Final Report
Lingwei, Xiaorui
March 15th, 2016

Introduction

Everyday media spins out a tremendous amount of information that fills our browsers and shapes our views. With materials produced by professionals and amateurs alike, it is increasingly difficult for readers to distinguish subjective opinions and factual remarks just by a quick scan at the news articles. By analayzing the features of the words, our model classifies an article as either news or opinion. It helps the readers make a quick decision on how they might prioritize their information intake given their limited time and judgement.

Progress From Midterm

- Data collection:
    - Changed news sources: Our model might be reflecting the differences between the writing styles of USAToday and FactCheck, rather than the real differences in facts and opinions. So we replaced FactCheck articles with articles in the USA TODAY news section to control for the site specific factors.
    - Added noise: To test the robustness of our models with regard to the selection of websites, we scraped 50 articles from WashingtonPost's opinion and news sections to add in noise to the corpus.
- Preprocessing:
    - The bags of words generated from the html objects are not clean enough, containing JavaScript codes. The get_words() function needs to be revised and tested. So we revisited the function to exclude extraneous scripts and words.
    - To get a better sense of the overall difference between the two sets of articles and to exclude the outliers, we visualized some important summary statistics.
- Feature Generation:
    - WordNet: Based on a set of "opinion seed words" that can be used to express positive or negative sentiment, we generated a set of opinion indicators by collecting words that are similar to these seed words using nltk WordNet.
    - POS tags: We considered two different ways of integrating part-of-speech tags into our feature space: using the individual tags versus using word-tag pairs.
    - Stopwords: Originally, we exclude all the stopwords from the model. However, news and opinion articles may use some stopwords differently, like

pronouns (I, he, she, we, they, etc) and negation (not). We evaluated how the inclusion of stopwords affects model performance.
    - Other indicators: In addition, we added some other indicators to the feature space, including the proportion of words that have been used by news and opinion corpora, proportion of stop words, and proportion of opinion indicators in the article. To account for different scales and densities of these variables, we used Python's standard scaler to standardize the feature arrays before feeding them to the models.
- Feature Selection: Since the total number of features is not very large, we did not implement feature selection.
- Model:
    - Parameter sweep: We used the grid_search function provided by sklearn to generate the best combination of parameters for each model.
    - Evaluation
        - We generated a learning curve to check whether the model is overfitting or underfitting.
- User Interface:
    - We prettified our user interface.
    - We saved training time by saving our model to a pickle file. When the interface is restarted, it only needs to load the pickle file rather than to retrain the model.

Pipeline Summary

We combined natural language processing and machine learning to classify an input article as either fact or opinion.

- Building Corpus

Initially setting out to use factcheck.org for factual content and articles from the opinion section from USA Today, we found that although our model performs well, it was reflecting the distinctive writing style of the two sites rather than the content and features of the language. Thus, we used news and opinions from USAToday and added in noises by adding in more opinion and factual articles from WashingtonPost to account for the probability that particular websites might have very different styles in its sections.

- Feature generation
    - TF-IDF: The product of two statistics, term frequency and inverse document frequency.
    - Parts of speech tag: Many researches have shown that a large number of adjectives indicate a high probability of the text being subjective.

    - Opinion indicator seed word: This include both positive and negative adjective words such as “good”, “awesome”, “bad” as well as verbs such as “think”, “believe”.
    - Stopwords: The proportion of stopwords in an article.
    - Ambiguous words: The proportion of words that appear in both news and opinion articles.
    - Bigrams
- Summaries and models:

Summary Statistics

1) Basic statistics

- num_words: the number of words in each article including stop words. We can find that the distributions of article length are pretty similar between news and opinion pieces, while the length of news articles has a larger variation than opinion articles.
- p_unique_words: the number of unique words divided by the number of words in each article. Opinion articles seem to use more non-repetitive words in a single piece.
- p_lowercase: the proportion of words in lowercases in each article.
- p_uppercase: the proportion of words in uppercases in each article. News articles tend to use more uppercase words than opinion articles.
- p_titlecase: the proportion of words whose first letter is uppercase and other letters are lowercases in each article. News articles tend to use more titlecase words than opinion articles.
- average_word_length: average number of characters in a word in each article. The distributions are pretty similar.
- p_stopwords: the proportion of words that are stop words in each article. Opinion articles tend to use more stopwords than news articles.

News:

	num_words	p_unique_words	p_lowercase	p_uppercase	p_titlecase	average_word_length	p_stopwords
count	125.000000	125.000000	125.000000	125.000000	125.000000	125.000000	125.000000
mean	732.520000	0.499234	0.805622	0.020797	0.182015	4.841143	0.409268
std	573.330108	0.083805	0.079914	0.022404	0.073228	0.260336	0.044713
min	81.000000	0.287686	0.207650	0.002421	0.082982	4.225434	0.147541
25%	362.000000	0.452113	0.784000	0.009387	0.142857	4.675549	0.390390
50%	636.000000	0.495105	0.813605	0.015848	0.171336	4.848249	0.410615
75%	887.000000	0.552727	0.848901	0.025581	0.196721	5.022684	0.433735
max	3622.000000	0.719626	0.915612	0.172840	0.699454	5.632308	0.492267

Opinion:

	num_words	p_unique_words	p_lowercase	p_uppercase	p_titlecase	average_word_length	p_stopwords
count	125.000000	125.000000	125.000000	125.000000	125.000000	125.000000	125.000000
mean	678.328000	0.532791	0.838758	0.016158	0.152403	4.885109	0.431480
std	264.792517	0.069707	0.037696	0.010771	0.036325	0.249836	0.033809
min	97.000000	0.390169	0.733945	0.000000	0.077430	4.210909	0.336377
25%	487.000000	0.489305	0.818805	0.008000	0.127726	4.728495	0.412054
50%	751.000000	0.516358	0.842105	0.015306	0.151306	4.890937	0.433030
75%	834.000000	0.565966	0.864542	0.022849	0.173112	5.011799	0.455843
max	1417.000000	0.748718	0.922570	0.047273	0.244709	5.529680	0.506887

2) Frequent words

We used two variables to measure word frequency: total number of occurences of a word in the corpus and the number of documents with the word. We reported the top 20 most frequent words in news, opinion and both types of articles. All stopwords are excluded.

