News Summarizer and Analyzer

Artificial Intelligence Project Report

Nidhi Panpalia
SBU ID : 110937180
{npanpalia@cs.stonybrook.edu}

Rushabh Shah
SBU ID : 110974349
{rumshah@cs.stonybrook.edu}

Siddharth Shah
SBU ID : 110957500
{siashah@cs.stonybrook.edu}

Abstract — The speed by which the information is growing on the Internet today has no bounds. For a user to read about particular topic can be painful given the amount of information. When it comes to news for a particular person/topic, almost all news articles on Internet give repetitive information other than few little new details in every article. So we decided to provide a summarization tool which aggregates important information from various news articles available on the Internet and provide a brief summary on it. Also, we further found tweets mentioning this news articles and performed sentimental analysis on them giving the user an overall view of the opinions of public for that news article.

I. INTRODUCTION

Problem Definition:

The application takes user query as input. This query can be any search term. Search term can be space separated as well. The application should fetch relevant news articles and summarize them. Also, fetch the tweets which have mentioned this news articles and perform sentiment analysis on those. Segregate them on the basis of positive and negative sentiments.

Motivation:

Internet is huge source of information. And today the world is shifting from paper news to electronic news articles. With people's busy schedule, it is difficult to go through all these news articles. Summary of important point from all of these is what most user would like to read. Another important reason is that many of the news websites are majorly focusing on advertisements for commercial purposes rather than giving succinct relevant information about the topic. They add a lot of redundant data on their pages so as to increase the spaces to add more advertisements. To remove such redundant data and getting the important gist of the topic, a news summarizer which will aggregate important information from multiple websites is a perfect solution.

The proliferation of social media in the recent past has provided end users a powerful platform to voice their opinions. Businesses (or similar entities) need to identify the polarity of these opinions in order to understand user orientation and thereby make smarter decisions. This gave us the motivation to further provide a sentimental analysis to the user on his/her search query.

This application will give user a compendium where he can find a summarized news as well as the opinions/sentiments on his search query.

Consider for example an application where are users are interested in political news. This application when given terms related to politics will give a summarized version of the current situation to them. Also such application is in the field of politics, where political entities need to understand public opinion and thus determine their campaigning strategy. Sentiment analysis on social media data has been seen by many as an effective tool to monitor user preferences and inclination.

Contributions:

For the news summarizer, we have used are first taking the search query from the user. This search query is passed to Google News and urllib2 library to open the links. Then using the BeautifulSoup library we are downloading the HTML page and converting to plain text format. We have set a limit on the number of articles which will be fetched from Google News. We then used Cosine similarity to find documents which are relevant to each other. These documents are then clustered using k-means algorithm. For summarization, we are using POS-tagging, TF_IDF and tex-rank algorithms have been combined and used.

We are passing the URLs obtained initially to Twitter API which starts the search for the tweets which have mentioned this news article. These tweets are then analysed for positive and negative sentiments.

To train the sentiment analyzer, we created a labeled dataset from streamed twitter data for few search terms. For the labeling we used 2 techniques : Manual labeling and Vader's Labeling technique. Model trained using manual labeling helped us correctly label the tweets sentinet for the person. This reduces the error of classifying the tweet as positive or negative for the person just because he/she is mentioned in the tweet.

Later we have used Vader's Labeling for labeling the general search terms. For classifying, after trying various classifiers, we found the SVM-linear classifier gave the best result.

Outcomes:

Our application will work for any kind of search (space separated allowed) queries. Even if the search query has wide scope of possibilities for being mentioned in the news, our application segregates these articles and summarizes them accordingly. So this application even works for wide scope queries.

