

# BBC News Document Classification

1<sup>st</sup> Varnika Toshniwal  
*Data Science*  
*Stevens Institute of Technology*  
Hoboken, USA  
vtoshniw@stevens.edu

2<sup>nd</sup> Yogesh Awdhut Gadade  
*Data Science*  
*Stevens Institute of Technology*  
Hoboken, USA  
ygadade@stevens.edu

3<sup>rd</sup> Patrick Cullinane  
*Data Science*  
*Stevens Institute of Technology*  
Hoboken, USA  
pcullina@stevens.edu

**Abstract**—We intend to classify news articles from the BBC news documents using a variety of classification techniques. To perform this project we will need to perform several steps that are characteristic of Natural Language Processing (NLP). First we will conduct preprocessing of the text which will consist standardizing text and then turning into a tokenized version which can be processed by our classification algorithms. Finally we will evaluate our algorithms using precision, recall, and F-1 score. For this task there is no preference for precision or recall and rather we will seek to maximize both. Previously there has been some work done on text document classification.

## I. INTRODUCTION

These days on the Internet there are a ton of sources that create colossal measures of everyday news. What's more, the interest for data by clients has been developing consistently, so it is significant that the news is ordered to permit clients to get to the data of interest rapidly and adequately. Thus, the AI model for mechanized news arrangement could be utilized to recognize subjects of unmanaged news or potentially make individual ideas dependent on the client's earlier advantages. Accordingly, our point is to assemble models that take as information news feature and short portrayal and yield news classification.

Text document classification is an important task for diverse natural language processing based applications. To perform document classification algorithmically, text documents need to be represented such that it is understandable to the machine learning classifier model.

The news articles are scattered into different text documents in different folders with category names. Before starting preprocessing we need to create a program to collect the scattered data and label it according to the folder names so that we have dataset with us into a single dataframe ready to preprocess for further analysis.

The report discusses about the preprocessing part, insights we got from the raw data, feature vector through which data is represented and later classified. The project aims at exploring text mining concepts to learn insights from the text document by performing EDA and comparing the machine learning algorithms linear SVM, Random Forest, XGBoost to classify BBC News articles into 5 different classes which are Business, Tech, Sports, Entertainment, Politics.

We are implementing preprocessing on given input data to create feature vector representation which will be the input to above mentioned algorithms. To test how well each of the three mentioned feature vectors perform, we used the BBC

new articles database and converted the documents to all the feature vectors. We will be evaluating the performance of these algorithms based on Precision, Recall and F1 score and will explore the techniques that can be used to optimize. Based on their performance we will be selecting the best model for BBC news text document classification.

## II. RELATED WORK

- 1) <https://www.cs.drexel.edu/~spiros/papers/NLP20.pdf>
- 2) <http://derekgreene.com/papers/belford18eswa.pdf>  
Stability of Topic Modeling via Matrix Factorization
  - a) This paper and some associated code shows a related topic on how to measure semantic themes within a corpus (Topic Modeling). The general approach to data cleaning and structure of tokenization of a corpus was helpful to understanding how to approach our project.
- 3) <https://arxiv.org/abs/1607.03055>: Exploring the Political Agenda of the European Parliament Using a Dynamic Topic Modeling Approach
  - a) This paper expands further on paper #3 and shows how to apply topic modeling to real world events.
- 4) <http://derekgreene.com/papers/ocallaghan15eswa.pdf>  
An analysis of the coherence of descriptors in topic modeling
  - a) This paper details an interesting approach to determining best k topics in a model. K-fold has silhouette and elbow methods for determining optimal cluster number, and this technique uses an interesting approach that combines tfidf and word2vec to calculate coherence score. Again this is a topic model/unsupervised approach but has some influence over how we can approach preprocessing.
- 5) [https://scholarworks.sjsu.edu/etd\\_projects/531](https://scholarworks.sjsu.edu/etd_projects/531)
  - a) The report discusses the different types of feature vectors through which document can be represented and later classified.

