Rosalie Day – Final Report, Capstone 2
Prediction of Category in News Text
March, 2020

The Purpose

Natural language processing is ubiquitous for the internet and the IOT universe. Classification provides an important tool for identifying subject matter of content. Results of classification models may provide information for composition that is easily classified and potentially insights on text that is not easily classified.

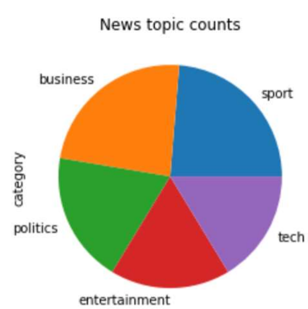
Summary of Findings

This analysis compares five standard Scikit Learn classification models and two extractors on the full text of more than 2000 news articles. The focus is on the discrete words for five news categories: business; entertainment; politics; sport; and tech. The models classify by the words, with no information on parts of speech, sequences or contexts in which they are used. The Logistic Regression, Naïve Bayes and Gradient Boosting classifiers, with default parameters, perform with an accuracy greater than nine out of ten on the two extraction types. The twenty most frequent words are found for the five respective categories.

The Data

The BBC news data set composed of 2225 full-text articles and appears in the Kaggle competition data sets. Downloaded as a comma separated file, the set has two variables: news category ('category') and full text of the articles ('text'). The text was already converted to lower case with punctuation removed. Data cleaning included removing duplicates.

	category	percentages
sport	504	24.0
business	503	24.0
politics	403	19.0
entertainment	369	17.0
tech	347	16.0



There remained 2126 articles and five categories.

The article text was prepared for tokenization by removing special characters. The list of "stopwords," articles, and other often used, meaningless words when taken out of context, was provided in the Natural Language Toolkit package. The stopwords were removed in separate steps so the length of the articles, in this case, number of tokens, could be compared with and without stopwords. Stopwords could be generated according to frequency for this corpus (or hand compiled) as a parameter tuning for the Bag of Words and Term Frequency-Inverse Document Frequency (TF-IDF) extractors in SciKit-Learn.

20 Most Common Words for Each Category

Business		Entertainment		Politics		Sport		Tech	
term	**count**	**term**	**count**	**term**	**count**	**term**	**count**	**term**	**count**
said	1655	said	803	said	2174	said	927	said	1368
us	787	film	711	mr	1614	game	468	people	828
year	623	best	563	would	1005	england	455	also	460
mr	596	music	423	labour	728	first	432	mr	451
would	459	also	382	government	712	win	408	technology	450
also	432	one	350	people	607	would	396	new	449
market	417	us	350	party	544	world	374	one	437
company	412	year	346	election	536	last	367	could	426
new	401	show	318	blair	535	one	354	mobile	420
growth	362	new	315	also	438	two	349	would	415
firm	358	awards	268	new	419	time	325	games	382
last	356	first	238	minister	413	also	323	users	343
economy	342	last	231	could	372	back	316	use	337
government	335	award	230	brown	356	players	304	us	325
bank	331	years	220	told	340	play	291	game	311
sales	316	uk	216	uk	335	cup	290	many	310
2004	304	two	206	howard	309	new	285	music	310
could	302	tv	205	public	304	side	267	net	306
economic	302	people	203	plans	302	year	265	digital	298
oil	291	band	203	one	302	team	263	software	298

Text Data Preparation for Classification Models

In preparation for the classification models, the tokens were extracted and vectorized, converted to numeric coding, for machine learning models. The two extraction outputs were: a "Bag of Words," word counts in the entire body of texts, the "corpus;" and Term Frequency-Inverse Document Frequency ("TF-IDF"), words weighted for importance.

Bag of Words

CountVectorizer was used to extract a Bag of Words, words pooled with no use or sequence information, for the corpus. This extractor creates a feature for each token. In its default setting, "ngram_range; tuple (min_n, max_n), default=(1, 1)" *from Scikit-Learn documentation*, a token is one word. Token, word and term will be used interchangeably, as will document and article because one refers to the other in this analysis.

The Bag of Words, in its transformed output, is a document-term matrix, composed of word features in the columns – a vocabulary of 29,241 and rows corresponding to the 2,126 articles. The cells in the matrix are filled with the count of how many times the word (feature column) occurred in the article designated by the row.

