Springboard

Web content sentiment analysis: News

Karolina Bzdusek

Springboard

July 22th, 2019

Contents

1. Introduction

Automation is a crucial part of any business. On the Internet we have immense source of data and information. To maintain web content, and to be able to allow users to access the information of interest quickly and effectively, is highly important for the owner of the various websites to have labeled their data correctly. To define categories, or keywords that would help users to get around in their content. The more accurate way then just using keywords it to build semantic search engine. So if someone writes a query "expressionism in the 19th century" we would be able to predict which articles are closest to this query (aka articles about expressionism in the 19th century, or related to this query).

We have concentrate of one of the above mentioned applications - tagging articles from news.This dataset contains 200,853 news headlines from the year 2012 to 2018 obtained from HuffPost. The model trained on this data set is used to identify tags for untracked news articles.

The questions that we want the answer are the following:

Can we categorize news articles based on their headlines and short descriptions?

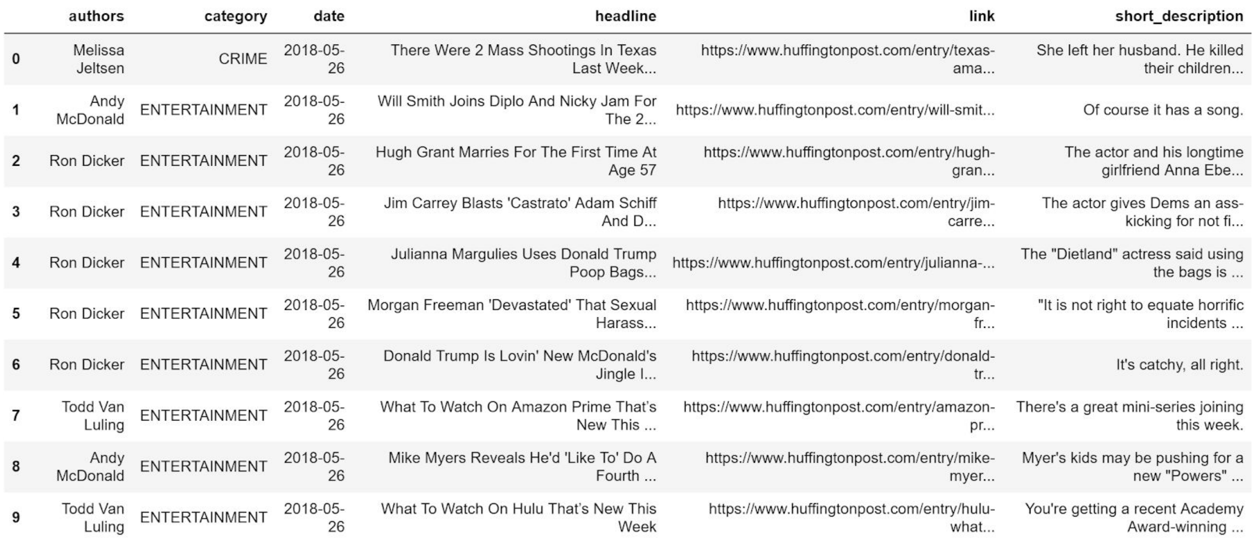
Can we create a semantic vectors that represent content of each article?

Source of data set: https://rishabhmisra.github.io/publications/

Data contain 6 features : authors, category, date, headline, link and short description.

Features link and date has no values in terms of predicting categories[1], so we would not consider them.

Typically authors is focus in some area within they are publishing their articles. However, in their main focus is or example ‘crime’, it does not mean that this is exclusive category for them. They may also publish within categories such as ‘parents’ or ‘politics’. This could lead to a leakage of the data. Therefore the feature ‘author’ we will not consider as well and our model would be based purely on ‘headline’ and ‘short description” features.



	authors	category	date	headline	link	short_description
0	Melissa Jeltsen	CRIME	2018-05-26	There Were 2 Mass Shootings In Texas Last Week...	https://www.huffingtonpost.com/entry/texas-ama...	She left her husband. He killed their children...
1	Andy McDonald	ENTERTAINMENT	2018-05-26	Will Smith Joins Diplo And Nicky Jam For The 2...	https://www.huffingtonpost.com/entry/will-smit...	Of course it has a song.
2	Ron Dicker	ENTERTAINMENT	2018-05-26	Hugh Grant Marries For The First Time At Age 57	https://www.huffingtonpost.com/entry/hugh-gran...	The actor and his longtime girlfriend Anna Ebe...
3	Ron Dicker	ENTERTAINMENT	2018-05-26	Jim Carrey Blasts 'Castrato' Adam Schiff And D...	https://www.huffingtonpost.com/entry/jim-carre...	The actor gives Dems an ass-kicking for not fi...
4	Ron Dicker	ENTERTAINMENT	2018-05-26	Julianna Margulies Uses Donald Trump Poop Bags...	https://www.huffingtonpost.com/entry/julianna-...	The "Dietland" actress said using the bags is ...
5	Ron Dicker	ENTERTAINMENT	2018-05-26	Morgan Freeman 'Devastated' That Sexual Harass...	https://www.huffingtonpost.com/entry/morgan-fr...	"It is not right to equate horrific incidents ...
6	Ron Dicker	ENTERTAINMENT	2018-05-26	Donald Trump Is Lovin' New McDonald's Jingle I...	https://www.huffingtonpost.com/entry/donald-tr...	It's catchy, all right.
7	Todd Van Luling	ENTERTAINMENT	2018-05-26	What To Watch On Amazon Prime That’s New This ...	https://www.huffingtonpost.com/entry/amazon-pr...	There's a great mini-series joining this week.
8	Andy McDonald	ENTERTAINMENT	2018-05-26	Mike Myers Reveals He'd 'Like To' Do A Fourth ...	https://www.huffingtonpost.com/entry/mike-myer...	Myer's kids may be pushing for a new "Powers" ...
9	Todd Van Luling	ENTERTAINMENT	2018-05-26	What To Watch On Hulu That’s New This Week	https://www.huffingtonpost.com/entry/hulu-what...	You're getting a recent Academy Award-winning ...