## III. OUR SOLUTION

### A. Description of Dataset

The dataset consists of 2225 rows of news stories from the BBC between the years of 2004 to 2005. The articles are assigned 1 of 5 different categories, consisting of: 'business',

```
1 [w for w in df.iloc[0:1,]['Text']]
```

['worldcom ex-boss launches defence lawyers defending former worldcom chief bernie ebbers aga  
inst a battery of fraud charges have called a company whistleblower as their first witness.  
cynthia cooper worldcom s ex-head of internal accounting alerted directors to irregular acc  
ounting practices at the us telecoms giant in 2002. her warnings led to the collapse of the f  
irm following the discovery of an \$1bn (£5.7bn) accounting fraud. mr ebbers has pleaded not  
guilty to charges of fraud and conspiracy. prosecution lawyers have argued that mr ebbers or  
cheated a series of accounting tricks at worldcom ordering employees to hide expenses and  
inflate revenues to meet wall street earnings estimates. but ms cooper who now runs her own  
consulting business told a jury in new york on wednesday that external auditors arthur ander  
sen had approved worldcom s accounting in early 2001 and 2002. she said andersen had given a  
green light to the procedures and practices used by worldcom. mr ebber s lawyers have said h  
e was unaware of the fraud arguing that auditors did not alert him to any problems. ms coop  
er also said that during shareholder meetings mr ebbers often passed over technical questions  
to the company s finance chief giving only brief answers himself. the prosecution s star w  
itness former worldcom financial chief scott sullivan has said that mr ebbers ordered accou  
nting adjustments at the firm telling him to hit our books . however ms cooper said mr sul  
livan had not mentioned anything uncomfortable about worldcom s accounting during a 2001 au  
dit committee meeting. mr ebbers could face a jail sentence of 85 years if convicted of all t  
he charges he is facing. worldcom emerged from bankruptcy protection in 2004 and is now know  
n as mci. last week mci agreed to a buyout by verizon communications in a deal valued at \$6.  
75bn. ']

Fig. 1. Example text from an article in the dataset.

'tech', 'politics', 'sport', 'entertainment'. Because this project will attempt to classify news stories based on the text content data preprocessing will consist of tokenization of words. To achieve this our team will perform several steps to ready the text to be tokenized for analysis.

The basic steps will be to first perform basic cleaning on the data which will consist of removing punctuation, lowercases, and de-accents each word though the gensim libraries function simple\_preprocess. Next stop words will be removed using the NLTK stop word list. Stop words are words that do not contribute to the overall semantic meaning of the document and can be removed. Examples for English are and, the, was, etc.. A full list can be found in NLTK documentation.

Next a stemmer will be applied to to reduce words to their word stem, such as 'sing' can be derived from 'sang' and 'sung', and ultimately will mean the same thing in the context of classification.

Finally, we will apply ngrams to the word. In the case of bigrams we are joining words that are next to each other together into one word. The idea behind this is that a word's meaning is thought to be derived from the company it keeps. Documents that have the words 'baseball', 'glove', and 'hitter' are more likely to be about baseball for example.

Once we have applied the preprocessing steps we can tokenize the words. The two major techniques that we will employ are bag of words and term-frequency inverse document frequency (TFIDF). In the bag of words approach we are just simply counting the times that a word appears in text. This is ultimately recording a document-term matrix where the dimensions are documents on one axes and terms on the other. The entries record how frequently a word appears in that particular document. We can then go a step further and use a technique called TFIDF as a means to better sort "important" words. The idea behind tfidf is to look for words that occur frequently within a small amount of documents and infrequently within the whole corpus. Term frequency is the same as the bag of words approach but is then weighted by inverse document frequency by penalizing words that appear too often throughout the whole corpus (all the documents).

In theory we should expect the most important words associated with entertainment, sports and other categories to appear in their associated documents. Of course we assume this will better help us to classify words that occur in their respective labels and increase the performance of our model.

