought we can improve upon that in the later parts of the project.

We still did look at using the Naive Bayes algorithm to perform classification, since that's what literature reported is it's primary use. It performed slightly better at classification than SGD but not by much.

2.2 Impact Score

To calculate the impact score, we explored multiple algorithms. Once a document is classified based on categories using the CountVectorizer, TfidfTransformer and StochasticGradient Classifier, we could calculate the impact score of a given category in a given test data/document. Following are some of the algorithms we explored.

2.2.1 TFIDF

Term Frequency-Inverse Document Frequency commonly known as TF-IDF can be used to find the frequency of vocabulary words in a document to perform sentiment analysis or extract relevancy from a document.

This algorihm has 2 algorithms working together.

- Term Frequency

$$tf(t,d) == \frac{noofoccurrencesoftermindoc}{totalnumberofallwordsindocument}$$

Before using this algorithm, we first do some preprocessing of the doc as follows:

- Remove all stop words.(ex: in, the,are it)
- Convert all words to lowercase.

  Now using the term frequency alone will end up giving us words that are not unique (for ex: repeated words, should not add up to the relevance of the document)
- Inverse document frequency

This gives us the uniqueness of a word:

$$idf(t,d) == log(\frac{nooftimesthetermappears}{noofdocumentscontainingtheword})$$

The final step is to multiply the two together to get the TFIDF score which would give us the impact score:

2.2.2 Naive Bayes

We explored and implemented a Naive Bayes classifier as an alternative classifier to build the model.

The Naive Bayes algorithm provides a very simple framework that builds on the probabilistic model based on training data. The algorithm is based on a statistical classification method called Bayes Theorem.

In order to understand how naive Bayes classifiers work, we can represent Bayes rule as the following probabilistic model as follows:

$$Posteriorprobability = \frac{conditionalprobability * priorprobability}{evidence}$$

Bayes' theorem forms the core of the whole concept of naive Bayes classification. The posterior probability, in the context of a classification problem, can be interpreted as: "What is the probability that a particular object belongs to class i given its observed feature values?" The answer to this question gives us a score of how much a document is relevant to a given topic.

Impact score needs to be a number that represents how relevant or impactful will a particular article be to the brand. Hence, we extended our previous approach of having a dictionary of words that are relevant to the brand and applying some NLP algorithms to come up with a score of how closely does the article's content relate to the dictionary relevant to the brand.

3 Algorithm

3.0.1 GloVe

3.0.2 Naive Bayes

The Naive Bayes algorithm: Naive Bayes consists of multiple algorithms, like Multinomial Naive Bayes and Multi-variate Bernoulli Naive Bayes.

We found that the Multinomial Naive Bayes algorithm is well suited for our project. This is because this algorithm gives us the "term" frequency of a dictionary of words of a specific brand. The term frequency is defined as the "number of times a given term appears in a given document". In practice the term frequency is often normalized by dividing the raw term frequency by the length of the document.

The term frequency can then be used to compute the max likelihood estimate based on the training data.

The algorithm to use for training using Naive Bayes:
function Train NaiveBayes(D, C)
for each category:
Calculate the prior probability num-docs = number of docs in D
num-category = number of docs in category

$$priorprobability[category] \leftarrow log\frac{N_c}{N_{doc}}$$

```
V = vocabulary of D
docs[category] <- append d for d subset of D in specified category
for each word in V:
Count(w,c) <- num of occurences of w in docs[category]
```

$$priorprobability[category] \leftarrow log\frac{count(w,c)+1)}{\sum_w count(w,c)+|V|)}$$

```
return priorprobability
```

We can then use the priorprobability to test our data to get the maximum likleihood.

4 Implementation

Talk about how the models are trained, what data sets did we use for these models. how are the data sets relevant. Future work may involve using and creating better data sets

5 Preliminary Results

Show similar words, maybe a fancy representation of similar words and important words as found by GloVe and TF-IDF respectively

6 Future Work

1. Get better training data for the algorithms 2. Have the application fetch testing data from today's newsfeed