As it is an NLP task, the dimensionality is high (each unique word is a feature), therefore some cleaning of the text is needed. Other possibility is stemming/lemmatization/stop words techniques and choice of our tokenizer as well. These are ways to reduce dimensionality before transforming words to vectors. Another would be to determine features that are the most helpful to recognize categories, in other words dimension reduction (using a technique such as PCA).

[1] This is not entirely true, if something shocking happens and it “it is all over the news” then date could play a role in weighting some category more.

2. Exploratory Data Analysis

2.1 Data Wrangling

As we are dealing with NLP, it is necessary to transform text into numbers. First of all we want to preprocess our text data. First we get rid of punctuation and we are using TweetTokenizer to create tokens. Other hyperparameters that we are using are stopwords.

Two approaches would be used : using nltk library and SpaCy library. Text features would be transform to TFIDF matrix.

All text features are labelled. Originally data contained 41 categories. There are some labels that are almost the same and were merged as the same category. Such as:

- 'THE WORLDPOST', 'WORLDPOST'
- 'ARTS' ,'CULTURE & ARTS', 'ARTS & CULTURE'
- 'PARENTING’, 'PARENTS'
- 'STYLE', 'STYLE & BEAUTY'
- 'COLLEGE', 'EDUCATION'
- 'TASTE', 'FOOD & DRINK'

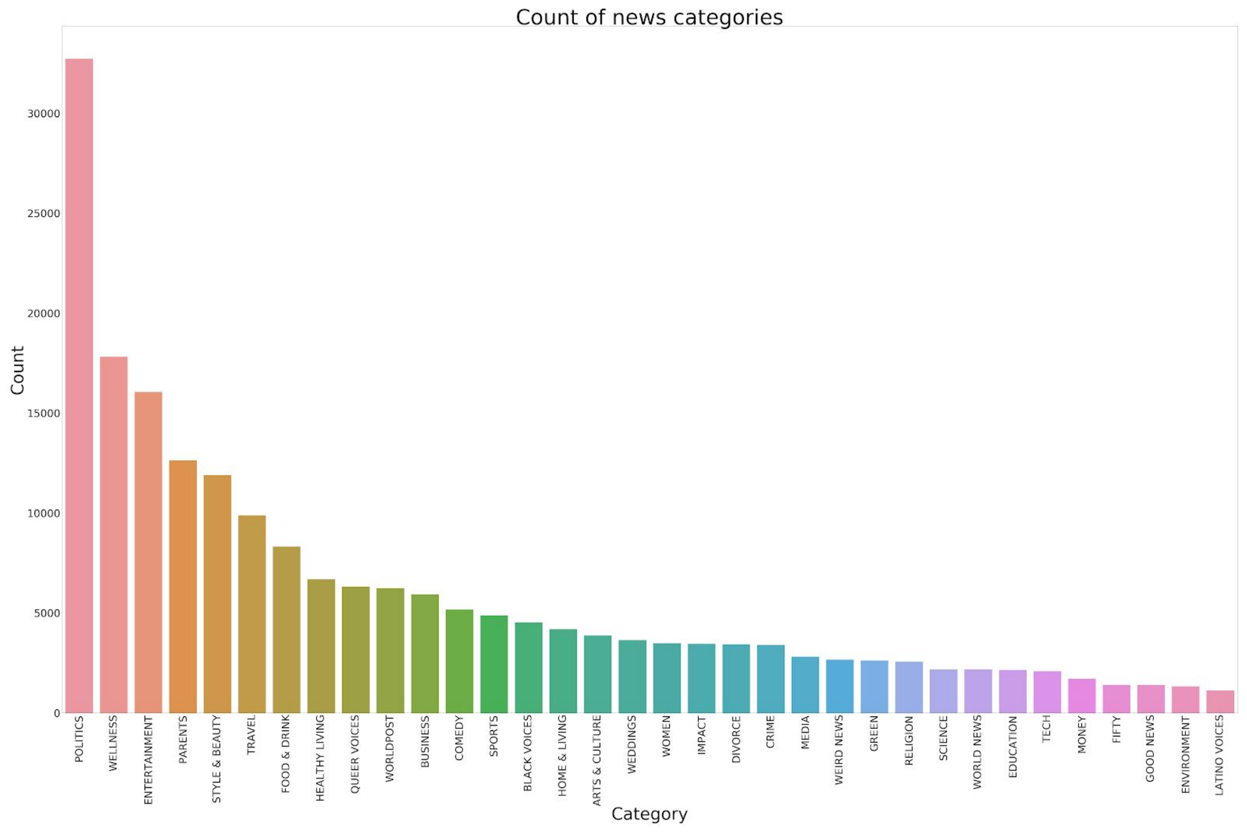
This resulted to have 34 categories instead of 41. Categories are not equally represented which leads to solving multi-classification problem with imbalanced dataset.

2.2 Data Storytelling

Let’s take a look on those categories:

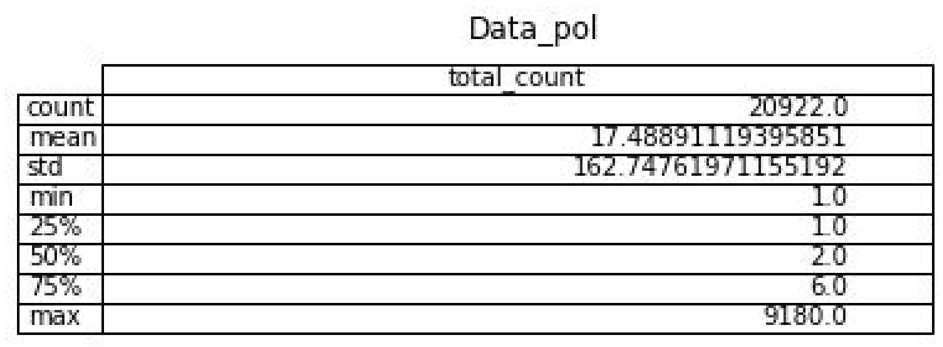
Category	Occurrences
POLITICS	32739
WELLNESS	17827
ENTERTAINMENT	16058
PARENTS	12632
STYLE & BEAUTY	11903
TRAVEL	9887
FOOD & DRINK	8322
HEALTHY LIVING	6694
QUEER VOICES	6314
WORLDPOST	6243
BUSINESS	5937
COMEDY	5175
SPORTS	4884
BLACK VOICES	4528
HOME & LIVING	4195
ARTS & CULTURE	3878
WEDDINGS	3651
WOMEN	3490
IMPACT	3459
DIVORCE	3426
CRIME	3405
MEDIA	2815
WEIRD NEWS	2670
GREEN	2622
RELIGION	2556
SCIENCE	2178
WORLD NEWS	2177
EDUCATION	2148
TECH	2082
MONEY	1707
FIFTY	1401
GOOD NEWS	1398
ENVIRONMENT	1323
LATINO VOICES	1129

Here you can see histogram of previous mentioned categories and their occurrences through histogram graph:



Let’s look closer on category ‘politics’ (category with the highest number of articles).