Some Preliminary data on the structure of our dataset: From the above output it can be seen that there is no missing

```
1 print("Null values: \n\n",pdInputData.isnull().sum())
```

Null values:  
NewsText 0  
NewsType 0  
dtype: int64

```
1 print("Dropping duplicates: ", len(pdInputData)-len(pdInputData[NewsText].drop_duplicates()) )
```

```
2 print("Before dropping duplicates number of rows: ", len(pdInputData))
```

```
3 pdInputData=pdInputData[[NewsText, NewsType]].drop_duplicates().reset_index()[[NewsText, NewsType]]
```

```
4 print("After dropping duplicates number of rows: ", len(pdInputData))
```

Dropping duplicates: 98

Fig. 2. Gaps in raw input data

data but we found there are 98 duplicate data from which we are going to keep only one row for each duplicate pair by using the pandas function drop\_duplicate(keep='last').

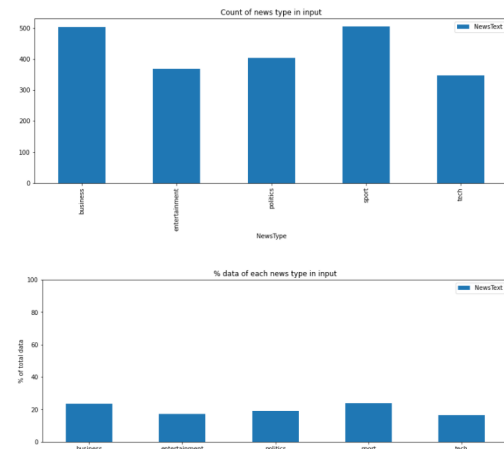


Fig. 3. Data representation

From the above graphs we can see the bare count of number of samples for each categories and % samples out of total data samples distribution of data is more or less same.

Below chart shows chi-square analysis to find correlation between features (importance of words) and labels (news category).

```
# Category: 'politics':
. Most correlated unigrams:
. business
. sport
. politics
. Most correlated bigrams:
.

# Category: 'business':
. Most correlated unigrams:
. politics
. sport
. business
. Most correlated bigrams:
.

# Category: 'entertainment':
. Most correlated unigrams:
. business
. sport
. entertainment
. Most correlated bigrams:
.

# Category: 'sport':
. Most correlated unigrams:
. politics
. business
. sport
. Most correlated bigrams:
.

# Category: 'tech':
. Most correlated unigrams:
. business
. sport
. tech
. Most correlated bigrams:
.
```

Fig. 4. For each category, find words that are highly correlated to it



accuracy: 0.9553990610328639

Classification report:

	precision	recall	f1-score	support	Confusion matrix:
0	0.94	0.99	0.96	97	[[ 96  0  0  0  1] [  3 65  4  1  0] [  1  0 76  1  0] [  0  0 105  0] [  2  1  1  4 65]]
1	0.98	0.89	0.94	73	
2	0.94	0.97	0.96	78	
3	0.95	1.00	0.97	105	
4	0.98	0.89	0.94	73	
accuracy			0.96	426	
macro avg	0.96	0.95	0.95	426	
weighted avg	0.96	0.96	0.95	426	

Fig. 9. MNB results

```
print_missclassified_samples(X_test, test_predict, y_test)
```

```
Row 24 has been classified as business and should be tech
Row 112 has been classified as sport and should be tech
Row 138 has been classified as politics and should be tech
Row 163 has been classified as sport and should be tech
Row 206 has been classified as entertainment and should be tech
Row 218 has been classified as sport and should be tech
Row 316 has been classified as sport and should be tech
Row 359 has been classified as business and should be tech
```

Fig. 10. Mismatched sentences in MNB

Linear Support Vector Classifier: SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The algorithm creates a line or a hyperplane which separates the data into classes.

The objective of a Linear SVC (Support Vector Classifier) is to fit the provided data, returning a "best fit" hyperplane that divides or categorizes your data. From there, after getting the hyperplane, you can then feed some features to your classifier to see what the "predicted" class is. LinearSVC uses one vs the rest classifier.

