Sentiment Analysis of Web News Articles

Gadde Venkata Sai Kumar
Department of Computer Science
University of Massachusetts,
Amherst, MA 01002
gvenkatasaik@cs.umass.edu

Anirudha Desai
Department of Computer Science
University of Massachusetts
Amherst, MA 01002
anirudhadesa@cs.umass.edu

Abstract

Search engines have surpassed the usage of news papers in providing essential news. Search engines can retrieve information from diverse sources. It is not feasible for an user to read a multitude of articles to get holistic information about a topic. Our end objective is to provide an in-house web-interface platform (refer appendix) for an end user to gauge the news articles along with the sentiment behind the corresponding article.

1 Introduction

Day by day, users are reliant on the web to get their daily news. News papers have a standard audience and users expect news in a certain way from the news papers. But web is a completely different platform where there is no ordering of articles. The most commonly known algorithm is page ranking algorithm by Google. We do not know the intention of the article even at the abstract level when reading news in the web. Therefore, our prime motivation in this project is to classify articles based on the sentiment as a first step to provide more information to the users who search web for news.

Companies have been using sentiment analysis to understand the impact of their products. Governments have been using sentiment analysis to understand the pulse of people. So, as an extension to this idea, we have thought whether this kind of data analysis is directly available for the consumers during search. This can help them to be cognizant of the other side of the coin. So, our project is about using sentiment analysis to differentiate between the articles whether they are positive,neutral or negative. Our primary focus is on search engines which are used frequently by online users.

We observe that extensive research is conducted with sentiment analysis of texts with high subjectivity such as movies or product reviews. But with news articles, we found limited research in comparison to that from customer reviews, twitter and other social media. We focused on building an end to end solution for this problem. Starting with data collection to cleaning and to modelling machine learning algorithms and then using it in our practical application. We built a custom search engine that sources its results from Bing and implements the model on the search results.

2 Related Work

This section briefly describes previous works in the field of sentiment analysis with focus on, in some cases, news.

In [1], the authors applied machine learning techniques (Naive Bayes, Maximum Entropy Classification and support vector machines) for sentiment analysis on movie reviews data set. The authors cite that machine learning techniques for sentiment analysis outperform human baselines with counting methods. This paper also highlights some of the key differences between topic based classification and sentiment analysis with machine learning perspective. The paper also highlights the effectiveness of applying machine learning techniques to a sentiment classification problem. The key challenge in this domain being the subtlety with which sentiments are embedded and is therefore a different and difficult problem in comparison to topic based classification.

News and Blogs are a powerful resource and sentiments are usually either positive or negative. In [2], the authors highlight interesting aspects drawn from analysis on news and blogs. They segregated news into seven sentiment dimensions(general, health, crime etc.) and a separate lexicon was defined for each of these. The focus is particularly on news entities such as celebrities and identifies them, based on the sentiment polarity scores, as either positive or negative people in a given time span.The paper also discusses interesting metrics such as polarity and subjectivity. Polarity is defined to be whether a sentiment associated with an entity is positive or negative. On the other hand, subjectivity is defined the amount of sentiment that an entity garners.

In [3], the authors bring in a novel way of sentiment analysis in news. The authors extract quotes from news articles as they are more subjective than news articles in general. In addition to the quotes, other entities such as speaker and the target of the quote are also extracted. This paper motivated us to extract news articles from the web to create a data-set as per our requirement and apply machine learning techniques for sentiment prediction. The authors emphasize the need for clearly defining the source and the target of a particular quote for effective opinion mining. Further, the authors emphasize that every newspaper articles has three different possible views - author, reader and the text. Each of these need to be addressed differently at the time of analysing sentiment.

In [4], the authors discuss another interesting hybrid approach for sentiment analysis of blogs where domain specific lexical knowledge is combined with Text Classification. The authors use the Naive Bayes classifier, after applying the background lexical information for features, for the classification task. This paper reinforced our approach of applying machine learning techniques for sentiment analysis.

3 Methodology

In this section, we describe the process through which we developed our project. Broadly, it consists of Data Extraction and Preprocessing, Feature Extraction, training Machine Learning model and testing the model.