We can find that some words appear frequently in both news and opinion articles, like “one” and “two”. Although most frequent words are words used commonly in English, there are still differences between the word distribution of news and opinion articles.

News:

Total Number of Occurrences		Number of Documents with the Word	
said	819	said	110
zika	265	one	79
people	234	also	78
state	225	people	76
one	210	year	73
new	195	new	72
trump	192	would	67
virus	183	two	62
police	179	first	62
would	176	state	61
rubio	170	time	61
year	169	last	55
also	153	may	55
according	146	like	54
two	139	could	54
health	125	according	53
first	123	many	52
federal	121	even	52
could	121	day	51
time	116	three	50

Opinion:

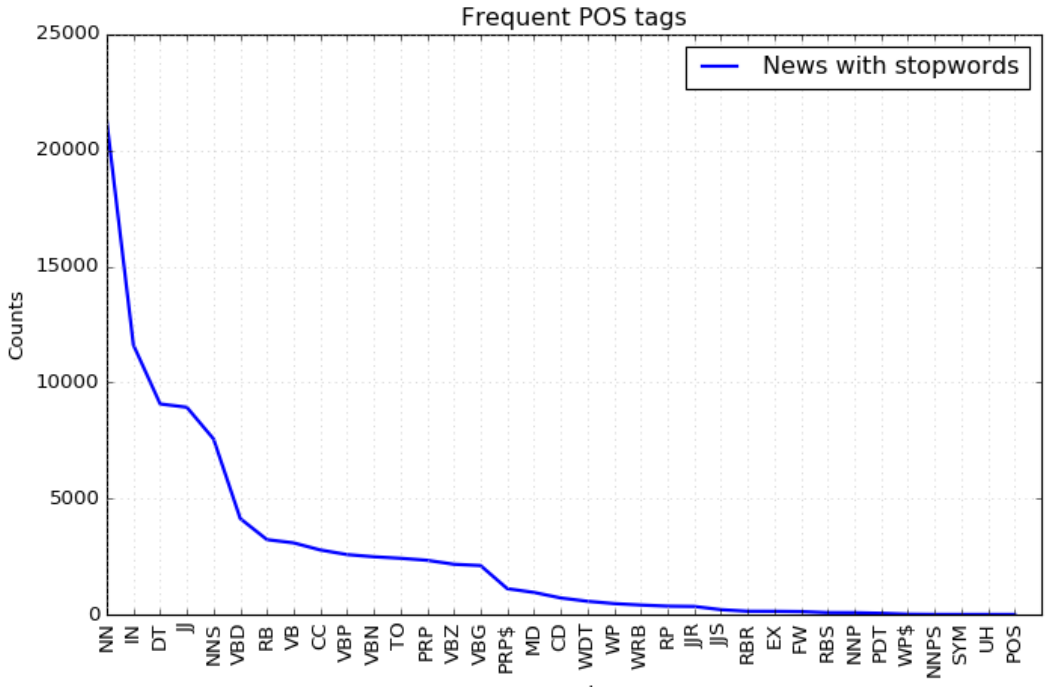
Total Number of Occurrences		Number of Documents with the Word	
would	269	one	97
one	216	would	92
president	204	read	83
trump	194	like	83
new	182	even	77
justice	180	people	76
like	179	also	74
court	176	time	73
people	159	new	72
sanders	144	opinion	71
even	140	go	69
obama	137	president	69
law	133	board	68
could	133	justice	68
year	131	year	66
many	128	look	65
clinton	124	last	65
also	121	could	65
time	120	make	65
political	117	including	65

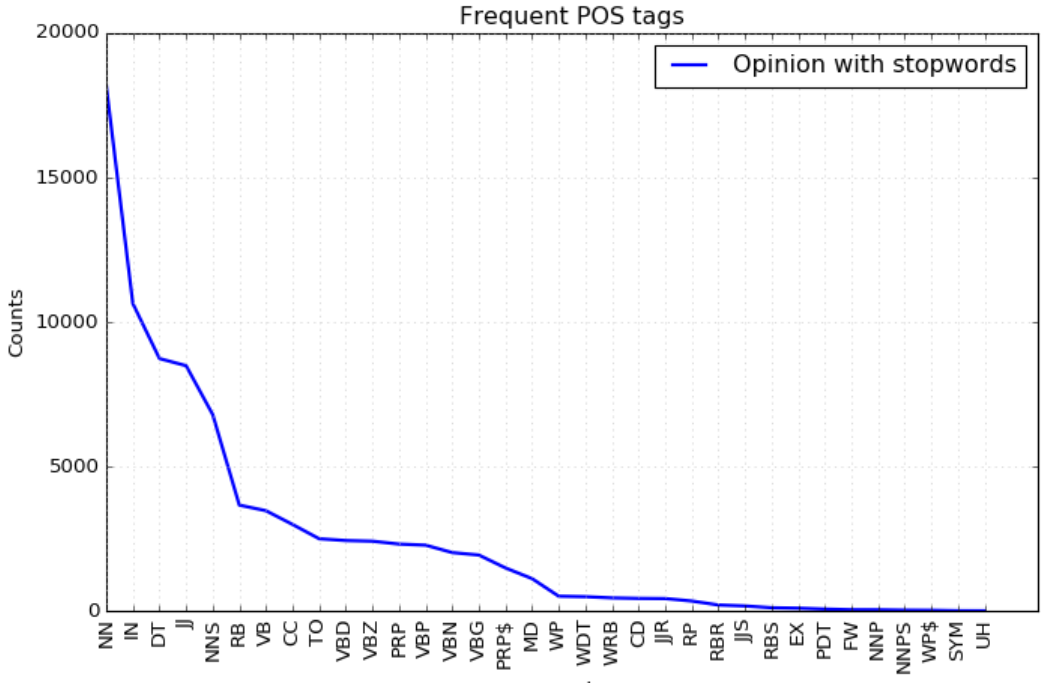
Overall:

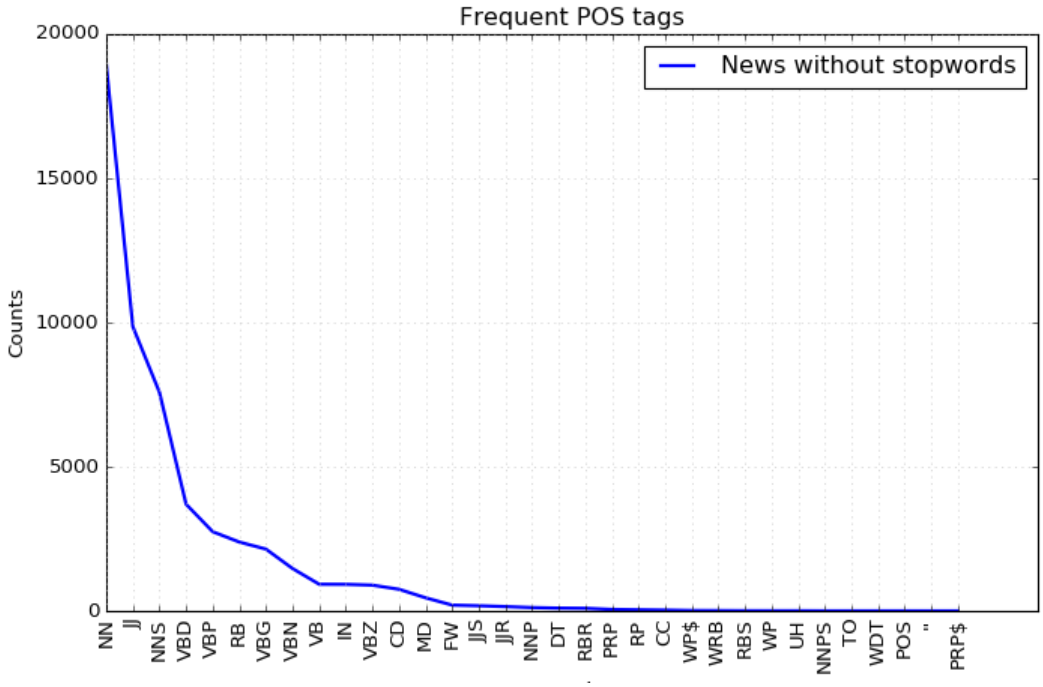
Total Number of Occurrences		Number of Documents with the Word	
said	896	one	176
would	445	would	159
one	426	said	156
people	393	people	152
trump	386	also	152
new	377	new	144
state	328	year	139
year	300	like	137
president	298	time	134
court	275	even	129

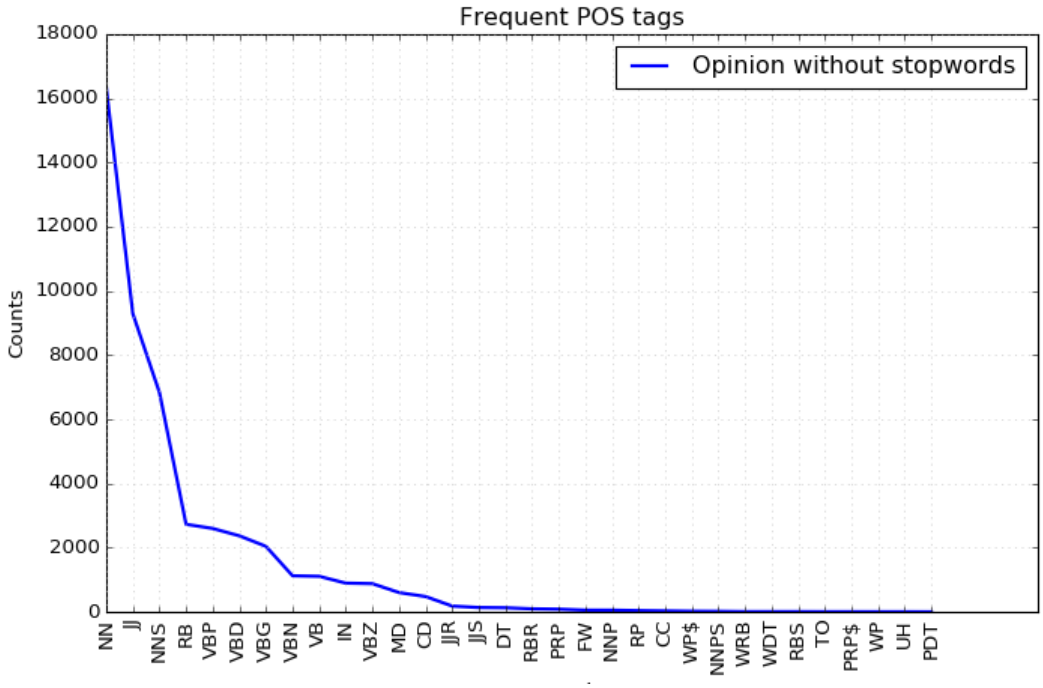
3) Frequent POS tags

We calculated the frequencies of Part-of-Speech tags in news or opinion articles with or without stopwords. The distributions of POS tags are pretty similar between news and opinion, but are very different with regard to including stopwords or not.