How many unique words does the feature ‘headline’ contain? (after preprocessing).



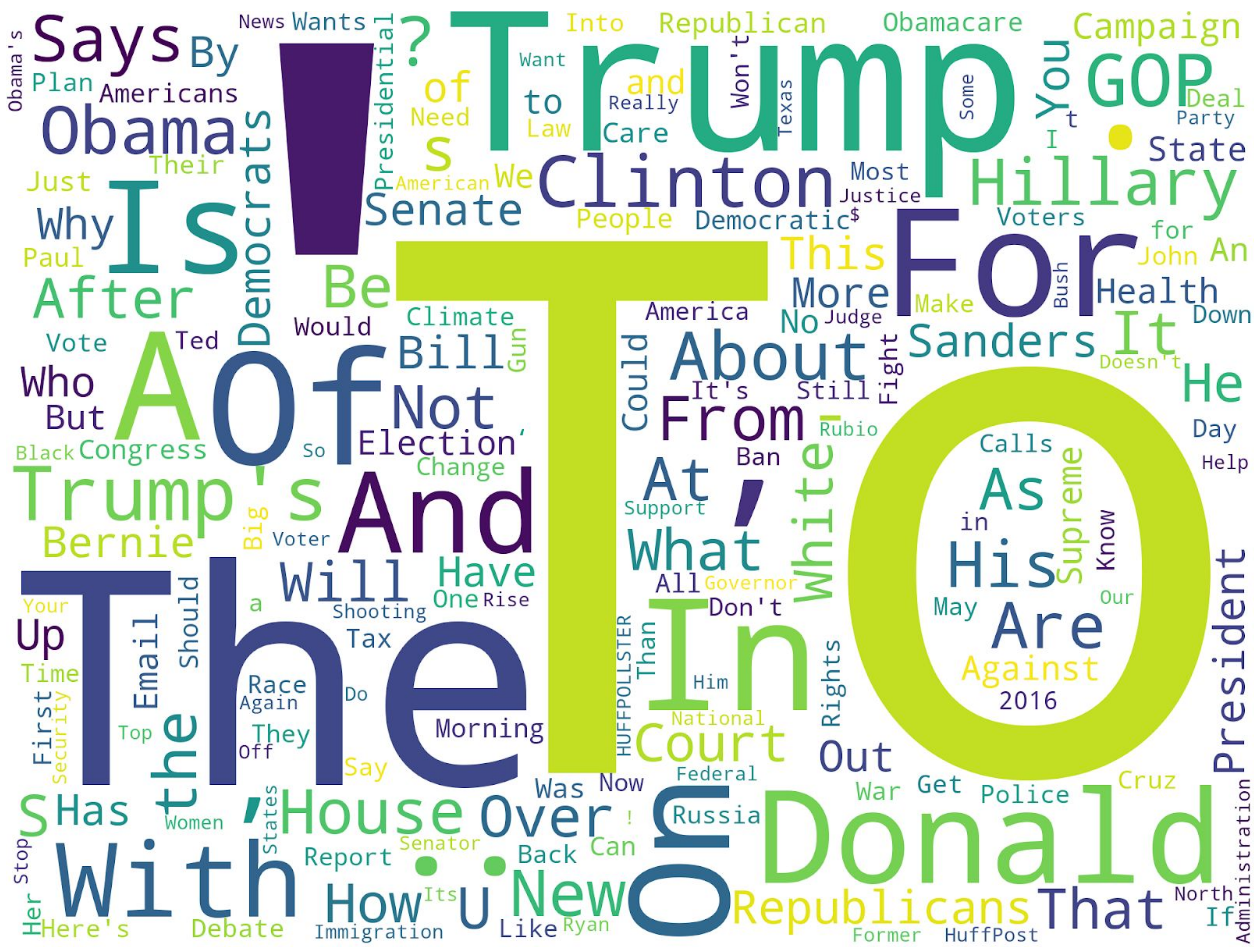
Data_pol

	total_count
count	20922.0
mean	17.48891119395851
std	162.74761971155192
min	1.0
25%	1.0
50%	2.0
75%	6.0
max	9180.0

Just for category politics we have 20922 unique tokens (and we have 34 categories!). Headline contain emojis (e.g. tacos emoji). This tokens were counted after our preprocessing. Unique words statistics table shows us, that only a small portion of dataset (25%) is occurring more than six times and maximal frequency(=occurrence) is 9180.

token	total_count
'	9180.0
To	9168.0
The	8252.0
Trump	7094.0
,	5558.0

Springboard



Note: How to understand WordCloud? Feature color you should omit entirely. The only relation is that the bigger font size the more frequent word is.

2.3 Inferential Statistics

In our data set of News we have 43 categories. We want to find out, whether some features are more important than others in order to predict whether it belongs to one category and the others. We are comparing in this example categories ‘Politics’ and “Media’. For the sake of demonstration we will pick just one word ("Donald").

Firstly, we need to find out whether the CLT (Central Limit Theorem) applies:

1) We have 32739 rows for ‘Politics’ and 2815 rows for ‘Media, so N is large enough
2) We assume that independent condition is satisfied as well

We state our null and alternative hypotheses and then we test them. We set $\boldsymbol{\alpha}$ = 0.5

$\mathbf{H_0}$: $\mu_{politics} = \mu_{medial}$

$\mathbf{H_A}$: $\mu_{politicsl} \neq \mu_{medial}$

In this Jupyter Notebook you can find details about testing our hypothesis. We here conclude our findings.