3.1 Data Extraction :

Sentiment analysis has been predominantly performed in classifying consumer reviews/comments as sentiment is relatively clearly evident. There are publicly available curated data-sets for classifying movie reviews, tweets and other consumer reviews etc. But when it comes to news articles, we faced challenges in obtaining a dataset for our requirement. For the purpose of learning, we refrained from using the highly curated data-set "Brown corpus" available in the "NLTK-Corpus". Instead, we decided to create our own classification data-set by scraping the web for news articles. Section 4 elaborately describes the steps we incorporated.

3.2 Data Preprocessing:

Once we obtained the article sentiments from stanfordCoreNLP, we proceeded towards feature extraction. We explored numerous methods to extract features from the article text data. Of these, we implemented *Bag-of-Words* and *Term Frequency-Inverse Document Frequency(TF-IDF)* vectorizer methods for our project.

Bag-of-words is a simplifying representation that represents a text document as a multi-set of its words and the feature vector for each article maintains the multiplicity of these words. Each feature is the individual token occurrence frequency and each sample is the vector of all token frequencies of the article.

Tf-Idf assigns weights to words based on the frequency of occurrence and builds the feature vector. Mathematically, it can be represented as

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

where, tf(t,d) is the **term frequency of the document** and idf(t) is the **inverse document frequency** calculated as

$$idf(t) = log\frac{1 + n_d}{1 + df(d, t)} + 1$$

where n_d is the total number of documents, and $df(d, t)$ is the number of documents that contain term t.

3.3 Model Training:

We trained our models with Support Vector Machines and Random Forest Classifier. The decision to choose these classifiers is largely influenced by our literature review. We found that these models have been repeatedly used for sentiment classification tasks in the past and is known to yield favorable results. For instance, in [5], the authors apply Support Vector Machines over a diverse set of information sources. In [6], Pang et. al. achieved favorable results with Naive Bayes and Support Vector Machine classifiers.

Support Vector Machines A Support Vector Machine is a discriminative classifier defined by a separating hyperplane. In this algorithm, each data item is plotted in a n-dimensional space (n -> number of features), where value of each feature is the coordinate value. The classification is then performed by finding a hyperplane that differentiates the classes. SVC is known to work well in high dimensional spaces. Support Vector Machines is used because of the high dimensionality of our dataset.

The decision function of Support Vector Classifier has the form :

$$f_{SVM}(\mathbf{x}) = sign(\mathbf{w}^T(\mathbf{x}) + b)$$

In case of SVMs with l_2 regularization, the model parameters are selected by minimizing the function:

$$C\sum_{i=1}^{n} L_h(y_i, g(x_i)) + ||w||_2{}^2$$

where the function $L_h(y_i, g(x_i))$ is the *Hinge Loss*

Random Forest Classifier RFC takes sub-samples of dataset, fits several decision trees to it and uses averaging to get the highest prediction accuracy. There is no simple mathematical representation for an RFC. It is an ensemble classifier that works by building multiple decision trees and then aggregating them. The hyper-parameter for this classifier is the number of total trees/estimators. Random forest classifier is chosen because it can handle thousands of input variables without variable elimination. It can be used when the dataset is skewed wherein the class distribution is uneven. Random Forest classifiers do not overfit as there are multiple trees which fit the data.

Metric for Test Accuracy: The number of samples in the negative class in our dataset, as discussed in further sections, is higher than that of the positive and neutral class. Thus, the identity function alone does not provide a good metric for measuring the performance of the model. Therefore, we employ a more stringent performance measure where we report the false positives and false negatives alongside the accuracy. False positives indicate the percentage of positive labels in test data that is incorrectly predicted by the model and like-wise for other classes. Mathematically,

$$Classification\ Accuracy\ Rate = \frac{1}{N}\sum_{i=1}^{N} I(y_i \neq f(x_i))$$

$$False\ Positive = \frac{Number\ of\ incorrectly\ predicted\ Positive\ Labels}{Total\ Positive\ Labels}$$

Similarly, we calculate the false negatives and false neutrals.

