Stanford University

CS221 Final Project - Impact Lens

Nimisha Tandon
Naman Muley
Shaila Balaraddi
December 14, 2019

1 Introduction

A lot of information on the internet like news, social media posts etc that can impact brands. Directly, in the form of positive or negative PR and indirectly in that their product strategy could be potentially informed from these. Identifying such news is extremely useful for large brands or analytics divisions to get ahead of the PR cycle and formulate an early response.

We attempt to identify information that can be potentially impactful to a brand. We decided to tackle this problem by using classification algorithms and impact analysis methods. This project focuses on using classification to break down the documents into various categories and then look through the lens of a specific brand to calculate an impact score of each document. We present a stack rank of such documents back to the brand.

2 Method Overview

Approach This project can be broken down into two stages: a) Classification : This is the component which given a bunch of documents is able to classify the category that they belongs to and give them a label. b) Impact analysis. : This is the component, where given an article/articles and a bunch of words for the domain of the article, it is able to give an impact score for it , indicating how significant this article could be.

Both of these operations will deploy ML techniques. The system will take as input some domain data which represents the brand. It will also present to the user certain categories or news groups that the brand is interested in.

3 Classification

3.0.1 Stochastic Gradient Descent with TFIDF Vectorizer

For classification of the documents into categories, we used the Logistic Regression classifier with TfIdf vectorizer which uses Stochastic Gradient Descent for optimization. SGD is a simple and

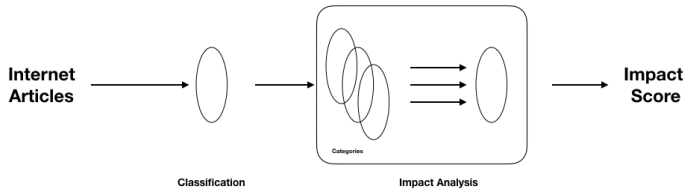

Figure 2.1: Process Overview

efficient optimization method which is used for learning of linear classifiers.

An SGD randomly chooses training data, gradually decreases the learning rate, and penalizes data points which deviate significantly from what's predicted. Since we had a large dataset to go through, SGD was the optimal and simplest algorithm to use.

For the purpose of this project we used the SGDClassifier provided by scikit-learn for training our models.

3.0.2 Feature Extraction process

TF-IDF which stands for Term Frequency – Inverse Document Frequency. It is one of the most important techniques used for information retrieval to represent how important a specific word or phrase is to a given document.

The tf-idf value increases in proportion to the number of times a word appears in the document but is often offset by the frequency of the word in the corpus, which helps to adjust with respect to the fact that some words appear more frequently in general.

$$tf(t,d) == \frac{no\ of\ occurrences\ of\ term\ in\ doc}{total\ number\ of\ all\ words\ in\ document}$$

Inverse document frequency

This gives us the uniqueness of a word:

$$idf(t,d) == log(\frac{no\ of\ times\ the\ term\ appears}{no\ of\ documents\ containing\ the\ word})$$

$$TfIdf(t,d) == tf(t,d) * idf(t,d)$$

3.0.3 Model fitting

We then fit the model with pre-processed training data and labels. We feed the model with training data and their categories to train it to accurately classify the documents into respective categories.

Classify We then run classification of test data on this trained model to get dataset categorized into various topics.

Model Tuning and Optimization For the purpose of tuning the models , we used the Grid Search where the Hyper-parameters for training the models were varied and the best Hyper-parameters where chosen to finally build the model/s based on the outcome of the F1 score from Grid Search. The table below talks about some of these hyper-parameters which were applied during building of the model and the impact of those parameters on the final F1 score.

Classifier/Hyper-parameters	TFIFD transformer	Loss	Regularization	Max Iterations	Alpha	Precision	Recall	F1
SGD	Yes	Hinge	l2	20	1.00E-03	0.825490583	0.8125149	0.81088067
SGD	Yes	squared hinge	l2	20	1.00E-03	0.836172017	0.8260603	0.826460415
SGD	Yes	modified_huber	l2	50	1.00E-03	0.836186256	0.8261859	0.826561936

Figure 3.1: Grid Search Results

4 Impact score analysis

4.0.1 Similar Words using GloVe embeddings

A measure of how relevant an article is to a brand is to know how many of semantically or linguistically similar words occur in that article. For each of the words in the vocabulary, we get a list of *similar words* using the GloVe learning algorithm [1]. We then use a simple TF algorithm to understand how commonly do these similar words occur in an article.