Frequent POS tags
News with stopwords
Counts
25000
20000
15000
10000
5000
0
NN IN DT JJ NNS VBD RB VB CC VBP VBN TO PRP VBZ VBG PRP$ MD CD WDT WP WRB RP JJR JJS RBR EX FW RBS NNP PDT WP$ NNPS SYM UH POS

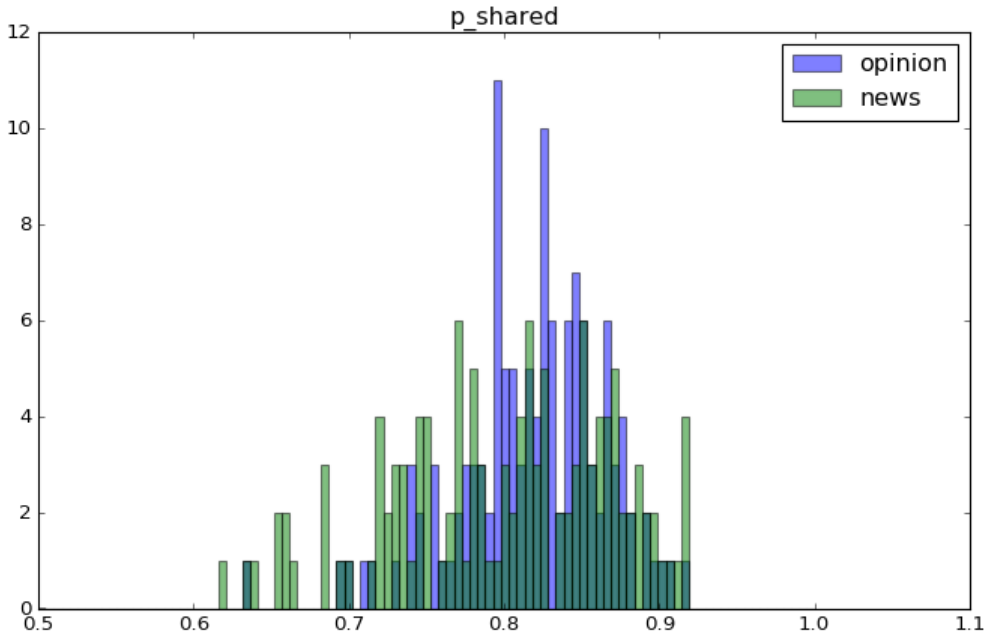

Frequent POS tags
Opinion with stopwords
Counts
20000
15000
10000
5000
0
NN IN DT JJ NNS RB VB CC TO VBD VBZ PRP VBP VBN VBG PRP$ MD WP WDT WRB CD JJR RP RBR JJS RBS EX PDT FW NNP NNPS WP$ SYM UH





4) Shared words

We also consider the specialty of words that occur in both news and opinion corpora. The figure below presents the proportion of words shared by news and opinion corpora in each article. We can find that most words used by news or opinion articles are common words that have been used by either type of articles. Opinion articles on average tend to use more shared words than news articles.



5) Words unique to each type

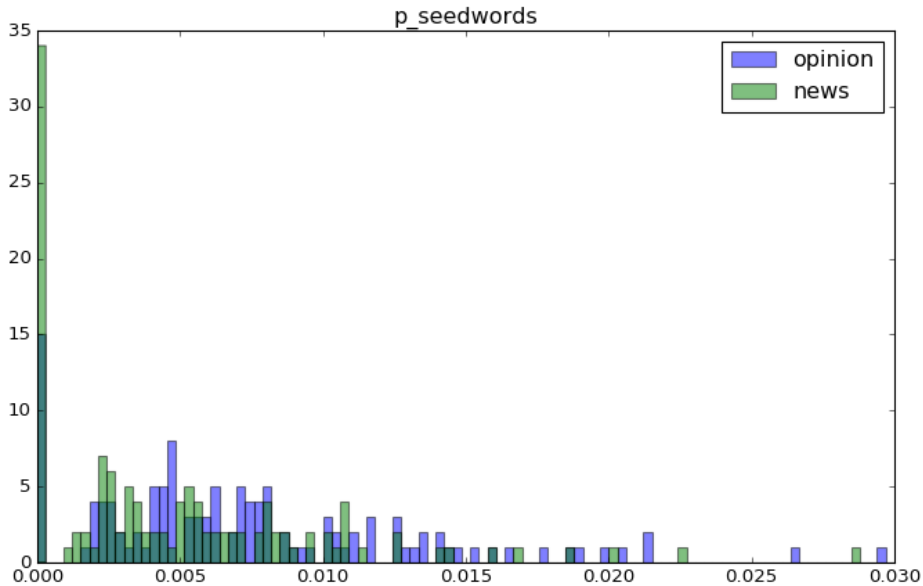
We also examined the unique words that appear in news but not opinion articles and vice versa. Here is a graph presentation of the most frequent words that are unique to either news or opinion corpus, generated by WordCloud. We can find clear differences between the words unique to news and those unique to opinion articles.

News:

Opinion:

6) Distribution of opinion seed words

We also examined the distributions of opinion seed words in news and opinion articles in our sample. The figure below shows the proportion of opinion seed words in an article. We can find that these opinion seed words appear in both opinion and news articles. Although 15 opinion articles do not contain these seed words, more than twice as many news articles do not have these seed words. Opinion articles tend to contain more seed words than news articles.