Cross Validation: We performed k-fold cross validation (k=3). In cases of text classification, with feature vectors as Bag of words or TF-IDF frequencies, the vocabulary of the feature vector is dependent on the training dataset. As we use K-fold Cross Validation, the feature vector vocabulary changes when the training samples are changed. Therefore K-fold cross validation results in an averaged accuracy. So, for deciding the optimal hyperparameters, it is important that the model is trained with cross validation so that the predominant features are given due weightage.

3.4 Technology Stack

In this section, we list the programming languages, development environments, code libraries, and other resources that we used for the end-to-end development of our project.

Machine Learning: Spyder IDE, Python, Sklearn, Pandas, Numpy, matplotlib.
Data Collection:Python, Beautiful Soup, Scrapy (Web scraping framework), Pandas, Javascript(REST APIs for extracting NYTimes articles).
Website Development: HTML5, Bootstrap, CSS, Flask Server, Bing APIs. We developed a website that provides a search engine interface (based on Bing API) interface for the end user. When a search is fired, the sentiments of the articles returned by the search engine are predicted at runtime using our trained model. For easier visualisation, positive sentiment articles are color coded to blue, neutral articles are color coded to green and the negative sentiment articles to red. With this, the end user is aware of the sentiment of an article before opening the contents of the page. The user also gets a clear picture of the general sentiment towards the topic. Currently, the website runs on local server.

4 Data Set(s)

For data sets, we explored the web for resources providing sentiments extracted from news articles. Since, most of the data-sets for sentiment analysis was for texts with high subjectivity, we created our own dataset by extracting news articles from the web. The method we followed is listed here:

1. *Extract Links to Articles :* With the help of NYT REST API, we collected a diverse set of 14000 URLS belonging to fashion, technology, World,politics, sports, business, books, art, automobiles, style,television etc. from New York Times.

2. *Web Scraping:* We programmed a python script to crawl through the 14000 URLs collected above. We used scrapy module from python for this purpose. We limited ourselves to the first 1000 characters of the news article with the objective that news articles are usually summarized initially before delving into further details.

3. *Cleaning data:* Using the BeautifulSoup module from python, we cleaned the data further such as removal of html tags, next line characters etc. Also, we observed that some of the articles that were scraped contained noise. We eliminated such articles from our collection.

4. *Calculate polarity using Stanford CoreNLP:* We established a stanford coreNLP server connection and wrote a python script to extract polarity of each of our news articles. Stanford coreNLP provides the polarity scores sentence-by-sentence for an article. We deployed two approaches here. First, we averaged the polarities of all the sentences in an article to obtain the generic sentiment of the corresponding article. Second, we trained the model with the sentence-polarity dataset and then applied the model to the complete article. Both the approaches are discussed further in Section 5.

5. *Feature Extraction:* We explored numerous methods to extract features from the text data. Of these, we discussed *Bag-of-Words* and *Term Frequency-Inverse Document Frequency(TFIDF)* vectorizer methods in this report. Words such as "kill", "killed", "kills" express the same sentiment, so we employed a stemmer that reduces all these words to the "root" words using the Porter Stemmer from NLTK library. Both Unigrams and Bigrams are used for Tf-Idf vectors as Bigrams include semantics.

Number of Data Cases: For models based on article level sentiment, we used 5468 data cases for training and 2320 data cases for testing. For models based on sentence level sentiment, we used 28959 data cases for training and 12738 data cases for testing.

Features: We trained our models over 5000 features. The features consist of a distribution between single (Unigram) and pairs of words (Bigrams) such as "United States" that frequently occur together. These features represent the most frequently appearing words (or pairs of words) from the training data set.

5 Experiments and Results

This section is about our experiments on the data-sets and models and results of multiple models implemented to improve the accuracy, false-positive, false-negative and false-neutral metrics. Sentiment classification is done to classify every article into three sentiment classes neutral, positive and negative. The first step in our experiments included the data auditing of the training data-set and the test data-set. Second step is about the feature engineering and model implementation on the data-set.