Algorithm 1: Extend Vocabulary with Similar Words using Glove Model

$Vocabulary \leftarrow$ ["guns","rifle","weapon","nra","handgun","firearm"];
$model \leftarrow train\_glove$(glove.6B.300d.w2vformat.tx);
GetGloveWords ($Vocabulary, model$)
    $extended\_vocab \leftarrow Vocabulary$;
    foreach *word* $w \in Vocabulary$ **do**
        $similar\_words \leftarrow model.get\_similar\_words$(word=w, $topN = 10$);
        $extended\_vocab.extend$(similar_words);
    end
    return $extended\_vocab$;

Once the vocabulary is extended, we use this extended vocabulary to identify articles that have the most occurrences of these words. In the baseline program, we had used a similar algorithm to calculate, with the vocabulary as the raw input set of words. Following is a figure that shows how the expanded vocabulary behaves for a brand like NRA and input vocabulary = *["guns", "weapons", "nra"]*.

We use this extended vocabulary to come up with an impact score by simply counting frequencies of these words. Our algorithm that performs this to come up with an impact score for each article in the category is given below. The score dictionary contains the impact score calculated for each article in the set of articles.

We have detailed our results of the above score dictionary in the Results section.

4.0.2 Relevant words using TF

Term Frequency is used to find the frequency of vocabulary words in a document to perform impact score analysis by extracting relevancy from a document. Before using TF to calculate, we perform a basic text pre processing on the categorized documents.

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

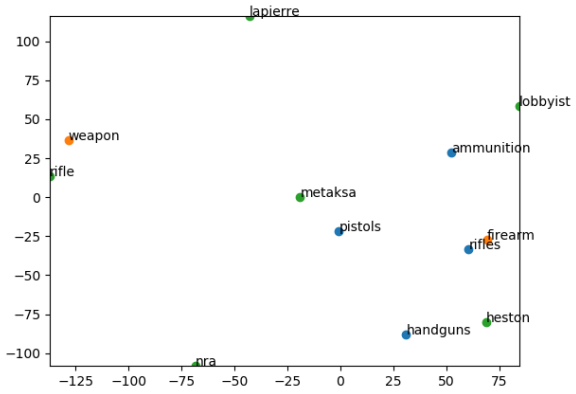

Figure 4.1: Similar Words obtained from GloVe Learning for *NRA*

- Convert all words to lowercase.

We then use the term frequency calculation to calculate the impact score..

- Term Frequency

$$tf(t,d) == \frac{no\ of\ occurrences\ of\ term\ in\ doc}{total\ number\ of\ all\ words\ in\ document}$$

5 Data

We used the following data sets for various learning algorithms.

1. ***20-News:*** The newsgroup data set contains 18000 news posts on 20 topics split. We used this data set to train the classifier and to come up with fundamental categories for the news articles.

   The categories provided by this data set seemed to work just fine for our purposes. Following are the categories that this data set provides. To showcase our system and it's functionalities, we created a fictitious client NRA and chose the *talk.politics.guns* category to work upon in detail.

   a) comp.graphics
   b) comp.os.ms-windows.misc
   c) comp.sys.ibm.pc.hardware
   d) comp.sys.mac.hardware
   e) comp.windows.x
   f) misc.forsale
   g) rec.autos
   h) rec.motorcycles
   i) rec.sport.baseball
   j) rec.sport.hockey
   k) talk.politics.misc

Algorithm 2: Calculate Impact Score for each article from a list of articles, given extended vocabulary

CalculateVocabImpactScore (V, D)

inputs : A list of words that form ExtendedVocabulary V; a list of articles D
output : A dictionary representing impact scores for all articles in D
$score \leftarrow \emptyset$;
$RangeD \leftarrow range(0, len(D))$;
foreach *index* $i \in RangeD$ **do**
    score[i] = 0 ;
    freq = [] ;
    /* iterate over all words in i_{th} article to count frequency */
    foreach *word* $w \in D[i]$ **do**
        freq[word] += 1 ;
    end
    /* iterate over all words in V to build a score for article i */
    foreach *word* $w \in V$ **do**
        score[i] += freq[w] ;
    end
end
score.Normalize() ;
return $score$;

l) talk.politics.guns
m) talk.politics.mideast
n) sci.crypt
o) sci.electronics
p) sci.med
q) sci.space
r) talk.religion
s) alt.atheism
t) soc.religion.christian

We used the 20 News group to also assess accuracy of our classifiers. The test data split out of this dataset was used to assess the accuracy of the classifiers.

2. ***Wikipedia 2014 + Gigaword 5*** The GloVe enhancement used the Wikipedia 2014 dataset with 300 dimensional vectors. This dataset helps get a set of enhanced positive words that complement a given word. For the input set of words, we find the positive GloVe embeddings and include them in an enhanced vocabulary, which we can then use for impact score algorithms.