Our 95% confidence interval is <0.011909, 0.0316317>. And therefore we can reject the null hypothesis. This is a result we would expect to see the word 'Donald' more often in 'Politics' then in 'Media' category.

3. Modelling

Our goal is to create a model which would be accurately predicting categories (labels) of the new articles (aka “News”). This is obviously a multi classification problem with imbalanced dataset. Therefore we need to have balanced dataset.

We will explore two NLP techniques and compare their results. The first one is using TFIDF matrix (NLP of the headline and description), the second one is using word vectors as features.

3.1. Data Preprocessing

1. **Data Splitting**

   We split data 80:20 (80% train data, 20% test data)
2. **Text**

   Headline + short_description, just headline, just short_description

Springboard

A) We proprocess text by applying TweetTokenizer(), then transforming to the TFIDF matrix, and apply one more transformation - PCA (principal component analysis) to reduce the dimensions of the matrix (dimension is high, as unique words from corpus/messages are features).

B) We proprocess text by applying TweetTokenizer() for TFIDF matrix case, and using spaCy tokenization for word vectors and then use pre trained word vectors (from spaCy library “en_core_web_lg”). To create one vector per document we sum them up.

3. **Weighting**

We are handling imbalance dataset and therefore we set in each classifier parameter class_weight='balanced'.

We use undersampling technique (random sampling) as other option.

4. **Scaling and feature selection**

TFIDF matrix can be reduced by PCA (dimension reduction aka feature selection). Scaling is not necessary in this case.

5. **Additional Hyperparameters**

- TFIDF matrix or using word vectors
- classifier
- parameters of classifier
- with TFIDFmodel: choosing number of samples per category,
  PCA, number of components,
  TFIDF, max document frequency, ngram_range.

6. **Evaluation**

Data set is imbalanced data set with 34 categories.

First, let’s talk about binary classification to show why accuracy score is no good metric:

sometimes you can have extremes such as having 1-2% of “negative” value (complainants, cancellation of flight, etc). If we were using a guess for our predictions being positive values, we would gain accuracy score 98-99%! However, we want not only to have a high accuracy score, we want to have good precision and recall. This means we want to have good balance between false positives and true negatives (misclassified spam as hams and vice versa). F1 score is the suitable option when one needs to have a balance between Precision and Recall AND has an imbalanced data set.

Now back to our multi-classification problem. In that case we can still use f1 score as a metric for evaluation of a model. From Scikit-learn is f1 score documentation:

“The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal.

f1_score*(y_true,y_pred,labels=None,pos_label=1,average=’binary’,sample_weight=None)*

In the multi-class and multi-label case, this is the average of the F1 score of each class with weights depending on the average parameter.

average : string, [None, ‘binary’ (default), ‘micro’, ‘macro’, ‘samples’, ‘weighted’]
This parameter is required for multiclass/multilabel targets. If None, the scores for each class are returned. Otherwise, this determines the type of averaging performed on the data:
‘Binary’:
Only report results for the class specified by pos_label. This is applicable only if targets (y_{true,pred}) are binary.
’Micro’*:*
Calculate metrics globally by counting the total true positives, false negatives and false positives.

*‘**Macro**’:*
Calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account.
*‘**Weighted**’:*
Calculate metrics for each label, and find their average weighted by support (the number of true instances for each label). This alters ‘macro’ to account for label imbalance; it can result in an F-score that is not between precision and recall.
*‘**Sample**’:*
*Calculate metrics for each instance, and find their average (only meaningful for multilabel classification where this differs from **accuracy_score**).”*

More about f1_score and its details can be found for example here: https://pdfs.semanticscholar.org/1d10/6a2730801b6210a67f7622e4d192bb309303.pdf

Therefore as a metrics f1 score will be used with parameter average=’micro’.

Hyperparameters:

Text transformation: TFIDF matrix, word vectors (documents vectors)

Logistic Regression, defaults: solver='liblinear', penalty='l1', C=1.0

Solver = {‘liblinear’, ‘newton-cg’}

Penalty = {‘l1’, ‘l2’}

C = {0.1, 0.5, 1.0}

Dimension reduction: PCA={100, 500, 1000}

TFIDF: max_df = {1.0, 0.9, 0.8, 0.7}

stop_words = {True, False}

ngram_range = {(1,1), (1,2)}

Undersampling: number_of_samples = {100, 500, 1000, 1129, 1500, 1750}

Logistic Regression (Scikit learn documentation):

“In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the ‘multi_class’ option is set to ‘ovr’, and uses the cross-entropy loss if the ‘multi_class’ option is set to ‘multinomial’. (Currently the ‘multinomial’ option is supported only by the ‘lbfgs’, ‘sag’, ‘saga’ and ‘newton-cg’ solvers.)”

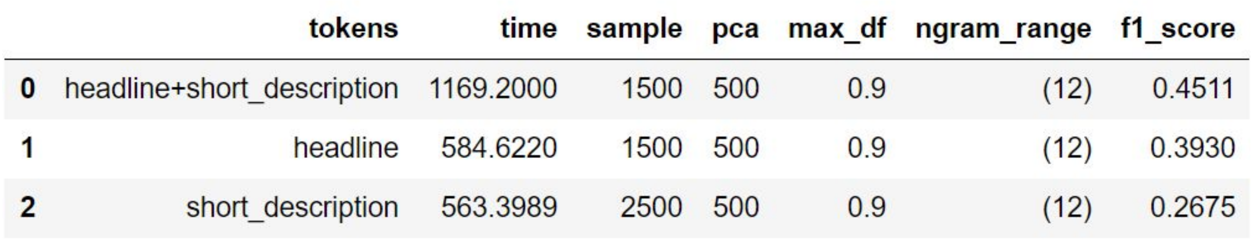
In this work it is not used solvers ‘lbfgs’ and ‘saga’ as they do not converge well. The ‘newton-cg’, and ‘sag’ solvers support only L2 regularization with primal formulation, or no regularization.