The distribution of classes in the train dataset and the test dataset is shown in Figure 1. We observe that the extracted dataset is skewed more towards the "Negative" data class. So, in addition to training our model on this dataset, we trained the model by balancing the the samples to overcome the skewness in the data. Further, as discussed in section 4, we employed a training model with sentences-only approach. We discuss each of these experiments in detail.

5.1 Document level sentiments

Histograms are drawn for the sentiment class labels for the training data-set and the test data-set. As shown in Figure 1, data-set is skewed towards the negative class. There are more number of samples in the negative class than positive and neutral class.

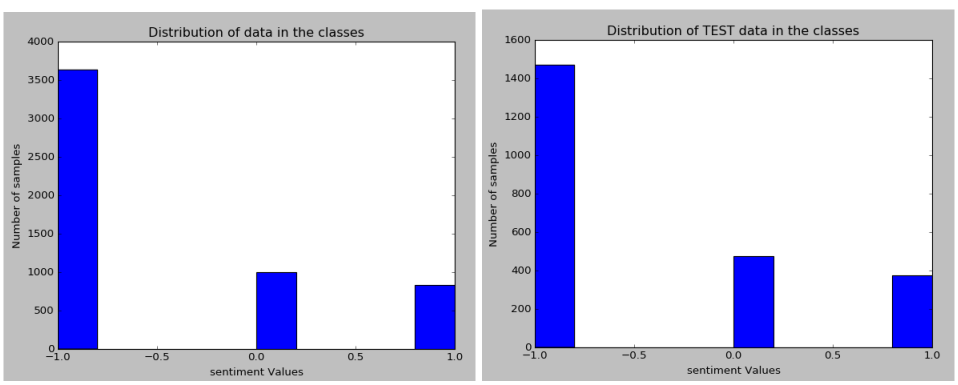

Figure 1: Distribution of data between classes for the Train dataset (*left*) and the Test dataset (*right*). As explained above, the dataset is skewed towards the negative class.

5.1.1 Classification without class weights

*Model Implementation:*We implemented Random Forest classifier and Support Vector Machines classifier and the results are shown in Table 1. This model is used as the base line model for further results. With the data distribution that we have for test data, predicting all negatives yields an accuracy of 60%. We observe that the accuracy value is better than predicting all the articles as negative. But the false positive, false neutral are very high which is evident since the data-set has more number of negative labelled values. So, in order to improve the performance of the model, we have assigned weights to the samples belonging to the particular class labels. This is discussed in section 5.1.2.

Table 1: Model Results

Classifier	Accuracy	False Positives	False Negatives	False Neutral
Random Forest Classifier	62.84%	0.9547	0.0265	0.9726
Support Vector Machines	63.27%	0.811	0.09	0.8736

5.1.2 Classification of articles with class weights

As observed in section 5.1.1, the negative class samples had a huge impact on the final results. So, the samples belonging to different classes are balanced using the class_weight model parameter. By default, all the classes are assumed to have unit weight. The samples are then balanced by adjusting the weights inversely proportional to class frequencies in the input data as:

$$ClassWeight = \frac{Number\ of\ Samples\ in\ Class}{(Number\ of\ Classes * np.bincount(sentiment\ labels))}$$

Results are tabulated in Table 2. Support vector machines classifier performed better than the Random forest classifier. Although we do not see an improvement in the accuracy of the model, we do notice a decrease in the false-positive and false-neutral metrics. But these results too were not on par with our expectations as 73.9% of samples to be sampled positive are predicted wrong and the same is 81.68% for neutral samples.

Table 2: Model Results with sampling Techniques

Classifier	Accuracy	False Negatives	False Positives	False Neutral
Random Forest Classifier	61.896%	0.068	0.925	0.92
Support Vector Machines	62.5%	0.138	0.739	0.8168

At this point, we observed that training models based on a document level sentiment might be inaccurate as a document can contain a combination of negative and neutral/factual statements. This observation led us to our next experiment wherein we created a data-set comprising of sentiment values of individual sentences from our articles rather than the complete article for training our model.

5.2 Sentence level sentiments

As concluded in section 5.1, we repeated the extraction process to create our train and test datasets, but this time with sentences instead of articles.