Glimpse of the Document-Term Matrix

	00	000	0001	000bn	000m	000s	000th	001	001and	001st	...	zooms	zooropa	zornotza	zorro	zubair	zuluaga	zurich
0	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0

3 rows × 29421 columns

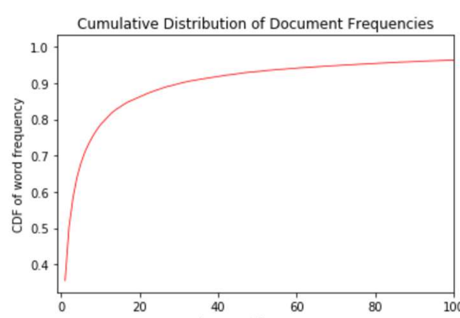
The above graphic shows a glimpse of the document-term matrix. The table shows the numeric code for the word and the sum of the frequency the words occurred for all the articles. The second column corresponds to the sum of every column in the matrix above, word code "000" occurs 756 times.

Word-Document Frequency Table

	Word	Article_Frequency
0	00	6
1	000	756
2	0001	1
3	000bn	1
4	000m	41

One intuitive parameter to tune in CountVectorizer is directing the extractor with respect to the minimum occurrence of that term. Called the minimum document frequency, "min_df," or the cut-off, the default is one time (min_df = 1). Cumulative distribution of document frequencies

By inspecting the cumulative distribution of word frequencies by number of documents, the minimum probably is greater than one. The fitted and transformed text with CountVectorizer with the minimum document frequency of 2 (min_df= 2).



This reduces the total number of features, the vocabulary, to 17240.

For optimal tuning, the model is also considered. A combination of trial parameters for the extractor and the respective preferred model(s) may be needed for optimization.

Classification Models

The performance of the supervised machine learning classification models on the Bag of Words was generally good, with at least 94% accuracy in three models. The data was randomly split for training on 70 percent and testing on 30 percent. This proportion was used throughout the analysis.

The classification models included K-Nearest Neighbors (KNN), Logistic Regression, Random Forest, Gradient Boosting, and Naïve Bayes, the multinomial classifier (Naïve Bayes). They were all run with their default parameter in Scikit-Learn.

The table below shows the performance metrics that were calculated and displayed in a classification report for each model. Confusion matrices show detailed summaries of how particular models classified on the news categories correctly (true positives and negatives) or incorrectly (false positives and negatives). The best performing model, Naïve Bayes, made a 97 percent accurate classification of the categories on the test data, the 30 percent held out when the model was fitted. The models are arranged in descending performance in the table below.

Sequence left to right and top to bottom is: business; entertainment; politics; sport; tech.

		precision	recall	f1-score	support
Naïve Bayes					
[[144 1 2 0 3]	business	0.96	0.96	0.96	150
[1 105 2 0 2]	entertainment	0.98	0.96	0.97	110
[2 1 116 0 0]	politics	0.95	0.97	0.96	119
[0 0 0 155 0]	sport	1.00	1.00	1.00	155
[3 1 1 0 99]]	tech	0.95	0.94	0.95	104
	accuracy			**0.97**	638
Logistic Regression		precision	recall	f1-score	support
[[144 0 4 0 2]	business	0.93	0.96	0.94	150
[2 105 1 0 2]	entertainment	0.95	0.95	0.95	110
[2 2 114 1 0]	politics	0.95	0.95	0.95	119
[0 0 0 155 0]	sport	0.99	0.99	0.99	155
[7 2 1 0 94]]	tech	0.94	0.89	0.92	104
	accuracy			**0.95**	638
Gradient Boost		precision	recall	f1-score	support
[[140 3 5 0 2]					
[3 105 0 0 2]	business	0.89	0.93	0.91	150
[8 0 108 2 1]	entertainment	0.93	0.95	0.94	110
[0 0 0 155 0]	politics	0.95	0.90	0.92	119
[9 1 1 0 93]]	sport	0.99	1.00	0.99	155
	tech	0.95	0.89	0.92	104