The idea of SVM is simple: The algorithm creates a line or a hyperplane which separates the data into classes. Since the text has a lot of features and the linear kernel is good when there is a lot of features. That's because mapping the data to a higher-dimensional space does not really improve the performance. [3] In-text classification, both the numbers of instances (document) and features (words) are large.

Random Forest Classifier: Random forests is an ensemble learning method using here for the classification that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. News stories are typically organized by topics; content or products are often tagged by categories; users can be classified into cohorts based on how they talk about a product or brand online. In this study, we investigate the application of RF for news articles classification. Moreover, an important motivation for using RF was the application of late fusion strategies based on the RF operational capabilities. For the design purpose we are going to use sklearn library function RandomForestClassifier(). A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and

accuracy: 0.9812206572769953

Classification report:

	precision	recall	f1-score	support
0	1.00	0.98	0.99	97
1	0.97	0.99	0.98	73
2	0.99	0.97	0.98	78
3	0.97	1.00	0.99	105
4	0.97	0.96	0.97	73
accuracy			0.98	426
macro avg	0.98	0.98	0.98	426
weighted avg	0.98	0.98	0.98	426

Confusion matrix:

```
[[ 95  0  0  1  1]
 [  0 72  1  0  0]
 [  0  0 76  1  1]
 [  0  0  0 105  0]
 [  0  2  0  1 70]]
```

Fig. 11. Linear SVM result

```
print_missclassified_samples(X_test, test_predict, y_test)
```

```
Row 119 has been classified as entertainment and should be tech
Row 163 has been classified as sport and should be tech
Row 393 has been classified as entertainment and should be tech
```

Fig. 12. Mismatched sentences in MNB

control over-fitting. The sub-sample size is controlled with the max\_samples parameter if bootstrap=True (default), otherwise the whole data-set is used to build each tree.

XGBoost Classifier, which stands for Extreme Gradient Boost is a state of the art gradient boosting tree model described in the paper XGBoost: A Scalable Tree Boosting System (2016). Gradient boosting starts by making a prediction on the dataset and calculates the residuals. Another model is then fit to those residuals and the process continues until a specified amount of iterations is reached or the residual becomes too small. A prediction then is the sum of the predictions from each tree.

On top of gradient boosting XGBoost several features to include regularization. As the authors point out regularization helps to smooth out the learnt weights which reduces models complexity. When the regularization parameter is turned off (set to zero) the model becomes a gradient boosted tree. Another feature of XGB is what is called weighted quantile sketch, and this feature helps with candidate splitting. The authors point out that most tree models use random subsets of data or heuristics to split, and there approach is novel and prunes the model to maintain accuracy, which is a problem with tree based models in general. There are several features which increase the computational efficiency of the

accuracy: 0.9272300469483568

Classification report:

	precision	recall	f1-score	support
0	0.90	0.95	0.93	107
1	0.92	0.88	0.90	78
2	0.97	0.94	0.96	71
3	0.90	0.98	0.94	103
4	0.98	0.84	0.90	67
accuracy			0.93	426
macro avg	0.94	0.92	0.93	426
weighted avg	0.93	0.93	0.93	426

Confusion matrix:

```
[[102  1  1  2  1]
 [ 6 69  0  3  0]
 [ 2  0 67  2  0]
 [ 1  0  1 101  0]
 [ 2  5  0  4 56]]
```

Fig. 13. Random Forest result

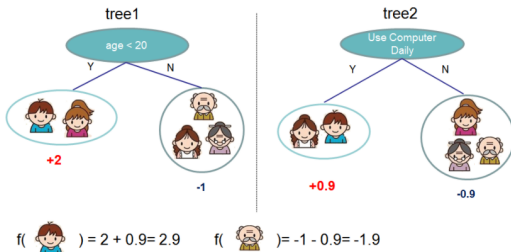


Fig. 14. Trees in an ensemble model: Credit <https://xgboost.readthedocs.io/en/stable/tutorials/model.html>

algorithm one of these features is out-of-core computation. The data is divided into multiple parts and stored in memory. Although outside of the scope of this paper the authors use block compression and block sharding to improve out of core computation. To use XGBoost for multi-class classification we had to use the multi:softmax objective function. The default value is binary:logistic, and the other options for multi-class was multi:prob. The model was implemented for this project in its out-of-the-box version and with max depth tuned. 3-fold cross validation was used to find the best max depth value, and due to the scope of our project the time to train other parameters was too computationally expensive. Ultimately a max depth value of 6 improved the model.