4. Results and Discussion

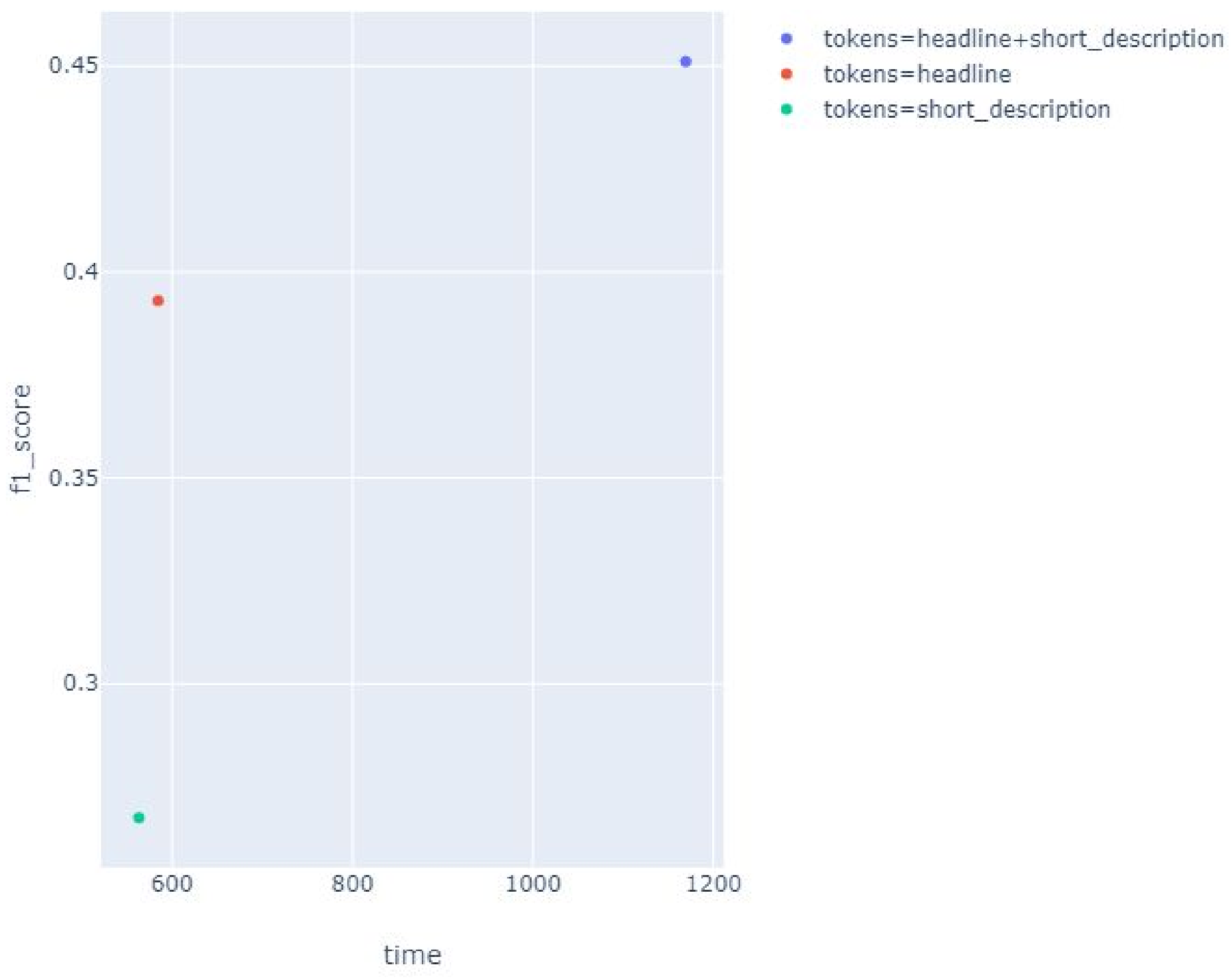
Jupyter Notebook can be found here:
https://github.com/bkarolina/DS-Projects-Springboard/blob/master/News_project_backup.ipynb

Firstly we take a look whether it's beneficial to keep both NLP feature (‘headline’ and “short description”). We try some simple model and look on f1_score result. All plots in this report include “time” on the x-axis. By “time” we mean training time of the model.

Model details: sample size=1500, PCA=500, ngram_range: (1,2), max_df: 0.9, stop_words: yes.



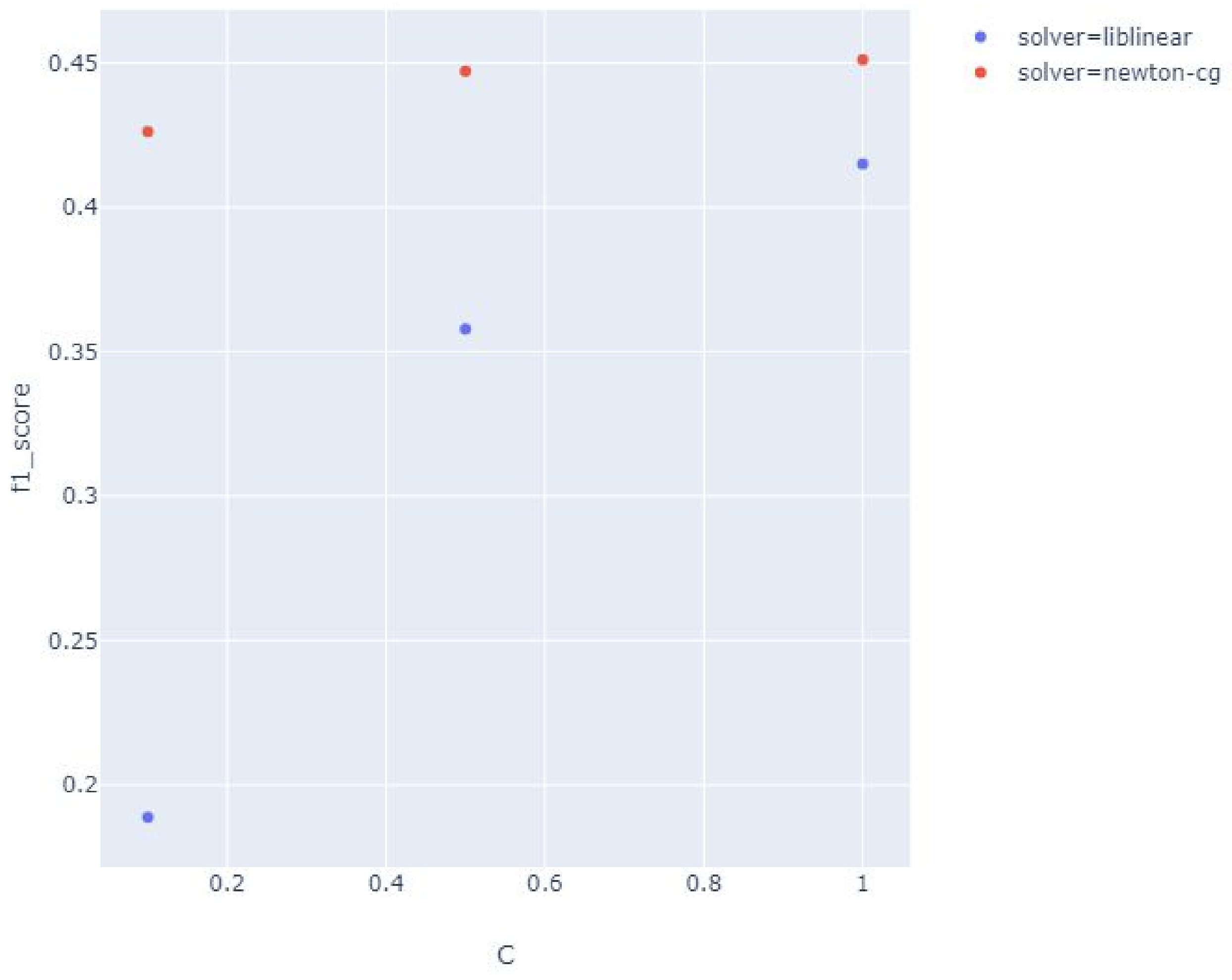
	tokens	time	sample	pca	max_df	ngram_range	f1_score
0	headline+short_description	1169.2000	1500	500	0.9	(12)	0.4511
1	headline	584.6220	1500	500	0.9	(12)	0.3930
2	short_description	563.3989	2500	500	0.9	(12)	0.2675