	accuracy **0.94** 638
Random Forest [[143 1 3 1 2] [4 100 2 3 1] [12 4 100 2 1] [5 3 4 143 0] [16 6 3 6 73]]	precision recall f1-score support business 0.78 0.97 0.87 150 entertainment 0.88 0.77 0.82 110 politics 0.93 0.84 0.88 119 sport 0.89 0.97 0.93 155 tech 0.95 0.73 0.83 104 **accuracy** **0.87** 638
KNN [[129 0 2 17 2] [19 59 2 28 2] [25 4 81 8 1] [5 3 2 143 2] [38 7 7 4 48]]	precision recall f1-score support business 0.59 0.88 0.71 150 entertainment 0.76 0.52 0.62 110 politics 0.86 0.68 0.76 119 sport 0.75 0.92 0.83 155 tech 0.88 0.47 0.61 104 **accuracy** **0.72** 638

The best performing model with an accuracy of .97 is unlikely to get better. Logistic Regression, second best, was sensitive to whether the special characters had been removed or not (not shown here), performed with f1-scores of .95 and .96 respectively. This model potentially could perform better on the Bag of Words with the tuning of the regularization parameter, "C." The default value is 1.

The parameter “C” was tuned by running a GridSearchCV, on 5 folds of the data, with alternative values of the parameter and the default. Cross validation partitioned the training data into fifths, withholding a partition for testing, and fitting every combination of the remaining 4 partitions. *Smaller values of “C” specify stronger regularization – Scikit-Learn documentation*. The default parameter, with alternatives 0.5 and 5.0, was the best with a score of .967.

Also, the small difference between the scoring on the Bag of Words with and without special characters could be an artefact of Scikit-Learn. The documentation notes that “the underlying C implementation uses a random number generator to select features when fitting the model. It is thus not uncommon, to have slightly different results for the same input data.”

Classification models on TF-IDF ("term importance") text –
Three best performing models: Naive Bayes; Logistic Regression; Gradient Boosting

The final step is using a less general extractor with the top performing models. The TF-IDF extraction calculates the term frequency in the documents weighted by the frequency of the term in the corpus for a relative importance measure. The Count Vectorizer and TF-IDF Vectorizer required the same processing and split of the data.

The Logistic Regression model classifies the news categories best on the TF-IDF extracted text. The table below shows the three classification models run the two extractions of the article text.

Confusion matrix, sequence left to right and top to bottom is: business; entertainment; politics; sport; tech.

Bag of Words		Importance weighted	
NaiveBayes [[144 1 2 0 3] [1 105 2 0 2] [2 1 116 0 0] [0 0 0 155 0] [3 1 1 0 99]]	f1 = .97 sport accuracy = 1	NaiveBayes on TF-IDF [[148 0 1 0 1] [5 87 7 11 0] [2 0 115 2 0] [0 0 0 155 0] [6 0 5 7 86]]	f1 = .93 accuracy decreases from Bag of Words decrease in accuracy in predicting politics and tech
LogisticRegression [[144 0 4 0 2] [2 105 1 0 2] [2 2 114 1 0] [0 0 0 155 0] [7 2 1 0 94]]	f1 = .95 f1 = .96 (w/o removing special characters) sport accuracy = 1	LogisticRegression on TF-IDF [[147 0 1 0 2] [3 104 0 0 3] [3 1 114 1 0] [0 0 0 155 0] [6 0 1 1 96]]	f1 = .97 accuracy increases from Bag of Words
GradientBoost [[140 3 5 0 2] [3 105 0 0 2] [8 0 108 2 1] [0 0 0 155 0] [9 1 1 0 93]]	f1 = .94 sport accuracy = 1	GradientBoosting on TF-IDF [[142 2 4 0 2] [6 101 0 1 2] [6 1 110 2 0] [0 0 1 154 0] [5 3 1 0 95]]	f1 = .94 sport accuracy = .99

The Logistic Regression classifier had an average accuracy of .96 across the two extractions of the text. Naïve Bayes averaged .95 and Gradient Boosting remained the same at .94. These classification models, with default parameters, classified the words associated with the five categories better than nine out of ten times and showed very little sensitivity to the method of extraction.

More research

Performance of unsupervised models, cluster analysis and principal component analysis, would be interesting for comparison.