Evaluation Results & Analysis

1) The inclusion of stop words

We compared model performances with and without stopwords as part of the feature space.

- Unigram features with stop words:

	accuracy	precision	recall	f1	auc_roc	average_precision_score	train_time	test_time
LinearSVC	0.900	0.939650	0.864229	0.897644	0.905393	0.937940	0.322263	0.001048
DecisionTree	0.876	0.859040	0.898975	0.876160	0.879178	0.905007	0.123513	0.002278
RandomForest	0.856	0.827906	0.912343	0.860926	0.950612	0.954957	0.036546	0.002116
Bagging	0.900	0.880737	0.930456	0.901730	0.967724	0.968782	0.604875	0.047198
NaiveBayes	0.648	0.739315	0.480925	0.572742	0.655028	0.742120	0.058259	0.015009
KNeighbors	0.556	0.788095	0.247267	0.284758	0.680043	0.725913	0.053011	0.202013
LogisticReg	0.896	0.931317	0.864229	0.894306	0.969878	0.963950	0.323196	0.001180
Boosting	0.924	0.917000	0.929945	0.922205	0.982863	0.984940	20.264506	0.002731

- Unigram features without stop words:

	accuracy	precision	recall	f1	auc_roc	average_precision_score	train_time	test_time
LinearSVC	0.888	0.914498	0.853380	0.879672	0.887333	0.919939	0.470899	0.001231
DecisionTree	0.900	0.895168	0.903525	0.898535	0.899998	0.923347	0.114772	0.002240
RandomForest	0.856	0.845568	0.871714	0.856414	0.933628	0.946247	0.036892	0.003258
Bagging	0.916	0.904607	0.924266	0.913086	0.972035	0.968201	0.567827	0.048393
NaiveBayes	0.620	0.678323	0.447287	0.536463	0.619859	0.700805	0.066404	0.019310
KNeighbors	0.556	0.950000	0.135761	0.222779	0.648247	0.756489	0.042642	0.186393
LogisticReg	0.892	0.915831	0.862076	0.884460	0.960721	0.956522	0.327148	0.001579
Boosting	0.920	0.905265	0.933357	0.918359	0.982823	0.982814	18.105019	0.002979

We can find that models with stopwords as features perform generally better than those without stopwords, except for Linear SVC and Decision Tree. Thus, we include stopwords into our feature space.

2) Parameter selection

We implemented grid_search on combinations of a set of parameter values as below:

```
# parameter selection
model_parameters = {LSVC(): {"C": (0.1, 1.0, 10.0)},
                    RFC(): {"criterion": ("gini", "entropy"), "n_estimators":(5, 10, 15)},
                    KNC(): {"n_neighbors":(3, 5, 7)},
                    DTC(): {"criterion": ("gini", "entropy")},
                    LR(): {"C": (0.1, 1.0, 10.0)},
                    BC(): {"n_estimators": (5, 10, 15)},
                    GBC(): {"learning_rate": (0.05, 0.1, 0.3), "n_estimators":(100, 150, 200)}}
```

We implemented the the best set of parameters for each algorithm given by grid search. The parameters selected are:

```
models = {"LinearSVC": LSVC(C=0.1),
          "RandomForest": RFC(criterion='entropy', n_estimators=15),
          "KNeighbors": KNC(),
          "DecisionTree": DTC(),
          "LogisticReg": LR(),
          "NaiveBayes":NB(),
          "Bagging": BC(),
          "Boosting": GBC(learning_rate=0.3, n_estimators=200)}
```

3) Using POS tags
We compared two ways of generating features using part-of-speech tags. One is to include POS tags individually as features, the other is to include word-tag pairs. We ran models with unigrams and these two tag features respectively and compared model performances.

The results are pretty similar. For some models, individual tags perform better while for others they perform worse than word-tag pairs. Consider that word-tag pairs perform better on models with higher f1 scores than individual POS tags, we decided to use word-tag pairs.

- Individual POS tags

	accuracy	precision	recall	f1	auc_roc	average_precision_score	train_time	test_time
LinearSVC	0.904	0.923918	0.886187	0.902129	0.908277	0.935052	4.961155	0.001294
DecisionTree	0.872	0.897886	0.849644	0.866203	0.874369	0.911765	0.137598	0.002191
RandomForest	0.908	0.937217	0.877811	0.904168	0.973327	0.976168	0.049045	0.002276
Bagging	0.900	0.887982	0.926417	0.902504	0.949792	0.941968	0.531201	0.049216
NaiveBayes	0.644	0.750677	0.468984	0.560957	0.654916	0.745830	0.069603	0.020693
KNeighbors	0.504	0.691304	0.238325	0.196464	0.656634	0.774792	0.041419	0.197326
LogisticReg	0.896	0.909174	0.886187	0.895816	0.967858	0.966264	0.402228	0.002083
Boosting	0.920	0.919863	0.923560	0.917821	0.981955	0.981724	18.011451	0.003721

- Word-tag pairs

	accuracy	precision	recall	f1	auc_roc	average_precision_score	train_time	test_time
LinearSVC	0.904	0.910358	0.899980	0.901860	0.908921	0.931169	7.453310	0.001438
DecisionTree	0.872	0.865884	0.881832	0.873337	0.870630	0.903858	0.176292	0.002992
RandomForest	0.856	0.881484	0.831476	0.850513	0.946441	0.956309	0.059534	0.003123
Bagging	0.892	0.887308	0.898763	0.891836	0.962408	0.965431	0.689714	0.054049
NaiveBayes	0.592	0.720448	0.347539	0.456656	0.599162	0.695993	0.073168	0.022334
KNeighbors	0.528	0.627273	0.194992	0.239543	0.662913	0.744644	0.053261	0.249840
LogisticReg	0.916	0.918788	0.914284	0.913859	0.977138	0.973371	0.439462	0.001348
Boosting	0.932	0.956113	0.906417	0.929302	0.990504	0.991215	22.025542	0.003499

4) Model selection
We compared model performances with different sets of features.