II. DESCRIPTION

Our application has 2 major sub parts:

1. Summarization
2. Sentimental Analysis

Summarization

For any search query entered by the user, 15 news articles are fetched from Google News. The number of articles can be changed. These fetched articles are converted to plain text from HTML using BeautifulSoup Library. These articles itself can be on various subtopics of the main query. There are 2 major challenges we faced here. First,Combining them to get a single summarized document may not always make sense. Consider for example, if the search query was "Donald trump", then if one article is about his current election status and the other is his about his personal life then summarizing these documents together doesn't make much sense. Secondly, our goal is give maximum information to the user and avoid any redundancy. This can be achieved if we remove the redundancy between similar documents in our summary. For example, if Donald Trump is in the news for his recent comment on Russia, it is highly likely that all the articles given by Google News will be about Russia only. Performing extractive summary on each article will result in redundant information being imparted to the user. In order to avoid this redundancy and incorrect merging of articles, we use Cosine- Similarity to find similarity between two documents and then cluster them to generate summaries that provide maximum information to the user.

1. Cosine Similarity:

We are using TF-IDF matrix to generate cosine similarity. Initially, to get the TF-IDF matrix, count the occurrences of the word document by document. A document-term matrix is formed which can also be called as term frequency matrix.

Words which frequently occur within a document but not within the corpus receive more weighting than other as these words are assumed to have more relevance to the document. We have also used ngram_range: Unigrams and bigrams.

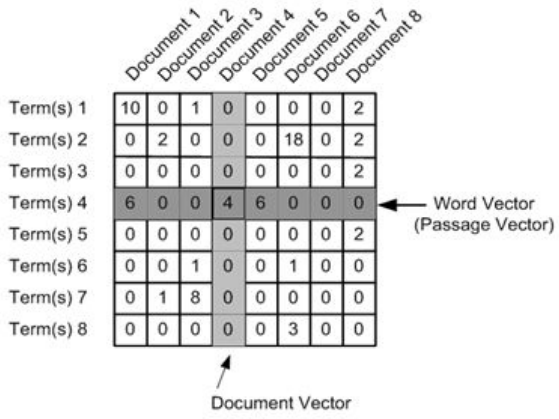

Fig 1: TF-IDF Matrix example

TF-IDF matrix is further used to measure cosine-similarity. This gives the similarity between each document and the other documents in the corpus. We have also removed English stop-words and used snowball stemmer for stemming and breaking down the word to its root.

Next, using this cosine similarity matrix, we cluster the documents which have high similarity indicating that these news articles are about similar topics. We use K-means algorithm for clustering. We have given the number of clusters to be 4.

2. K-means Clustering:

K-means takes a predetermined number of clusters. Each document is assigned to a cluster such that there is little variation within a cluster. Next, the mean of the clustered observations is calculated and used as the new cluster centroid. Then, observations are reassigned to clusters and centroids recalculated in an iterative process until the algorithm is stable and no more changes to cluster occur.
The output of the K-means algorithm are 4 clusters of similar documents. Thus from each cluster, we can generate a single summary, to avoid any redundancy in the summary.

We can divide the process of summarization in 3 steps. Initially to create intermediate representation of the sentences, scoring the sentences and then selecting important ones to create a summary.

We are using POS-Tagging to convert the text into intermediate representation which can be interpreted as the topics discussed in the sentence. We have initially used POS-Tagging to find important tags in the sentences. We used TF-IDF to find the weightage of the words. Here, words weighing highly indicate the topics.

3. TF*IDF weighting
(Term Frequency * Inverse Document Frequency)

We can eliminate the stop words from consideration because they occur frequently and can affect the word probability. Instead of deciding which words to be considered. This weighting exploits counts from a background corpus, which is a large collection of documents, normally from the same genre as the document that is to be summarized; the background corpus serves as indication of how often a word may be expected to appear in an arbitrary text. The only additional information besides the term frequency c(w) that we need in order to compute the weight of a word w which appears c(w) times in the input for summarization is the number of documents, d(w), in a background corpus of D documents that contain the word. This gives us the inverse document frequency:

$$TF * IDF(w) = c(w) * log\left(\frac{D}{d(w)}\right)$$

In many cases c(w) is divided by the maximum number of occurrences of any word in the document, which normalizes for document length. Descriptive topic words are those that appear often in a document, but are not very common in other documents. Words that appear in most documents will have an IDF close to zero. The TF*IDF weights of words are good indicators of importance, and they are easy and fast to compute.