Now on we will use as features both of them, “headline” and “short description” as well. In the following graph you can see the result for different solvers (“liblinear”, “newton-cg”).

Model details: sample size=1500, PCA=500, ngram_range: (1,2), max_df: 0.9, stop_words: yes.

	C	f1_score	solver
0	1.0	0.415000	liblinear
1	0.5	0.357843	liblinear
2	0.1	0.188784	liblinear
3	1.0	0.451078	newton-cg
4	0.5	0.447157	newton-cg
5	0.1	0.426176	newton-cg

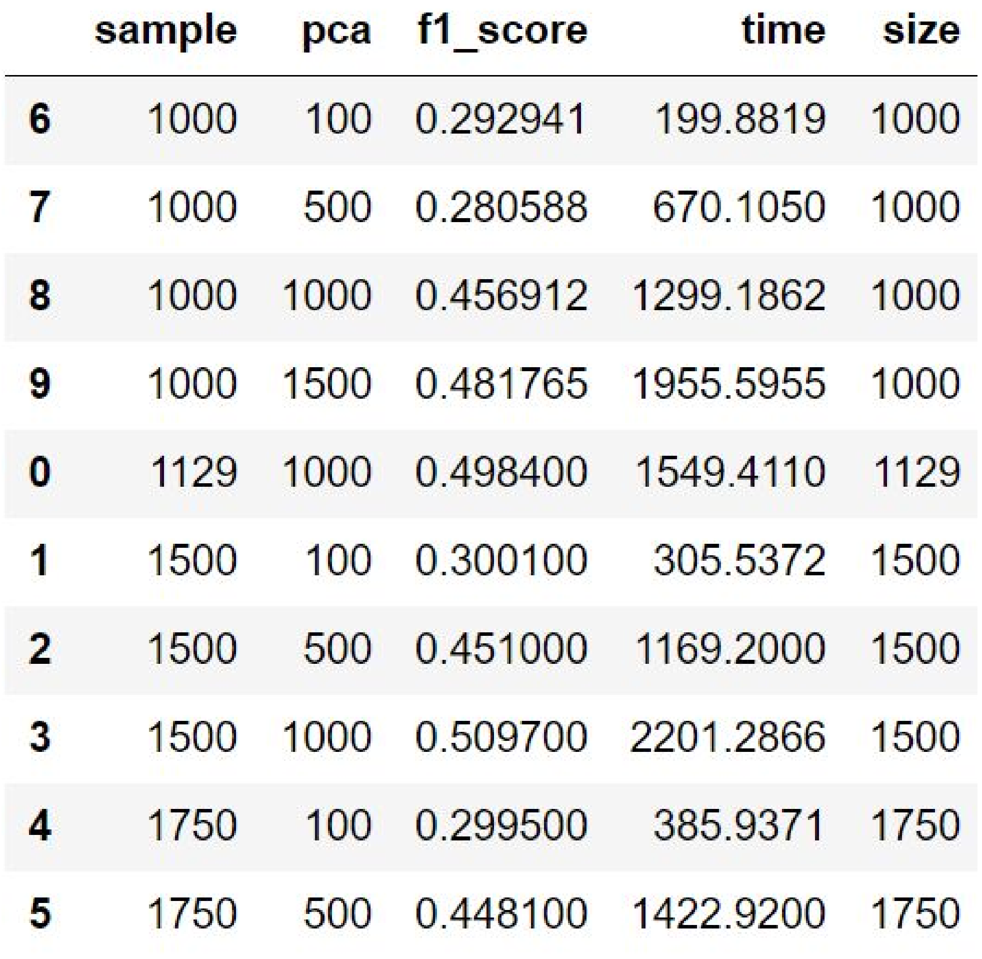


As a result we decided to set solver=”newton-cg” for further investigations too reveal influence of other hyperparameters.