- Unigrams

	accuracy	precision	recall	f1	auc_roc	average_precision_score	train_time	test_time
LinearSVC	0.900	0.939650	0.864229	0.897644	0.905393	0.937940	5.594639	0.001834
DecisionTree	0.884	0.862955	0.913014	0.885768	0.883341	0.909985	0.174374	0.002756
RandomForest	0.908	0.948528	0.874740	0.905486	0.968728	0.975470	0.066037	0.002912
Bagging	0.896	0.893337	0.902324	0.895329	0.965362	0.969195	0.621158	0.049912
NaiveBayes	0.648	0.739315	0.480925	0.572742	0.655028	0.742120	0.057101	0.015755
KNeighbors	0.556	0.788095	0.247267	0.284758	0.680043	0.725913	0.039948	0.180371
LogisticReg	0.896	0.931317	0.864229	0.894306	0.969878	0.963950	0.308389	0.001104
Boosting	0.928	0.934516	0.923049	0.927326	0.984974	0.986217	19.437648	0.003016

- Unigrams and bigrams

We can find that models with unigrams and bigrams outperforms those with only unigrams. Thus, we include bigrams in our final model.

	accuracy	precision	recall	f1	auc_roc	average_precision_score	train_time	test_time
LinearSVC	0.928	0.958730	0.902573	0.928531	0.931346	0.956651	24.326770	0.005808
DecisionTree	0.888	0.897493	0.871585	0.883314	0.887940	0.916539	0.836012	0.015537
RandomForest	0.880	0.888002	0.875411	0.879668	0.959762	0.963628	0.170270	0.011813
Bagging	0.908	0.893826	0.919699	0.904999	0.969739	0.973983	3.008883	0.322608
NaiveBayes	0.700	0.838567	0.497380	0.619530	0.704460	0.793973	0.345212	0.108331
KNeighbors	0.536	0.675333	0.544236	0.476781	0.562618	0.734975	0.282020	1.324990
LogisticReg	0.936	0.967778	0.909469	0.936850	0.984035	0.983669	3.104324	0.007696
Boosting	0.932	0.939268	0.920421	0.929233	0.986608	0.987291	109.724525	0.017041

- Unigrams, bigrams and POS tags

The evaluation results are pretty similar to the models with unigrams and bigrams, so adding POS tags do not significantly increase performance for most models, except for Bagging.

	accuracy	precision	recall	f1	auc_roc	average_precision_score	train_time	test_time
LinearSVC	0.920	0.965185	0.879290	0.917715	0.924168	0.954238	4.020943	0.009637
DecisionTree	0.872	0.850430	0.893508	0.871010	0.870669	0.897969	1.302327	0.019799
RandomForest	0.848	0.895614	0.800505	0.839785	0.935241	0.944617	0.208373	0.015759
Bagging	0.920	0.928983	0.904036	0.915361	0.955950	0.960266	4.577954	0.396316
NaiveBayes	0.660	0.901010	0.359144	0.507102	0.662561	0.790077	0.524992	0.159401
KNeighbors	0.508	0.792000	0.448325	0.346976	0.545862	0.770878	0.327485	1.677164
LogisticReg	0.932	0.965185	0.899980	0.930352	0.985678	0.981863	3.476007	0.008810
Boosting	0.928	0.953864	0.899520	0.923463	0.984576	0.986926	151.239506	0.018986

After comparing the performances of these models, we decided to use LogisticRegression with default parameters and the whole set of features as our final model.

5) Feature importance

The following table reports the top 20 most important features where the importance scores are evaluated by a Random Forest Classifier. The first column is the feature name, which can be either a unigram or bigram with POS tags. The second and third columns report the

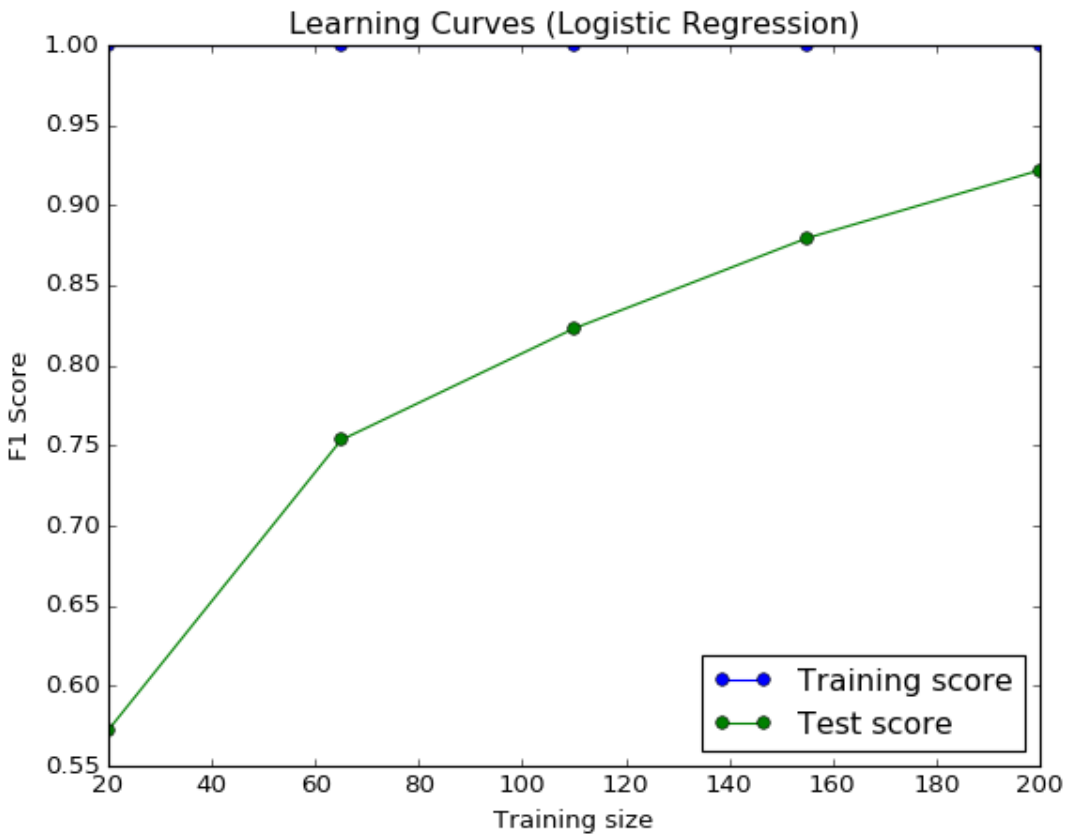
average number of occurrences of the feature in news or opinion articles, respectively. The last column reports the importance score evaluated by the model.