3. ***New York Times articles:*** A set of 6179 articles from New York Times archive of December 2018. These provide a perfect testing dataset to see how our system classifies articles into categories and provides impact scores for each.

6 Experiments

Below are some of things we tried in addition to those listed above, however did not chose to include them in the main flow because of lack of good results using the same. 1. Aspect extraction and their polarity extraction for Impact Analysis.

Figure 6.1: Aspect and Polarity of Aspect Extraction

Aspect is For the purpose of extracting the impact score it seemed almost natural to think of some things about the product and or brand in question that the people would be talking about. Which led us to the notion of extracting Aspects from the articles and then studying its polarity in order to further understand if that aspect is being positively or negatively perceived.

As described in the diagram above, Aspects were extracted using a very rudimentary approach: Step 1: Here we clean the articles, by removing all stop words Step 2: We then tokenize the articles using a Sentence tokenizer Step 3: POS Tagging for each token in the sentence were then identified. Step 4: We then filtered those tokens which were Noun and Noun Phrases Step 5: Using the tokens which were nouns or noun phrases we then identify the Aspects and also the associated orientation/polarity.

After passing the articles through the pipeline of the aspect and sentiment extraction, below are some of the results that were obtained:

Token	Sentiment	Score
EXISTENCE	Positive	66.67
IMPORTANCE	Positive	50
UNITED	Positive	5.56
DISCUSSION	Positive	42.86
STATEMENT	Positive	41.67
SAKE	Positive	35.29
DESIRE	Positive	33.33
NEW GUN WEEK	Positive	33.33
SECOND AMENDMENT FOUNDATION	Positive	33.33
PART	Positive	32.43
ONE	Positive	31.91
SECURITY	Positive	31.48
GUN CONTROL	Positive	30

Figure 6.2: Aspects and Polarity Score of Aspects

As the results indicate, since the aspect extraction is purely based on the Noun words and Noun phrases, they are not all truly aspects. Additionally some of these identified as Aspects do not depict the correct polarity hence we decided to not include these results in the main stream for calculating the impact score.

7 Results

We wanted to showcase a few sets of results from our experiments.

7.1 Classification: Naive Bayes vs. Stochastic Gradient Descent

The SGD classifier did slightly better than the Naive Bayes classifier. We used the test data split in the News20 group dataset to test out our classification and verify with the labels already present in the dataset.

The Naive Bayes classifier had an accuracy of 0.80 and the SGD classifier with hinge loss had an accuracy of 0.82. We believe that is too close a difference to conclusively say one is better than another.

7.2 New York Times results

The NYT archive of 2018 December provided a great real world ground for our complete end-to-end testing. We went with the fictitious scenario that NRA is our client and they have selected the category *guns*. They have then provided the following set of words as vocabulary: *["guns", "shooting", "lobby", "rifles", "weapon", "nra", "handgun", "politics"]*

Following were some of the noteworthy results:

Article Index (out of 6200)	Score	Article Index (out of 6200)	Score (with GloVe)
2888	0.060	3967	0.071
158	0.057	2888	0.044
182	0.057	158	0.044
1492	0.043	182	0.044
1502	0.043	5237	0.039
3967	0.040	5255	0.039
3834	0.032	1492	0.038
188	0.031	1502	0.038
632	0.031	4846	0.031
5224	0.025	3834	0.028

Figure 7.1: NYT Articles Stack Rank Without GloVe embeddings and With Glove Embeddings

- ***Classification*** The SGD classified 189 articles into the *talk.politics.guns* category. Articles were on variety of topcis like gun shootings, poaching in africa, drug cartels around the world, police aggression in US etc. All of these were rightly classified.
- ***Impact Scores***

  We found the impact scores change after enhancing the vocabulary with GloVe embeddings. With the raw input vocabulary, the articles that came in the top 10 index included the following:

  1. Article 2888: Mass shootings in Parkland, FL and gun restrictions
  2. Article 158: Police shootings of black population
  3. Article 3967: Nigerian Military shooting unarmed protesters

  After the use of GloVe embeddings, with the vocabulary that included the positive GloVe embeddings, we found the articles change impact scores. New articles came into the top 10, while those that were present changed their positions. For example we now found *Article 3957 - Use of American guns in poaching in Mozambique* entering into the top 10. This article

talks about shipment of American guns being sent over for illegal activities in Africa and can be important from a strategy perspective for NRA, our fictitious client.

8 Conclusion

9 References