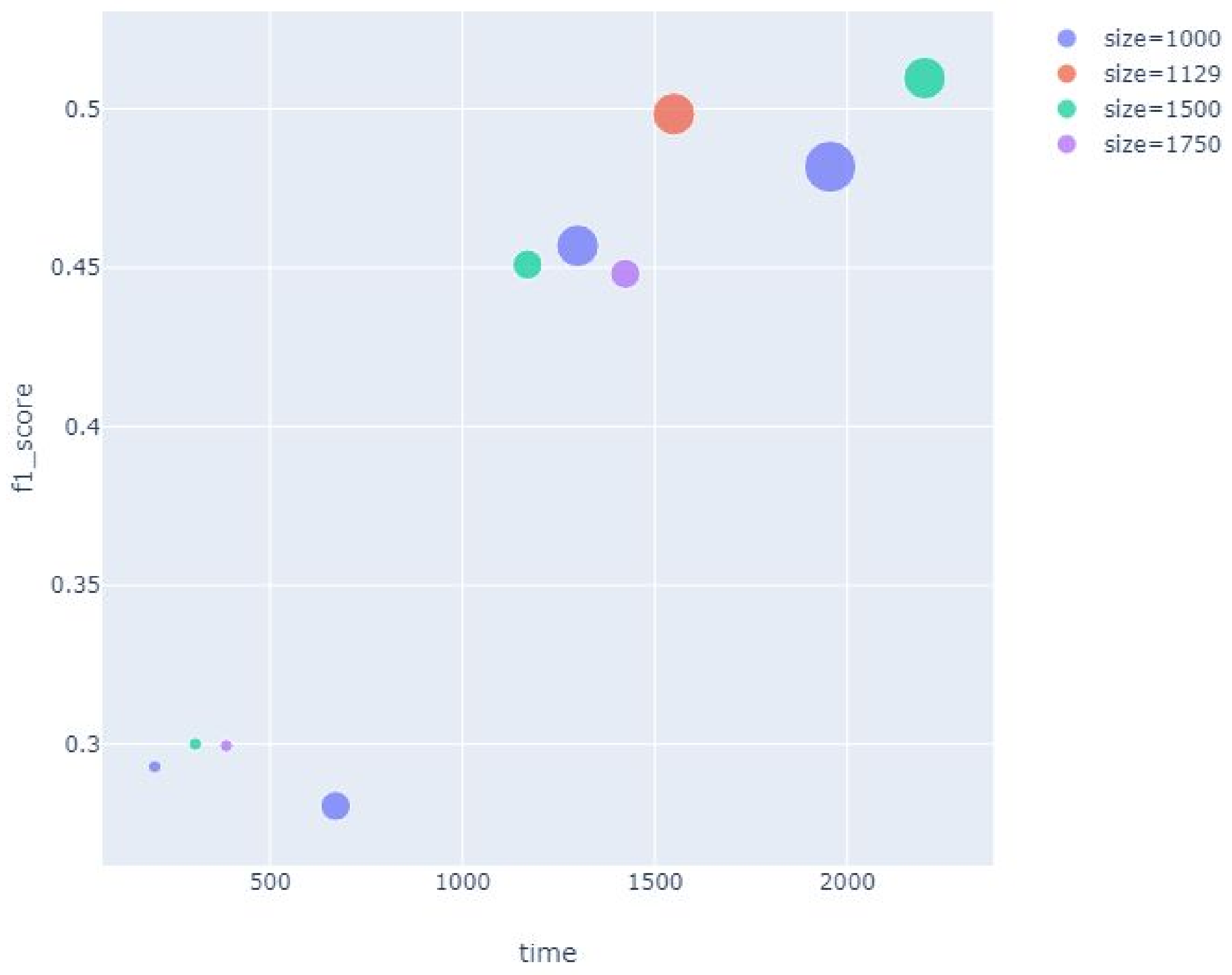
Other models differs in number of samples per category and PCA.

Model details: sample size=1500, ngram_range: (1,2), max_df: 0.9, stop_words: yes, solver=”newton-cg”.

PCA number of components is associated with a color in the graph (the bigger size the bigger PCA). PCA = {100, 500, 1000}.