Feature	NEWS	OPINION	importance
publishes_nns diverse_vbp	0.0	0.376	0.025285862785862766
read_vb more_jjr	0.0	0.472	0.020932257043368162
writers_nns including_vbg	0.0	0.376	0.019461865922699363
this_dt go_nn	0.0	0.376	0.0175451000238152
like_in	0.696	1.432	0.015132767218799978
a_dt look_nn	0.008	0.424	0.012739961907871489
editorials_nns usa_vbp	0.0	0.376	0.012357029226194565
opinions_nns from_in	0.008	0.376	0.010984579185002398
tuesday_jj	0.264	0.024	0.0101236698303135
so_rb	1.84	2.104	0.009984458837734531
president_nn	0.752	1.632	0.009788934367306185
to_to	19.368	19.936	0.00952062115610884
actually_rb	0.112	0.176	0.00930847155978999
policing_vbg	0.008	0.36	0.00919992680821306
media_nns	0.136	0.744	0.008650121153292396
why_wrb	0.232	0.456	0.008344935979320912
presidency_nn	0.008	0.144	0.008225108225108215
problem_nn	0.192	0.44	0.008150813397129184
t_jj	9.28	8.664	0.007926239067218796
decided_vbn	0.016	0.112	0.007745625961801899

6) Learning curves

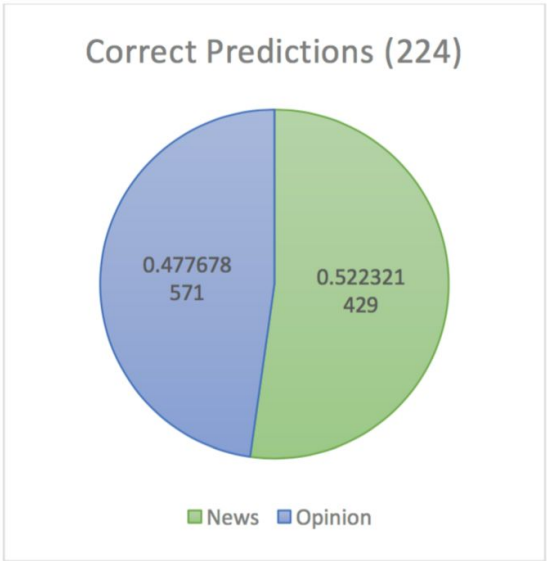
After selecting the final model, we plotted learning curves for the model to show how training and testing f1 scores change along with training size.

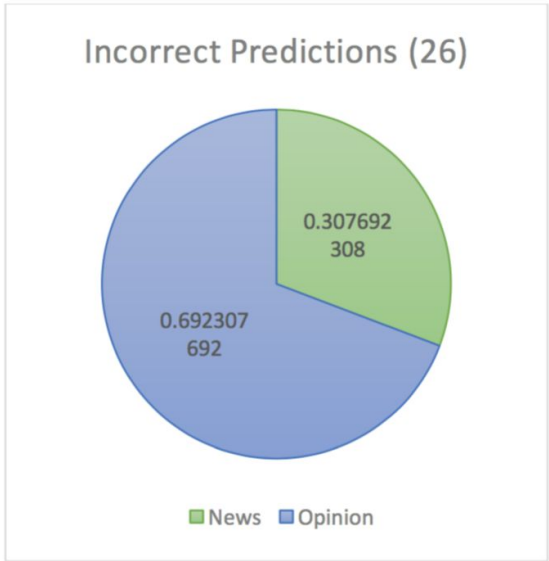
We can find that the training scores are always 1.0 regardless of training size, so there are no underfitting problems. The test scores increase along with training size, and when the training size is large enough, the gap between the two curves become acceptable. Therefore, when the training size is large enough, there are no serious overfitting problems.



7) Misclassification analysis

To analyze which articles are more likely to be misclassified by the model, we output some basic summary statistics on both the correctly and incorrectly classified groups. Among the 250 articles in the corpus, 224 were correctly classified, while 26 were incorrectly classified. As shown in the figures below, among the incorrect predictions, nearly 70% are opinion articles. Therefore, our model is more likely to misclassify opinion articles as news articles than the other way around.





The tables below show other summary statistics of these two groups. It is surprising that the incorrectly classified articles tend to be longer than those correctly classified. For other statistics, there are no significant differences between the two groups.