5.2.1 Classification of sentences without class weights

In this approach we obtained sentiment values from stanfordCoreNLP for each sentence in the data-set. We repeated the procedure in section 5.1.1 to understand the base line performances of the Support Vector Machine and Random Forest Classifiers when the data-set comprises only sentences. The values are tabulated in Table 3. The distribution of samples in various classes remains the same. But the accuracy observed has improved significantly to 81.66% with RFC.

Table 3: Model Results with considering sentences for the classification task without class-weight

Classifier	Accuracy	False Negatives	False Positives	False Neutral
Random Forest Classifier	81.66%	0.0649	0.908	0.595
Support Vector Machines	81.75%	0.032	0.825	0.833

From these observations, the classifier with sentences dataset performs better than the articles data set. As we observed improvement in performance of the classifier when class-weights are given in section 5.1.2, we tried a similar approach here which is discussed in section 5.2.2.

5.2.2 Classification of sentences with class weights

As described in section 5.1.2, class_weight is used as a model parameter to balance the classes belonging to various classes. This experiment resulted in better results in terms of false positives and false neutrals. The results are tabulated in Table 4. We observe that, there is an improvement in false positives and false neutrals as compared to that with articles discussed in section 5.1. We propose this model to perform relatively better than the previous model.

Table 4: Model Results considering sentences for the classification task with balanced sampling

Classifier	Accuracy	False Negatives	False Positives	False Neutral
Random Forest Classifier	81.69%	0.073	0.8896	0.54
Support Vector Machines	80.1%	0.086	0.73	0.65

5.2.3 In this case, we omitted few samples from the data-set which had negative sentiment labels such that we get a uniform distribution between the three class labels. The procedure explained in prior sections is repeated again and the results are tabulated in Table 5. We see that this resulted in under performance of the classifier as, from our observations, decreasing the number of samples reduced the feature vocabulary thereby resulting in lower accuracy.

Table 5: Model Results with manually balancing the negative, neutral and positive data cases

Classifier	Accuracy	False Negatives	False Positives	False Neutral
Random Forest Classifier	65.31%	0.3255	0.7606	0.2916
Support Vector Machines	63.6%	0.336	0.642	0.4082

5.3 Hyperparameter Optimization

Both the classifiers Random Forest Classifier and Support Vector Machines, performed equivalently well in our experiments. Therefore, we performed hyper-parameter optimization for both the classifiers.

Random Forest Classifier Hyperparameters considered for Random Forest Classifiers are n_estimators (number of trees in the forest) and minimum sample split. Figure 2 plots the variation of classifier accuracy against each of these values. K-fold cross-validation (k=3) is used in the hyperparameter optimization. We obtained the highest accuracy of 82.1% with Number of Estimators equal to 40.

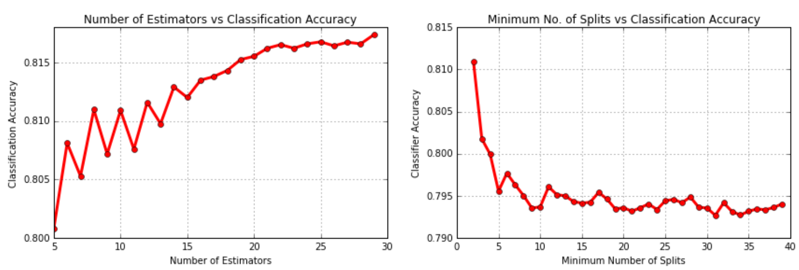

Figure 2: Classifier Accuracy vs Hyperparameter value for RFC (*left to right*) for n_estimators, minimum split

5.4 Additional Classification Model

This model is out of our curiosity to reduce the number of features for the modelling. From the data-set, we observed that the number of nouns, verbs, adjectives etc. do have influence on the sentiment of the article. So, we extracted the number of nouns, verbs and adjectives from each text in the data-set. Using the Sentiment Intensity Analyzer library from NLTK module, we extracted positive sentiment, negative sentiment and neutral sentiment for each word. As sentiment values of nouns doesn't influence the sentiment of the article, we have considered the positive, negative and neutral sentiments of the verbs and adjectives and used them in our feature vector. In summary, the final training dataset features is: [number of verbs; number of nouns; number of adjectives; adjectives_sentiment_positive, adjectives_sentiment_neutral, adjectives_sentiment_negative, verbs_sentiment_neutral, verbs_sentiment_positive, verbs_sentiment_negative] With such a few number of parameters we attained an accuracy close to our earlier tf-idf model.

