News Classification with NLP

Advait Sharma, Harsh Jain, Naman Ranawat, Siddhanth Satish

Abstract

News and media have become an integral part of people's views and opinions in our society. News reporting has developed so much in the 21st century that there's much more content out there on each news platform. This provides news platforms a new challenge of classifying the news in the right way. It is important to improve this classification to recommend the right news for the user. To do this, we need to uncover which algorithms work best for classification, which categories are being confused the most, and least to improve our classification task substantially. In this paper, we are testing our data on a robust array of classification algorithms like KNN, Logistic Regression, Naive Bayes Classifier, Random Forests Classifier, Support Vector Machines. Post classification, we are finding the answers to our research questions through the confusion matrix of our best-performing model.

Introduction

As the crux of this research paper is to predict the category of a news headline more accurately, it is important to analyze why exactly it's crucial. A survey of online users in the United States revealed that 49 percent of respondents felt that social media and messaging apps were among the three types of apps that they spent the most time on, meanwhile only 17 percent felt the same about news apps. This statistic shows that a larger percentage of people who wish to stay updated on current affairs go to social media to stay updated than a verified news application.

While it would be almost impossible for the news applications coverage of users to overtake that of social media apps due to the wide array of features, freedom of usage, and personal relevance offered by social media, it is important to increase this proportion. Having people go to social media to learn about news leads to a risky predicament as it could potentially lead to a vastly misinformed population.

The fundamental advantage social media applications have over news applications is better recommendation systems and classification of their data. As a result, we think the first step to improving the usage rate of news applications is to recommend more relevant news headlines to users.

This improvement in accuracy in this classification can result in more news viewing by users and solve the problem of presenting the user with unnecessary/inaccurate suggestions. This can further be implemented to merge all news of the same category and finally recommend users the piece of news they'd like to read.

This leads us to the goal of this project is to answer the following research questions we are trying to answer are:

(1) What is the category for a future headline?

(2) What news categories cover relatively similar news content?

(3) What news category covers the most unique content?

(4) Which models perform well in text classification?

From this research project if we run the algorithm on KNN, Logistic Regression, Naive Bayes Classifier, Random Forests Classifier, Support Vector Machines to find the models that perform best, which categories are most unique, and which ones are confused the most.

Related Work

The two closest research projects that are similar to our approach were 'News Headlines Classification Using Probabilistic Approach' and 'News Classification: A Data Mining Approach'.

'News Headlines Classification Using Probabilistic Approach' [1] proposes a probabilistic approach for classifying news headlines. The three components of pre-processing, model learning, and classification are applied in this approach. After pre-processing, this paper discusses generating a probabilistic model of news categories from the training data.

The frequency of words concerning headline categories is computed and then, tokens that are less domain-specific (ie. non-informative) are identified and added to the general stop-word list.

The probability matrix of words for headlines is then generated and used to classify the data which gives the learned model of probabilities. In the end, test data is sent through the same preprocessing pipeline with the new stop-word list and is then used to compute the probabilistic contribution of each word in various categories, and the category with the maximum cumulative probability is predicted as the class for the headlines.

'News Classification: A Data Mining Approach' [2] discusses the text classification by applying multiple algorithms: NaiveBayesMultinomial, J48, JRip, MultiClassClassifier (Logistic), SMO, HyperPipes. The text is converted to a word vector after it is cleaned up in a similar way to the previous research.

---

[1] Rana, Mazhar Iqbal, et al. "News Headlines Classification Using Probabilistic Approach." VAWKUM Transactions on Computer Sciences, vol. 7, no. 1, Sept. 2015. DOI.org (Crossref), https://doi.org/10.21015/vtcs.v7i1.341.

[2] Ramchandra Kawade, Dipak, and Kavita S. Oza. "News Classification: A Data Mining Approach." *Indian Journal of Science and Technology*, vol. 9, no. 46, Dec. 2016. *DOI.org (Crossref)*, https://doi.org/10.17485/ijst/2016/v9i46/84444.

The classification process consists of a training and a testing phase. The various algorithms are applied to the test vectors in the training phase and validated in the testing phase. NaiveBayesMultinomial turns out to be the best algorithm in accuracy and time efficiency.

We have taken a similar approach to the problem by using a labeled data set with headlines and their categories. After an eighty twenty train test split we are training various models on this labeled data of News Headlines and returning the confusion matrix. Our models are also equipped to predict the category of a new news headline after training.

Data

For training news classification algorithms label we are going to use the News Category Dataset[3]. The dataset contains 200853 news headlines from the year 2012 to 2018 obtained from HuffPost. We acquired this dataset from Kaggle, so the overall structure of the data is suitable for applying different supervised machine learning algorithms for text classification

Supervised machine learning algorithms lead us to insights on general patterns in the data and help us generate models from labeled data so we can generate new models based on them.