4. POS Tagging

Though TF*IDF weights are good indicators of importance, but they can often lead to words which are used quite too frequently in the corpus, but do not represent the real meaning of the corpus.

Here, we use the POS tags we generate for every sentence. We then use the highest scored words given by our tf*idf weights, and search for them in our topics for every sentence, if the topic exists then we add that sentence to our summary, else discard it, thus ensuring that only the words which describe the topic of a sentence are contributing our summary. Depending on the sentences which contain words having more important topics representation. The score assigned to the sentences depend on how well a sentence expresses some of the most important topics in the document or to what extent it combines information about different topics.

5. TextRank

The sentences we shortlisted from the above pipeline represent the sentences that contain the maximum information about a particular cluster. We now generate a cosine similarity matrix for these sentences and using the similarity score from the matrix, we generate a graph using the Networkx library. We then perform PageRank on the graph, resulting in a score for every sentence. The score on the sentences signify how similar is a sentence to the other sentences. However, we need sentences which are not similar and thus we choose the lowest scoring sentences, ensuring that these sentences are not similar to each other and do not represent redundant information. We can arbitrarily decide the size of the summary. For now, we are using 30% of the total size of the selected sentences, thus generating a short summary.

Sentimental Analysis

The data for sentimental analysis will be collected dynamically from the last 2 weeks of tweets which have mentioned the news articles. We used Twitter Streaming API to fetch relevant data. The Streaming APIs give developers low latency access to Twitter's global stream of Tweet data. The input parameters to the streaming functions were the links to news article. Tweets corresponding to the given parameters were returned in JSON format. The JSON result basically comprised of key-value pairs. Some keys were created at, id, re-tweeted, screen name, location etc. The json responses were culled to extract only the body of the tweet and stored in a csv file.

This data was pre-processed special characters like '@'. Additionally, in the Machine Learning modules, to improve the classifier accuracy, we employ the tf-idf (term frequency – inverse document frequency) technique, to identify terms which are more relevant to sentiments.

We have 2 staged framework for labelling data. This two stage framework helps in achieving a data set that is both contextual and not sparse at the same time which aligns with the requirements of a supervised learning algorithm.

Stage 1: Manual Labeling using hashtag clustering:
Many a times it's possible that even though a person is mentioned in the tweet, the tweet might not be necessarily for him/her. The first stage of this framework comprises of manually labelling the Twitter data. However, the entire Twitter data set need not be labeled manually. We introduce a technique called hashtag clustering. Often while mining data from Twitter, users can find multiple tweets consisting of the same hashtag. For example, when Donald Trump is mentioned in positive sense by his supporter then the hashtag #MakeAmericaGreatAgain will be there. So tweets with hashtag #MakeAmericaGreatAgain can be considered positively for trump. So just by associating a label with a hashtag, thousands of tweets consisting of the same hashtag can be automatically labeled via a code. However before using this technique, it is necessary to sort the hashtags in their decreasing order of frequency. This will make sure that higher frequency hashtags get labeled prior to lower frequency hashtags. Depending on the application, developers or analysts can even choose to not label lower frequency hashtags or hashtags which are ambiguous unlike #MakeAmericaGreatAgain since they will be handled in our next section.
The emphasis on manual labeling is bolstered by the fact that a candidate maybe linked to a scam or an initiative. In the case of Benghazi controversy, which is negatively linked to Hillary Clinton, tweets consisting of #Benghazi cannot be labeled negatively for Hillary by using a cross domain data set. Such

contextual data pertaining to scams, controversies or political initiatives, which is a quite common in politics, can be handled only by human intervention.
As a proof of concept we have implemented this part just for 'Donald Trump' and 'Hillary Clinton'.

Stage 2 : Using Vader to label remaining tweets: Vader
(Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. It is basically a sentiment intensity polarizer developed by Hutto and Gilbert. Vader takes a sentence as input and provides a percent value for three categories - positive, neutral and negative and compound(overall polarity of the sentence).