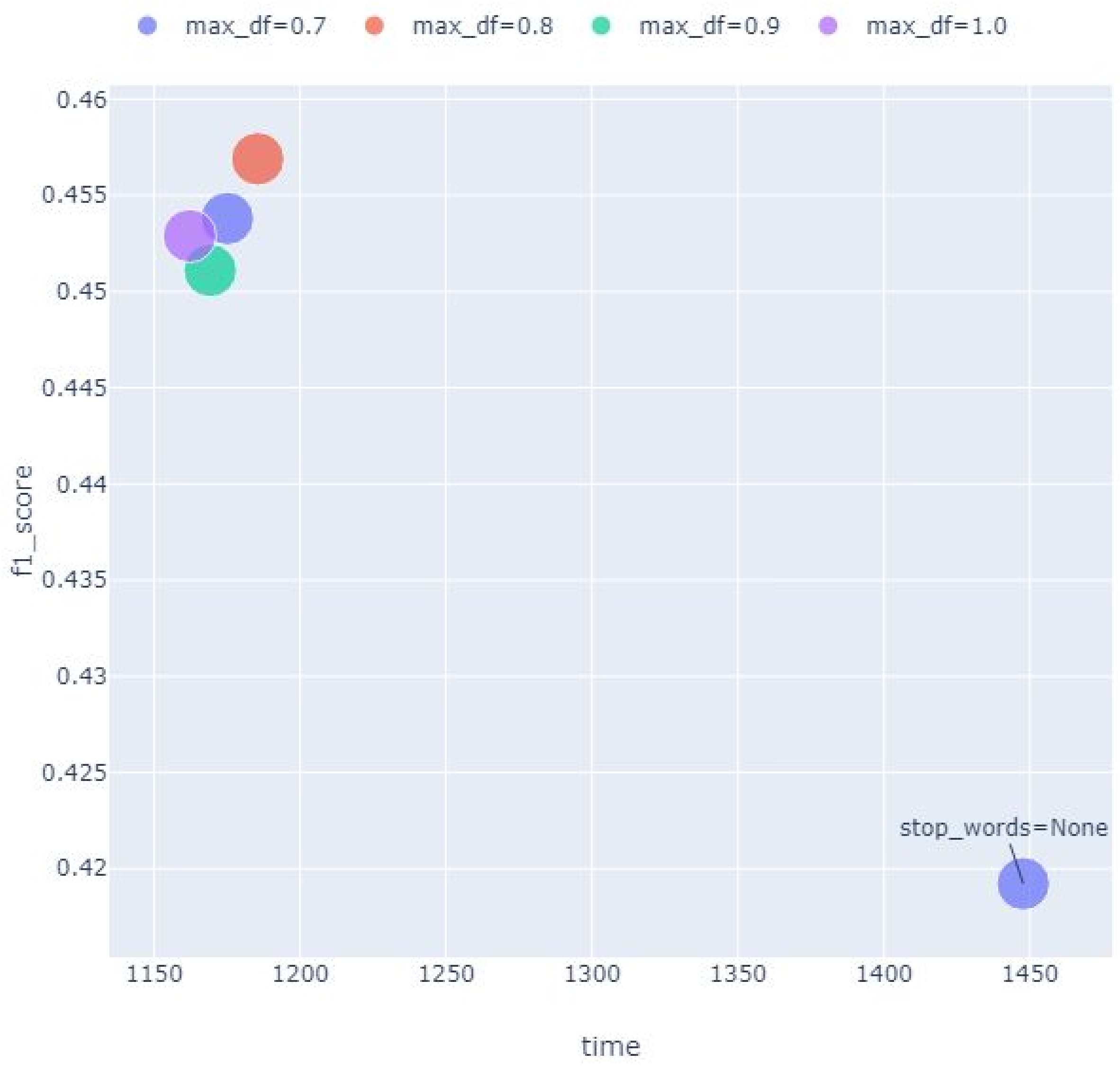


	sample	pca	f1_score	time	size
6	1000	100	0.292941	199.8819	1000
7	1000	500	0.280588	670.1050	1000
8	1000	1000	0.456912	1299.1862	1000
9	1000	1500	0.481765	1955.5955	1000
0	1129	1000	0.498400	1549.4110	1129
1	1500	100	0.300100	305.5372	1500
2	1500	500	0.451000	1169.2000	1500
3	1500	1000	0.509700	2201.2866	1500
4	1750	100	0.299500	385.9371	1750
5	1750	500	0.448100	1422.9200	1750

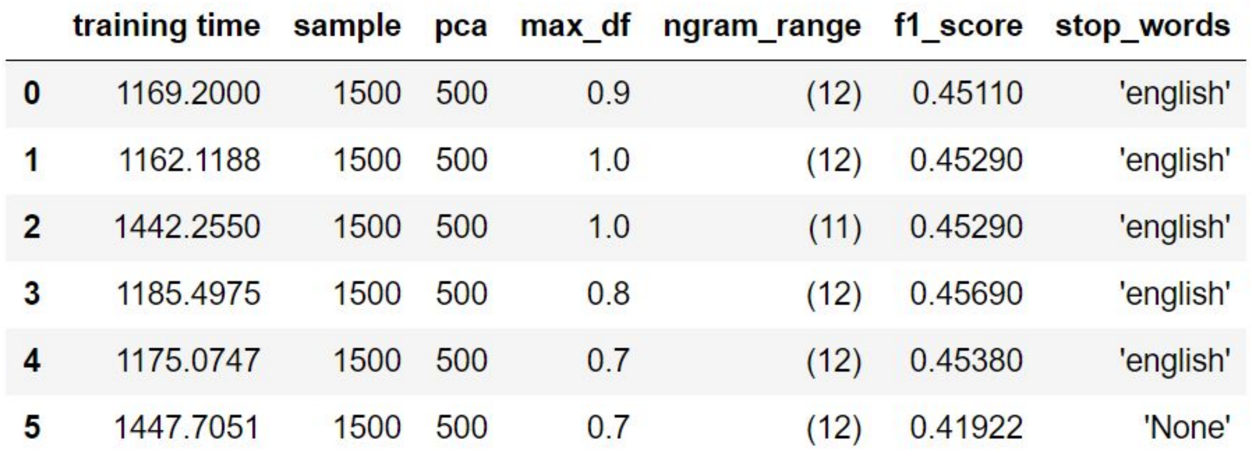


Here is clear relation - with bigger sample size and bigger PCA, we have better results. However, runtime (“time”) is longer as well.

Following graph shows how hyperparameter max_df is changing f1 score.



Max_df is slightly changing our f1_score (comparing to other hyperparameters) and it does not play such a high role. When we try to change hyperparameter stop words to "no", our result drop significantly.



	training time	sample	pca	max_df	ngram_range	f1_score	stop_words
0	1169.2000	1500	500	0.9	(12)	0.45110	'english'
1	1162.1188	1500	500	1.0	(12)	0.45290	'english'
2	1442.2550	1500	500	1.0	(11)	0.45290	'english'
3	1185.4975	1500	500	0.8	(12)	0.45690	'english'
4	1175.0747	1500	500	0.7	(12)	0.45380	'english'
5	1447.7051	1500	500	0.7	(12)	0.41922	'None'

That were results for using transformed NLP features as TFIDF matrix. Our next approach is tokenize NLP feature with spaCy and "translate" tokens to word vectors. For that we need to download 'en_core_web_lg' (small does not contain word vectors). After "translating tokens to word vector, we create document vector by summing word vector within each document (=row). As an outcome we have matrix having only 300 features (+ 'category' as the target). We train our document matrix with the best mode Logistic regression we had with TFIDF matrix.

Transformation of each NLP feature ("headline" and "short_description" took about an hour on laptop).

Note: *Our computation is performed on a subset of the original dataset (first 40,000 rows). Therefore we are predicting not 34 categories, just 27.*

Finally, let's look on comparison of models using TFIDF matrix and word/documents vectors.

Model details: LogisticRegression(class_weight='balanced', C=1.0, solver = 'newton-cg', multi_class='auto').

	F1_score	**Runtime**
TFIDF	0.4306	309.1294
Document matrix	0.4676	490.0666

4.1. Future Work

- Sample size (working in batches to manage oversampling)
- Use neural networks with word vectors
- Use different word vectors

4.2. Conclusion

In this Capstone Project we demonstrated the techniques in NLP (TFIDF, PCA, word/document vectors) on News from HuffPost data set.

We found out that our best model was using word vectors (spaCy), with f1_score = 0.4676 (subset of the original data set).

Model details: LogisticRegression(class_weight='balanced', C=1.0, solver = 'newton-cg', multi_class='auto').