This dataset has the following attributes:

1. Category:
    a. 41 unique categories.
    b. Datatype: String (the category itself - crime, politics, sports, etc)
2. Headline:
    a. It is a sentence that summarises the news article.
    b. Datatype: String
3. Author:
    a. It contains the name of the news author.
    b. Datatype: String
4. Link:
    a. The URL of the post.
    b. Datatype: String
5. Short description:
    a. The description of the news article.
    b. Datatype: String
6. Date:
    a. The date of publication.
    b. Datatype: String

[3] Rishabh Misra. *News Category Dataset*. Unpublished, 2018. *DOI.org (Datacite)*, https://doi.org/10.13140/RG.2.2.20331.18729.

The Fig. (a) below is a pie chart that portrays the percentage of entries for each of the 41 categories in the dataset. From this pie chart, we can infer that 'Politics', 'Wellness' and 'Entertainment' are the most covered news categories on HuffPost. 'Education' is the least covered news category with an entry count close to 0% of the dataset. The inequivalent distribution of the news categories would make the models more biased towards categories with more representation in the dataset. But that bias should not negatively affect our models as we expect the future headlines to maintain a similar distribution of categories so the probability of a headline belonging to the 'Politics' category would be higher.

Distribution of Dataset Based on Categories

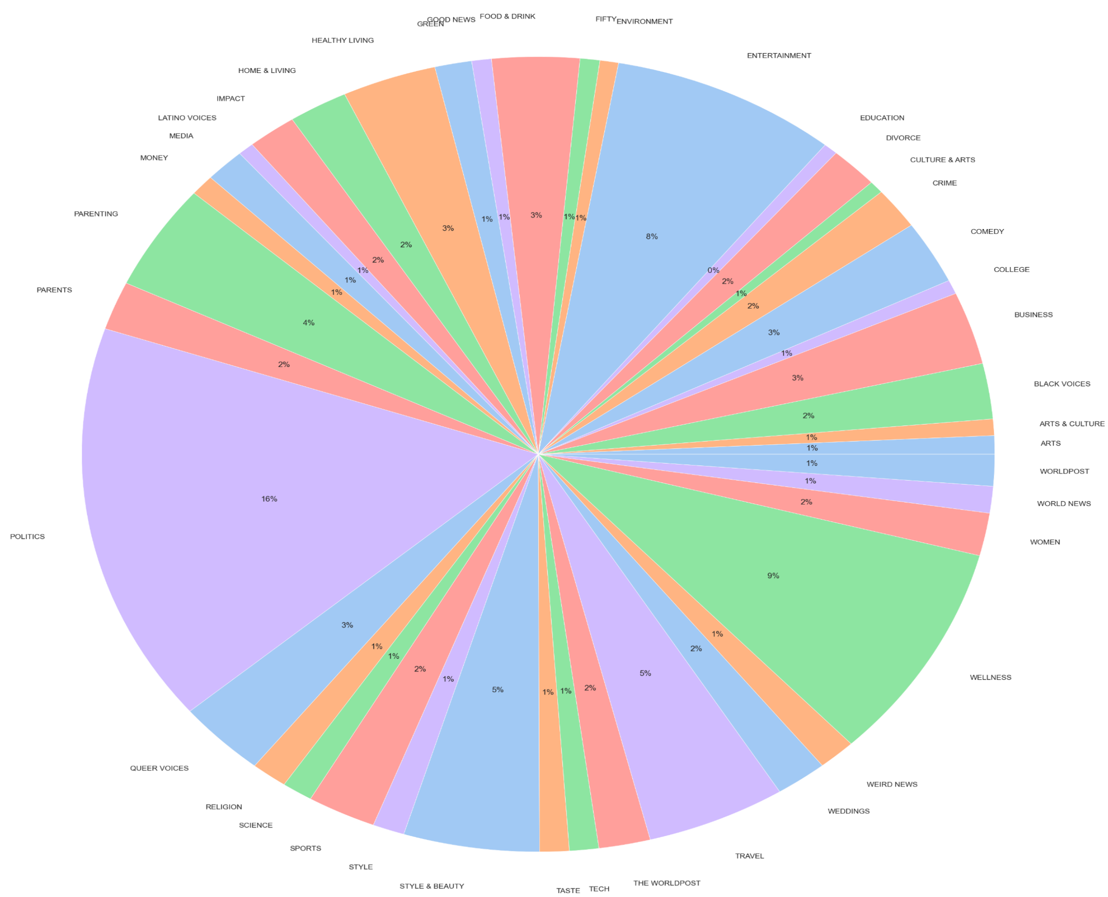

Fig. (a)