Sentence	comp	pos	neg	neu
He is smart, handsome, and funny	0.8316	0.75	0.0	0.254
A really bad, horrible book.	-0.82	0.0	0.791	0.20
It kinda sux! But I'll get by, lol	0.22	0.274	0.195	0.53

Table 1 : Vader Sentiment Analysis Output

The above table provides three examples of sentences analyzed using Vader. The first sentence is highly positive, second highly negative and third one neutral. For performing sentiment analysis, a training data set should consist of sentences that are unambiguously either positive or negative. Hence, based on our observations, only those sentences having compound value >=0.8(highly positive) or compound value <=-0.8(highly negative) should be included in the training data set and the remaining sentences can be discarded. Python implementation of Vader is readily available on GitHub as an open source project.

Thus, the two stage framework proposed above can be used to create a training data set for Twitter. Based on the nature of application or the degree of fault permissible, the above two stages can be altered. For instance, the threshold of 0.8 in stage 2 can be increased or decreased depending on the needs of the user. Similarly, in cases where the frequency of sentences for individual hashtags is extremely less (like 10 or 20 sentences) then stage 1 can be directly eliminated and all sentences can be labeled using Vader.

We have given 25% weightage to Manual labeling and 75% to Vader Labeling mechanism. The sentient ratio can be between -1 to +1. We only classify the tweet to positive or negative if its ratio is above +0.5 or below -0.5 respectively.

To implement the supervised classification model design, we compared the performance of the following classifiers popularly used for text - based classification, on our preprocessed labeled data set:

Classification Technique	F1-Score
SVM - linear kernel	0.97
SVM - rbf kernel	0.39
SVM - liblinear	0.97
Multinomial NaiveBayes	0.94

Table 2: F1 scores for classifiers
Based on the F1- scores we decided to use SVM linear kernel for our sentiment classification.

III. EVALUATION

Summarization

We tried different techniques for summarization. Our initial attempt was to to use all the 15 articles to generate a single summary. We used textrank on all the sentences from all the articles, however, this resulted in a mix of duplicate and irrelevant sentences. The reason for this is -
1. The articles are about completely different topics, resulting in the sentences that do not seem relevant.
2. The articles are extremely similar, resulting in the sentences that impart redundant information.

For example - Consider, articles in the below order
1. Donald Trump wins election
2. Donald Trump fights with his wife
3. Donald Trump wins 100 seats in the election

The resulting summary would contain the most important sentence from the all the 3 articles in the same order, which does not make sense, as article 1 and 2 are completely irrelevant and article 1 and 3 are extremely similar resulting in sentences in the same order.

In order to overcome this challenge, we clustered similar documents using k-means and generated a summary for every cluster thus avoiding the irrelevancy from article 1 and 2. In order to obtain disjoint sets of sentences from the same cluster we used POS tagging and got the main topics of every sentence and used disjoint topics to generate the summary for that cluster.

This summarizer gives improved results as compared to our initial summarizer. We do not have human generated summaries across multiple documents and thus cannot compare them with human generated summaries to test our accuracy. However, we have verified the summaries manually, going through the clusters of documents and checking them with summaries generated by other summarizers available online and we get similar results.

Our clustering algorithm clustered the below links together, which are essentially about stony brook basketball games.
http://www.newsday.com/sports/college/stony-brook/lucas-woodhouse-helps-stony-brook-hold-on-against-northeastern-1.12703390,
http://www.newsday.com/sports/college/college-basketball/hofstra-routes-stony-brook-in-battle-of-long-island-1.12753809,
http://www.sbstatesman.com/2016/12/04/stony-brook-earns-first-home-win-against-northeastern/,
https://www.sbstatesman.com/2016/12/18/woodhouse-dominates-as-stony-brook-wins-at-saint-francis/
http://thehofstrachronicle.com/scouting-report-hofstra-preps-stony-brook-li-rivalry/'
Our summarizer generated a summary of the articles for the above links. This would reduce the redundancy across these articles reduce by a huge extent.

