News Subject Predictor

News has grown to take on more forms than ever. It's no longer just the 7pm Broad-cast on television. It's the websites we visit, the tweets we read, the facebook posts we like, it's any number of media platforms that we engage with on a daily basis. In these busy times the demand for short doses of crisp and authentic news has been on an all-time high. In that bid In-shorts is a mobile news application which produces 60 word news content and serves all kind of subjects to its users. I will create a news classification model that will tell the type of news subject from that 60 word news content.

1. Data

The scrapped data from the Inshorts website is available on Kaggle on the below mentioned link. It contains around 55,000 news sample of varying news types. The news type has not been explicitly mentioned and that will be discussed in the pre-processing part. The data however contains the headline of the subject, the actual 60 word news content, the source of the news, the time of publish of the news and the date of publish.

https://www.kaggle.com/shashichander009/inshorts-news-data

2. Data Wrangling and Pre-Processing

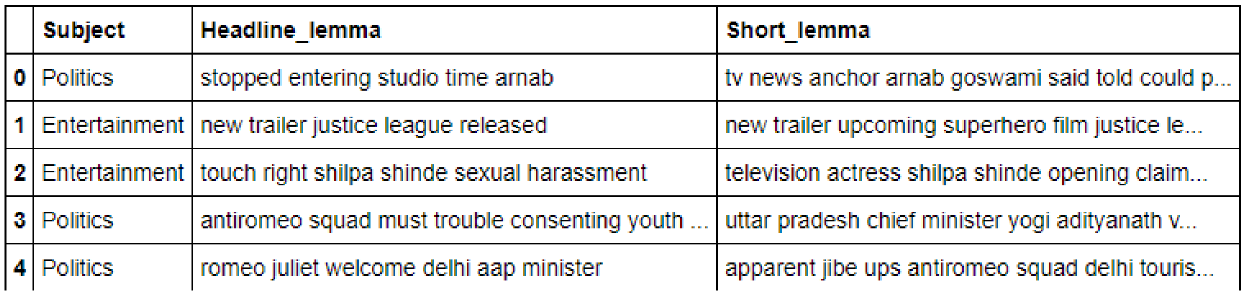
Since our project was based on subject prediction, the source, time & date of publish of news were dropped from the data-frame. Now since this is a classification task we need target labels depicting the news subject of the content provided. Unfortunately we do not have the subjects already stated hence in that case we need to manually label the news. Since this is a tedious task therefore we only labelled around 25k samples. Five topics were chosen as below:

1.) Sports
2.) Business
3.) Science
4.) Entertainment
5.) Politics

After that the news content just like any other text data had to be cleaned and in order to do that we performed the following cleaning steps:

1.) Removing Punctuations & Special characters
2.) Removing numbers
3.) Removing stop-words
4.) Lowercasing the words
5.) Lemmatization

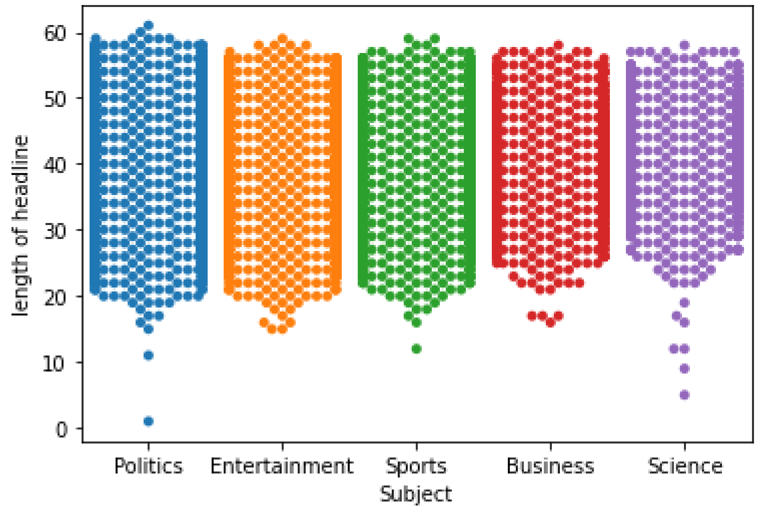
The final data-frame that we got was as below:

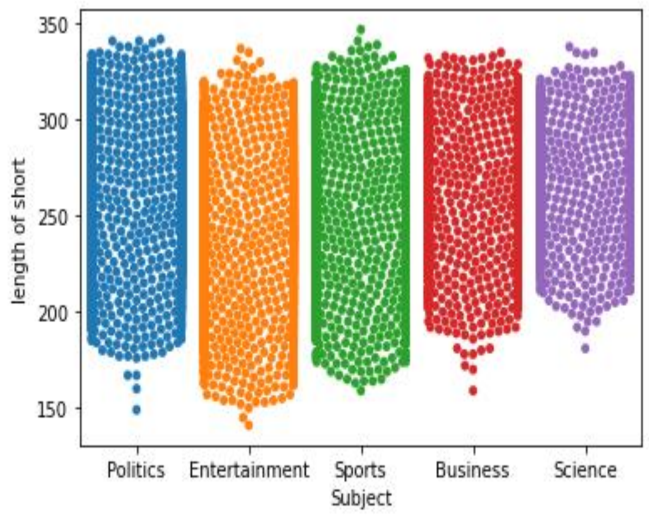


	Subject	Headline_lemma	Short_lemma
0	Politics	stopped entering studio time arnab	tv news anchor arnab goswami said told could p...
1	Entertainment	new trailer justice league released	new trailer upcoming superhero film justice le...
2	Entertainment	touch right shilpa shinde sexual harassment	television actress shilpa shinde opening claim...
3	Politics	antiromeo squad must trouble consenting youth ...	uttar pradesh chief minister yogi adityanath v...
4	Politics	romeo juliet welcome delhi aap minister	apparent jibe ups antiromeo squad delhi touris...

3. EDA

It would have been interesting to see if there was any relationship between the word count of headlines and the content for each news subject. Also the comparison of word count for every subject of news should be compared.

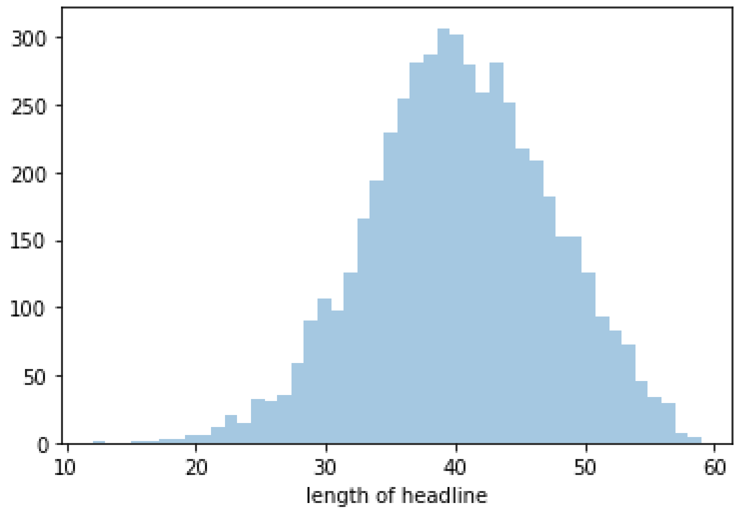




As can be seen from the above two plots, the variation in word count for both headline and the short content is almost comparable irrespective of the news subject. Similarly for the distribution of word count, no particular trend was observed. The spikes were random in the histograms, however for the sports subject it is noteworthy to mention that the word count follows a normal distribution.

```
# Distribution of the word count for Sports subject
sns.distplot(df[df['Subject']=='Sports']['length of headline'], kde=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xb3dcdfed88>
```



This trend however doesn't aid in the problem that we are trying to solve. Although it might be interesting to research further on this.