### C. Implementation Details

The text needs to be transformed to vectors so as the algorithms will be able make predictions. In this case it will be used the Term Frequency-Inverse Document Frequency (TFIDF) weight to evaluate how important a word is to a document in a collection of documents. After removing punctuation, lower casing, stop words, and lemmatization

accuracy: 0.9530516431924883

Classification report:

	precision	recall	f1-score	support
0	0.92	0.94	0.93	90
1	0.99	0.93	0.96	72
2	0.93	0.95	0.94	84
3	0.98	0.99	0.99	108
4	0.94	0.93	0.94	72
accuracy			0.95	426
macro avg	0.95	0.95	0.95	426
weighted avg	0.95	0.95	0.95	426

Confusion matrix:

```
[[ 85  0  4  0  1]
 [ 1 67  2  1  1]
 [ 2  0 80  1  1]
 [ 0  0  0 107  1]
 [ 4  1  0  0 67]]
```

Fig. 15. XGBoost before hyperparameter optimization

accuracy: 0.9553990610328639

Classification report:

	precision	recall	f1-score	support
0	0.95	0.96	0.96	108
1	0.93	0.93	0.93	61
2	0.97	0.92	0.94	91
3	0.97	0.99	0.98	97
4	0.94	0.96	0.95	69
accuracy			0.96	426
macro avg	0.95	0.95	0.95	426
weighted avg	0.96	0.96	0.96	426

Confusion matrix:

```
[[104  2  1  0  1]
 [ 1 57  2  0  1]
 [ 1  2 84  2  2]
 [ 1  0  0 96  0]
 [ 2  0  0  1 66]]
```

Fig. 16. XGBoost after parameter optimization

importance of a word is determined in terms of its frequency.

The original data will be divided into features (X) and target (y), which then will be split into train (80%) and test (20%) sets. 10 percent holdout for validation set for tuning hyper parameters. Thus, the algorithms will be trained on one set of data, validated and then tested out on a completely different set of data (not seen before by the algorithm). Our confusion matrix will have 5 classes.

We are going to use 3 algorithms with ensemble models such as Random Forest, XGBoost, Linear SVC. We will compare their test accuracy scores, precision, recall, and F1 scores.

when we will be evaluating we thinking we want to maximize f-1 score but also we want to see what will happen to precision and recall e.g. either precision or recall is higher. Ultimately those we want to classify as many documents correctly as possible the ROC curve won't matter because it is a multi-class classification problem.



#### IV. COMPARISON


$$\begin{aligned} \text{Precision} &= \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \\ \text{Recall} &= \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \\ \text{Accuracy} &= \frac{\text{True Positive} + \text{True Negative}}{\text{Total}} \end{aligned}$$


Fig. 17. Precision, Recall, Accuracy

$$\text{F1 Score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

Fig. 18. F1 score

Overall the models had very similar performance but there were subtle differences that can be seen after further examination of the confusion matrices. LinearSVC's recall scores varied between 0.97 and 1.00 while XGBoost had recalls ranging from 0.93 to a max of 0.98. Focusing in on the entertainment class (class 1) we can see XGBoost missed 4 samples while linearSVC only missed one. Both models had similar performance in terms of recall on class 3, 'sport', where linearSVC captured all the samples and XGBoost only missed one. In terms of precision XGBoost's scores ranged from 0.93 to 0.97 while linearSVC ranged from 0.97 to 1.00. Again looking at the sport label we see XGBoost misclassified 4 examples while linearSVC only misclassified 2. Both XGBoost and LinearSVC misclassified 3 samples for 'sport'. The random forest as expected performed worse than the previous two mentioned. For class 0 which corresponds to business it's precision score was 0.90 meaning it misclassified 11 samples. It's worth noting that hyperparameter tuning had an effect on the performance of the xgboost models between the out-of-the-box vs. the tuned model. Because of the size of the testing model the difference comes down to only a few examples though.