Sentiment Analysis

The following screenshot show the success of our sentimental analysis as it has efficiently separated positive and negative tweets for him using our trained model using manual labelling. We have observed that for "Donald Trump" and " Hillory Clinton" the success rate for classification of tweets using manually trained classifier was higher than that of Vader Labelling.

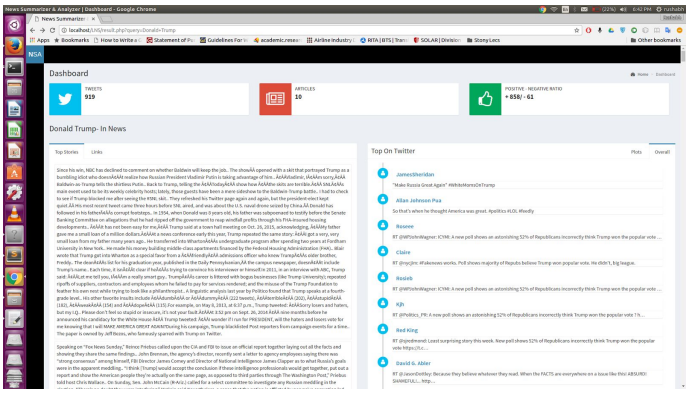

Fig 2 : Positive Tweets

But our analysis fails in detecting sarcasm.

IV. CONCLUSION AND FUTURE SCOPE

In this application we have tried various techniques to improve the summarization of news articles fetched from Google news. We used clustering algorithms to cluster news articles which are similar before summarizing them. Also, in sentimental analysis we have tried to introduce a new way to train the model to label tweets using manually labeled tweets. This improved the classification of tweets to negative and positive tweets. After trying various classification algorithms we found Linear SVM worked the best.
During this project we learned about ways to find similarity between documents, clustering algorithms, ways in which a document size can be reduced to remove redundant data. Also, how training the model to label tweets can help in efficient classification.

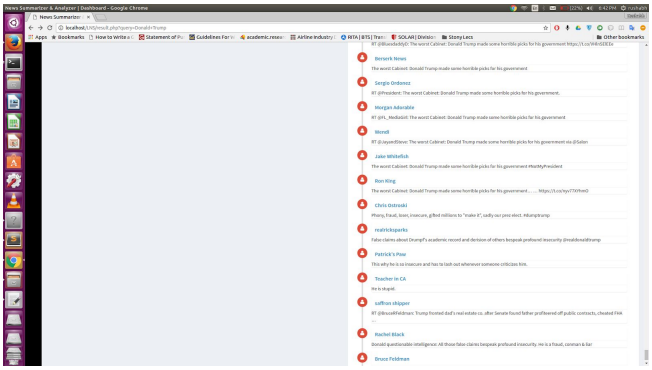

Fig 3: Negative Tweets

As a future scope for this application, summarization can be improved by abstractive summary rather than extractive. Also, currently the dataset we are getting from twitter is quite redundant because it is retweeted many times and it is fetched again. As a solution to this we can apply similarity algorithm on tweets and if similarity is more than 0.95, then we can ignore those tweets. Another possible work can be that the process of manually labeling can be done for a wider range of generic news topics.

REFERENCES

[1] NLTK www.nltk.org/
[2] Vader https://github.com/cjhutto/vaderSentiment
[3]POSTagger
http://thetokenizer.com/2013/05/09/efficient-way-to-extract-the-main-topics-of-a-sentence/
[4]"The Streaming APIs — Twitter Developers", Dev.twitter.com,https://dev.twitter.com/streaming/overview.
[5] Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14).
[6] "A Survey on Automatic Text Summarization" https://www.cs.cmu.edu/~afm/Home_files/Das_Martins_survey_summarization.pdf
[7]https://thetokenizer.com/2013/04/28/build-your-own-summary-tool/
[8]https://thetokenizer.com/2013/05/09/efficient-way-to-extract-the-main-topics-of-a-sentence/
[9] "k-means clustering" :http://brandonrose.org/clustering
[10] "A survey of Text summarization Techniques" "https://www.cs.bgu.ac.il/~elhadad/nlp16/nenkova-mckeown.pdf