Methodology

Using a dataset from Kaggle we first converted the json file to a data frame.
Since we're dealing only with the headline column to predict the news category, we dropped all the other columns and just kept "category" and the "headline" column.
X = headline
Y = category
We did preprocessing on the above data as described below:

I. Tokenization

We tokenized the data in two different ways:

1. Basic tokenization on the headline where we removed punctuation and stop words because they do not contain a lot of useful information. We then normalize the data. It is important to normalize mainly to reduce randomness. By that we mean, words like connect, connection, connections, connects, etc get reduced to connect. This improves the efficiency of the system. To do this, we have used the snowball stemmer.
2. BERT base uncased model for tokenization. According to the hugging face documentation, it allows the model to learn the bidirectional representation of the sentence. This can be used for extracting important features.

II. Stop Word Removal

The stop words are language-specific and do not carry any information. It generally includes conjunctions, pronouns, and prepositions. They are considered of low worth and are removed eventually. These words need to be percolated before the processing of data. Stop words can be removed from data in many ways. We used the stop words from the NLTK corpus and passed them to the Count Vectorizer function to remove stop words from our data.

Removing stop words from the documents benefits the performance of the classifier in two major ways. First, it reduces the feature space for classifiers which in turn improves the speed of classifiers significantly. Secondly, as only the important words remain, the classifier would now be able to determine the weight distribution of these words more accurately, which also in turn improves the accuracy of the applied classifier.

III. Word Stemming

After the removal of stop words, we then performed stemming. This step aims to reduce each token to its root. The motive behind using stemming is to remove the prefixes and suffixes so that the number of words could be brought down and it would be an easier task for our model to perform. For example, the words like user, users, used, using all can be reduced to the word

"USE". This will reduce the required time and space. For stemming there exist many stemmers like Snowball Stemmer, Porter Stemmer, Paice/Husk Stemmer. Out of these, Snowball worked out best for us.

Then we simply passed this data to a count vectorizer to get the frequency of each word.
The output of the vectorizer is stored in the X variable and the categories are stored in the Y variable.

Then, we use an 80-20 train-test split to get the training and testing data.
We used four different models for training on the data described above.

I. Logistic regression

Logistic regression is a classification algorithm that predicts a binary outcome based on a series of independent variables. This would mean predicting whether you would pass or fail a class. Logistic regression can also be used to solve regression problems, but it's mainly used for classification problems. So, we decided to use Logistic Regression for this task as we would be classifying news into different categories.

II. SVM

SVM has been used a lot for news text classification. Specifically, because it has a unique feature that includes both negative and positive training sets, unlike other algorithms. It involves both dimension reduction and text classification. From experimental research, we found that SVM algorithms outperform Naive Bayes and most text classification algorithms because it can handle a high dimensional input space and at the same time sparse feature vectors.

III. Decision Trees

The decision tree is a classifier for text categorization represented in the form of a tree. Here, each node acts as a decision node. They are quite easily perceived. popular in use and rules can be easily produced through them. However, it comes with a clause that the training-decision tree is a computationally expensive task. Besides the algorithm is such that if one news can be connected to one branch only. If a mistake occurs at the higher upper level it can cause the whole subtree to go invalid.

IV. Naive Bayes

Naive Bayes is a probabilistic classifier based on text features. It calculates class labels and the probability of classes. Naive Bayes isn't made up of a single algorithm for classification but it includes a large number of algorithms that work on a single principle for training classifiers and the principal states that the value of a particular feature is autonomous of the value of any other feature specified in a class. In the past classification of news articles, Naive Bayes was used. But due to its incorrect parameter assessment revamped accuracy was reported. The best thing about the Naive Bayes algorithm is that it works equally well on both textual as well as numeric data and it is easy to implement and calculate. But it shows poor performance when the features are correlated like short text classification.

These models we used along with the basic tokenization. In the case wherein we used the BERT model, the Logistic Regression model turned out to be the one with the highest accuracy.

Results

The following bar graph displays the accuracies of the models we have used. We realized choosing and implementing classifiers are very important for getting satisfactory results. Logistic Regression was the first classifier trained on the corpus. It achieved 58% accuracy after applying data-preprocessing and stop-word removal. After experimenting with the Logistic Regression classifier, we moved to Naive Bayes approaches to get better results. However, the precision of the Naive Bayes classifier decreased to 50%. Post Naive Bayes, we moved to Decision Tree Classifier, where our accuracy went down further to 46%. Finally, we implemented a Support Vector Machine Model. We expected an improvement in this accuracy knowing the nature of the algorithm, and as expected it rose to 55%. However in conclusion Logistic regression (with BERT tokenization) worked best for our data.