#### V. FUTURE DIRECTIONS

Future work will consist of examining the keywords that cause models to misclassify text and coming up with pre-processing strategies to mitigate these effects. Additionally, another area of improvement for this project would be to focus on more robust hyperparameter tuning of the model. In the case of the XGBoost model we would need to examine strategies to increase the computation power where the model is being trained. Possibly this can be done on a colab notebook with GPUs or a similar strategy. Although linearSVC performed well we should still examine parameter tuning strategies to improve upon it further. Additionally we would like to explore either moving these models to similar datasets with more examples to train and test on, or other use-cases. It would be interesting to see how well these models generalize to other newspapers or other forms of media (audio transcribed text).

Perhaps this model could be used at scale to sort unlabelled news stories.

#### VI. CONCLUSION

In summary we took text data from the BBC news dataset with each stories associated label and we fit models to classify those data. After exploratory data analysis we assumed that there was enough class balance to go ahead with the dataset as is. The first step we performed was to preprocess and clean the data using various techniques to ensure the most important words would come to the surface. To the text stopwords such as "a", and "the" were removed among a larger list imported from the NLTK library. Additionally we stemmed words, removed punctuation, lowercased, and turned the text into bigrams After cleaning the data we vectorized it using a technique called tfidf, which turned the news stories (documents) into a document-term matrix that was ready to be input into the model. As we approached the model training aspect of our project we took the class labels and encoded them into numeric values (0 through 4) and then split up our data into training and testing sets. In some cases a base line was established to see what our outcomes would be with techniques such as decision trees. The models we chose were random forest, naive bayes, linear support vector classifier, and XGBoost, implemented from scikit-learn and the XGBoost library. Each model was then trained and the output generated. In the case of XGBoost the max depth parameter of 6 was obtained through RandomizedSearchCV with 3 folds to obtain better performance from the XGBoost model. Random Forest was tuned on decision criteria, and ultimately gini was chosen. Each models performance was evaluated with precision, recall, and F1 scores. Additionally a confusion matrix to investigate stories that were misclassified.

On the BBC dataset it appears that linearSVC with one-vs-rest had the best performance of the four models tested on this dataset. LinearSVC had a 0.98 F1 score average while parameter-tuned XGBoost had 0.95. Random forest had similar performance to XGBoost after parameter tuning. The best performing model across all models appears to be label 3, "sport". LinearSVC had the highest recall score by capturing all instances of sport news stories in the dataset but labeled 3 stories as sport when they were other categories. XGBoost similarly labeled 3 stories as sport when they were other categories and missed one storie about sport so it had a slightly lower recall, and random forest was slightly right behind the first two models. In examining some of the outputs that were misclassified it appears that some of these contain keywords that other news topics can share.

#### REFERENCES

- 1) Document calssification Using ML: [https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1531&context=etd\\_projects](https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1531&context=etd_projects)
- 2) Data Source: <http://mlg.ucd.ie/datasets/bbc.html>
- 3) [https://en.wikipedia.org/wiki/John\\_Rupert\\_Firth](https://en.wikipedia.org/wiki/John_Rupert_Firth)
- 4) Machine Learning, by Tom M. Mitchell, McGraw-Hill: <http://cscog.likufanele.com/calvo/>

*Inteligencia Artificial files/McGrawHill,  
Tom.Mitchell. – Machine.Learning.pdf*

- 5) Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd Ed., by Aurelien Geron, O'Reilly Publication  
*https : //www.knowledgeisle.com/wp – content/uploads/2019/12/2 – Aur*

Our source code repo: <https://github.com/cullinap/cpe695-project>