Table 6: Model results with Additional classification model.

Classifier	Accuracy	False Negatives	False Positives	False Neutral
Support Vector Machines	80.7%	0.05	0.91	0.71
Random Forest Classifier	71.36%	0.22	0.81	0.47

We have done this as an experiment in feature engineering. We haven't used this for the final pickle file, as this model relies on the sentiment values given by the nltk and there is no enough literature on this method.

6 Discussion and Conclusions

We trained multiple models with multiple feature extraction techniques. From all our analysis of models, we concluded to use the Random Forest Classifier model trained using sentence level sentiments to classify the sentiments of the articles. The hyper-parameters obtained from the hyper-parameter optimization done in section 5.3. are used to generate pickle files of the classifier and the vectorizer.

We think that we identified a problem regarding the search engines. We started our project with a live problem statement and built our solution to solve this problem. We implemented an end to end pipeline in this project from data extraction to the model implementation in real life scenario. The challenges involved were significantly higher than the assignment problems where the dataset was clean and evenly distributed between the classes. And the problem has very large scope outside of news articles, as search engines give results from wide variety of sources.

We compared search results of two search queries "hate" and "celebrate". The results are clearly evident from the results shown in Figure 3, as the number of neutral sentiments are more when the search query was "celebrate". Almost all the articles are negative when the search query is "hate". Refer to appendix (after references) for scaled images of the same.

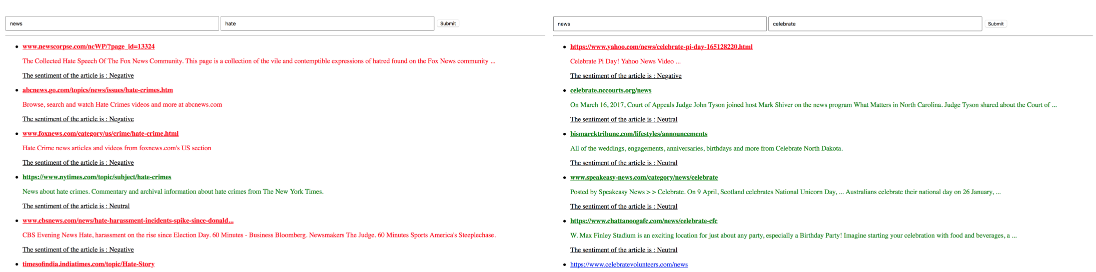

Figure 3: Website results with search term "hate" (*left*) and "celebrate" (*right*). We see that there are more negative articles with search term "hate". Green color implies neutral, Red implies Negative and blue implies Positive. These results can be regenerated by running the server.py file in the code submitted and open the url: http://0.0.0.0:8000 and make a search query.

Future Work: We see a lot of opportunity in this field where users are provided with more relevant information about the search results. We would like to continue our work to include more analysis into the search queries. In the future we intend to use deep learning to understand the dependencies between the words and extract the positive or the negative sentiment about the search query term rather than the whole article. One idea we are deeply interested is not the sentiment of the article but to notify the sentiment of the text which is related to the search query **?**.

References

[1] Bo Pang,Lillian Lee & Shivakumar Vaithyanathan *Thumbs Up? Sentiment Classification using Machine Learning Techniques*

[2]Namrata Godbole, Steven Skiena & Manjunath Srinivasaiah *Large-Scale Sentiment Analysis for News and Blogs*

[3] Alexandra Balahur, Ralf Steinberger, Mijail Kabadjov, Vanni Zavarella, Erik van der Goot, Matina Halkia, Bruno Pouliquen, & Jenya Belyaeva *Sentiment Analysis in the News* .