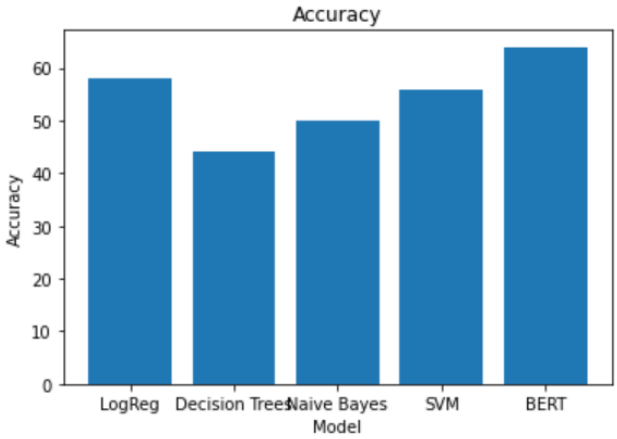

Fig (b)

Using the results of the best model, in terms of accuracy, we created a confusion matrix to understand what classes are the most confused ones. According to the results, we created a small data frame to display what the most confused classes were. "Crime" turned out to be the most confused class. It got confused with Food & Drink and Sports. The next most confused classes were "Entertainment", "Tech", "College", "Wellness", "Weddings".
The confusion probably occurs due to lack of data or bias towards the data which contain the most number of samples.

Results: The Most Confused Classes

	CRIME	ENTERTAINMENT	TECH	COLLEGE	WELLNESS	WEDDINGS
1	FOOD & DRINK	TASTE	SPORTS	SPORTS	FOOD & DRINK	TASTE
0	SPORTS	SPORTS	FOOD & DRINK	TASTE	MEDIA	FOOD & DRINK

Conclusion and Future Work

This paper reviews news classification in general using various algorithms. We have gone through pre-processing, feature selection, and news headlines classification. We have also done stop word filtering as a part of pre-processing from the nltk corpus.

The accuracy was approximately 50%, which was lower than what we expected,but we realised this mainly because there are too many categories in our data set. Also it is clear from the pie chart above that the data is not equally partitioned, which means most of the data belongs 3-4 categories despite there being 41 in total. This was an apparent reason for most of the algorithms having a low accuracy. Hence one important question for future work in classification tasks like these is: How many categories are ideal for classification?

We also realised, structure of data plays an important role in text classification problems. Since we found our dataset on Kaggle, it was ideal for running machine learning tasks on it. However if we had obtained our data from a native web scraper, then our data would not have been so structured. This would have demanded much more work in terms of data cleaning. Moreover, all the data we trained our algorithms on were from the same news company Huffpost. So we did

not have trouble with naming conventions. Question still remains how the results would change if we obtained data from different news websites which use different naming conventions.

Another important observation we made was the importance tokenization methods have on the results of a machine learning algorithm. At first we were using the basic NLTK tokenizer and we were getting accuracy of 58% on training our data on logistic regression. We were using this as our baseline model. After this, we built on our baseline model and included the BERT Base Uncased Model for tokenization. On doing this, our accuracy after training the algorithm on logistic regression increased to 64%.

Artificial Neural Networks are able to handle complex classification tasks more effectively, especially when provided larger volumes of data. So, we would like to implement neural network models like RNN (Recurrent Neural Network) and LSTM (Long Short Term Memory). The reason for us to consider these models specifically is that they perform well on sequential data. We plan on comparing the results from these models with the results from our current models and exploring the factors that best affect the models' understanding of the text's sentiment.

In conclusion, our future work in News Classification would involve selecting a dataset that is equally partitioned, running experiments on the relation between number of categories and accuracy, comparison of algorithm performance from different datasets and testing our algorithms on Neural Network based models.

References

[1] Rana, Mazhar Iqbal, et al. “News Headlines Classification Using Probabilistic Approach.” VAWKUM Transactions on Computer Sciences, vol. 7, no. 1, Sept. 2015. DOI.org (Crossref), https://doi.org/10.21015/vtcs.v7i1.341.

[2] Ramchandra Kawade, Dipak, and Kavita S. Oza. “News Classification: A Data Mining Approach.” *Indian Journal of Science and Technology*, vol. 9, no. 46, Dec. 2016. *DOI.org (Crossref)*, https://doi.org/10.17485/ijst/2016/v9i46/84444.

[3] Rishabh Misra. *News Category Dataset*. Unpublished, 2018. *DOI.org (Datacite)*, https://doi.org/10.13140/RG.2.2.20331.18729.