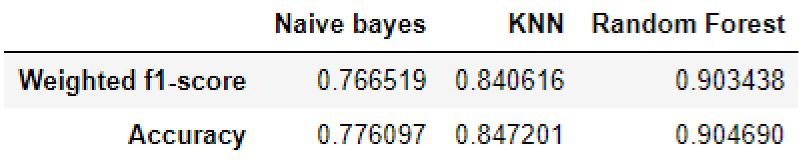
4. Pre-processing and modelling

The cleaned text however cannot be directly fed into the model but has to be first transformed into some sort of word embedding. In our approach we have used tf-idf technique to make a vocabulary which will govern the digit assignment to the words.

We split the dataset into training and testing in the ratio 2:1. The tf-idf resulted in a vocabulary size of 38,879 unique words. Each unique word will be treated as a feature to the chosen model. The shape of the training set turns out to be (16881, 38879).

Since processing of such sample and feature size is not feasible on a local CPU. We decided to shrink the dataset to 2000 samples and split it in the same ratio.

The model performance on the metrics were as follows:



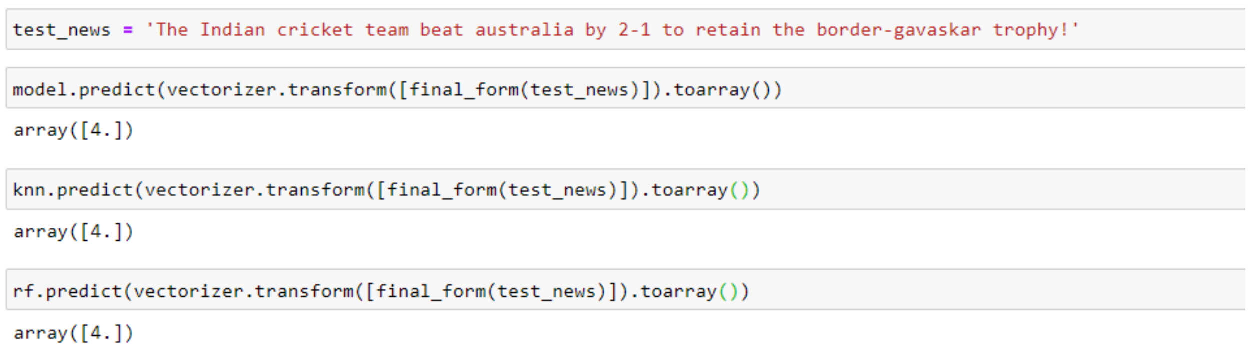
	Naive bayes	KNN	Random Forest
Weighted f1-score	0.766519	0.840616	0.903438
Accuracy	0.776097	0.847201	0.904690

Clearly the winner is Random Forest with 90% accuracy. We performed the grid search on the random forest model to figure out the best hyper-parameters.

Below were the findings:

n_estimators=1000, n_jobs=-1, random_state=1

5. Predictions:



```
test_news = 'The Indian cricket team beat australia by 2-1 to retain the border-gavaskar trophy!'

model.predict(vectorizer.transform([final_form(test_news)]).toarray())

array([4.])

knn.predict(vectorizer.transform([final_form(test_news)]).toarray())

array([4.])

rf.predict(vectorizer.transform([final_form(test_news)]).toarray())

array([4.])
```

As can be seen from the image above, all of the models are correctly able to predict the subject of the news as Sports. Here the number 4 was encoded as Sports.

6. Future Improvements

Given that we have to solve the same problem, there are multiple things that can be done in future:

(i) Other types of word embeddings can be adapted instead of tf-idf such as GloVe, word2vec or BERT based.

(ii) Having more data is always beneficial but due to our constraints of RAM we couldn’t make full use of our data. The model in future can be trained with a GPU support.

(iii) Although we got a 90% accuracy with random forest, still advanced models such as XGboost can be used for prediction in future for better accuracies.

7. Credits

I would like to thank my Springboard mentor Mr. Himanshu Jain for being super supportive and guiding me to solve this problem.