[4] Prem Melville , Wojciech Gryc & Richard D. Lawrence *Sentiment Analysis of Blogs by combining lexical knowledge with Text Classification*

[5] Tony Mullen & Nigel Collier *Sentiment analysis using support vector machines with diverse information sources*

[6]Bo Pang & Lillian Lee *A sentiment education: Sentiment Analysis using subjectivity summarization based on minimum cuts*

7 Appendix

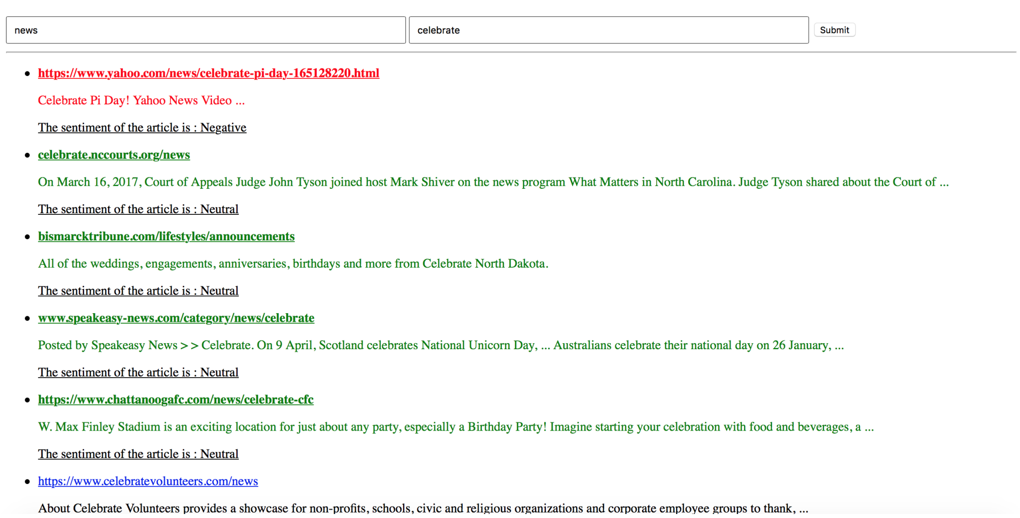

Figure 4: Website results with search term "celebrate". This returns lesser number of negative sentiment articles.

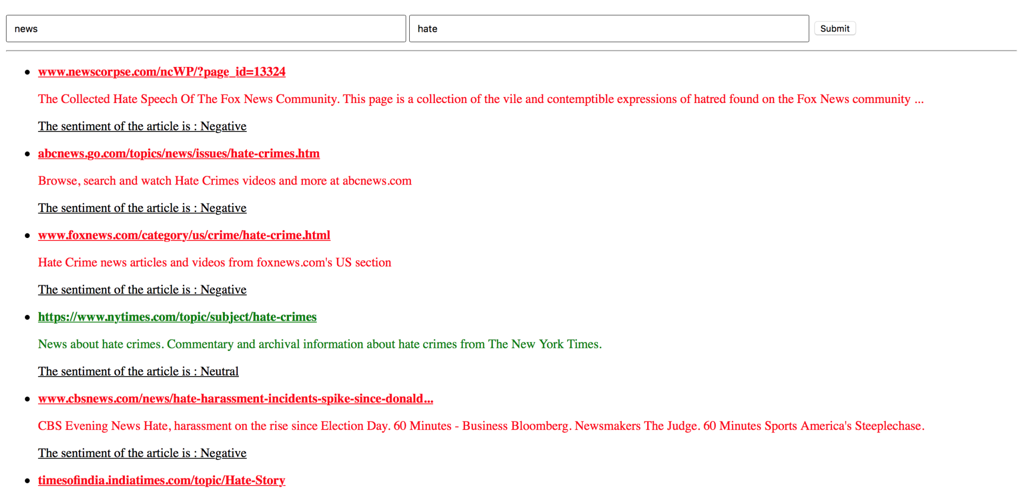

Figure 5: Website results with search term "celebrate". This returns lesser number of negative sentiment articles.