Correct:

	num_words	p_unique_words	average_word_length	p_stopwords	p_shared	p_seedwords
count	224.000000	224.000000	224.000000	224.000000	224.000000	224.000000
mean	686.410714	0.517220	4.857989	0.418968	0.468331	0.003493
std	425.951713	0.078464	0.256947	0.041850	0.042671	0.003232
min	81.000000	0.287170	4.224638	0.145946	0.362734	0.000000
25%	395.250000	0.466715	4.693793	0.396526	0.441783	0.001236
50%	683.500000	0.509167	4.868562	0.420320	0.468004	0.002865
75%	819.750000	0.565219	5.006169	0.446343	0.493700	0.004851
max	3632.000000	0.746193	5.629800	0.506887	0.670270	0.017134

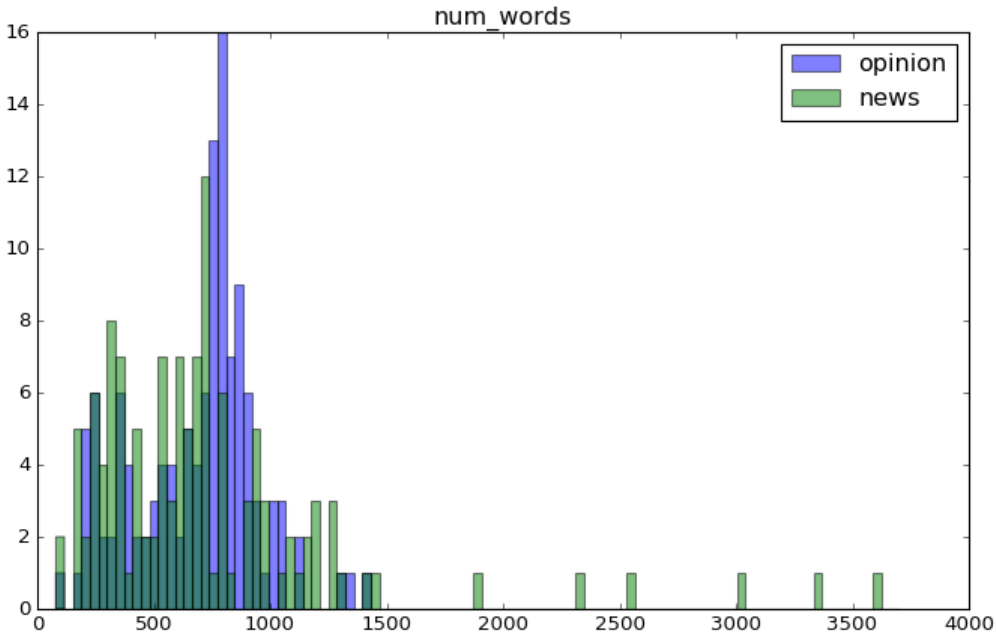
Incorrect:

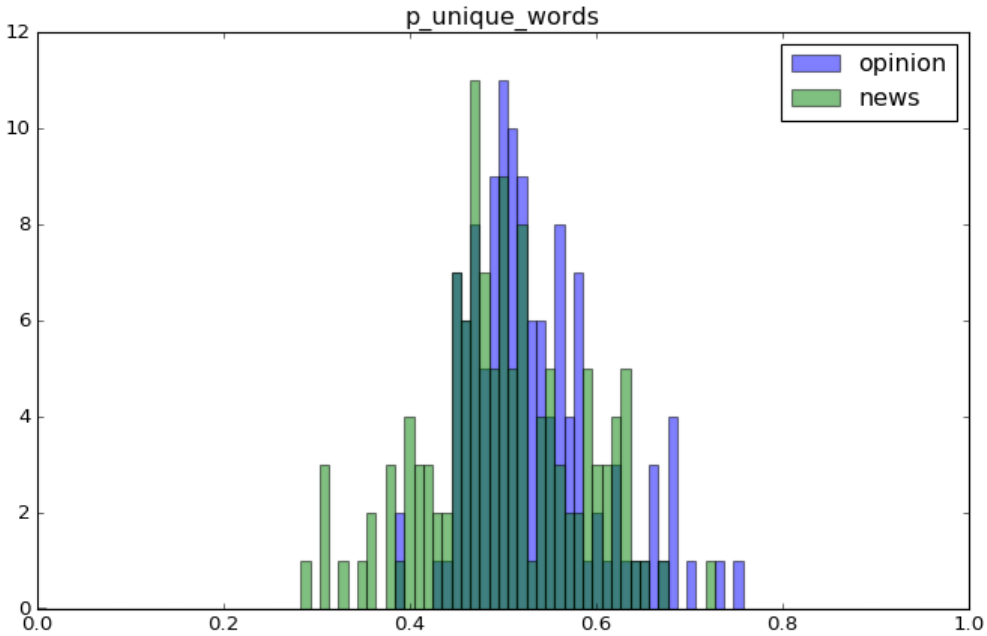
	num_words	p_unique_words	average_word_length	p_stopwords	p_shared	p_seedwords
count	26.000000	26.000000	26.000000	26.000000	26.000000	26.000000
mean	880.192308	0.505502	4.896653	0.424724	0.475896	0.003384
std	580.663208	0.080085	0.243265	0.031645	0.031147	0.002625
min	101.000000	0.311719	4.415842	0.349745	0.405941	0.000000
25%	707.750000	0.466478	4.742144	0.412317	0.462480	0.001515
50%	810.000000	0.502294	4.874581	0.421253	0.476790	0.002648
75%	952.000000	0.540331	5.080284	0.442064	0.500967	0.004636
max	3362.000000	0.680365	5.300509	0.492267	0.522920	0.009901

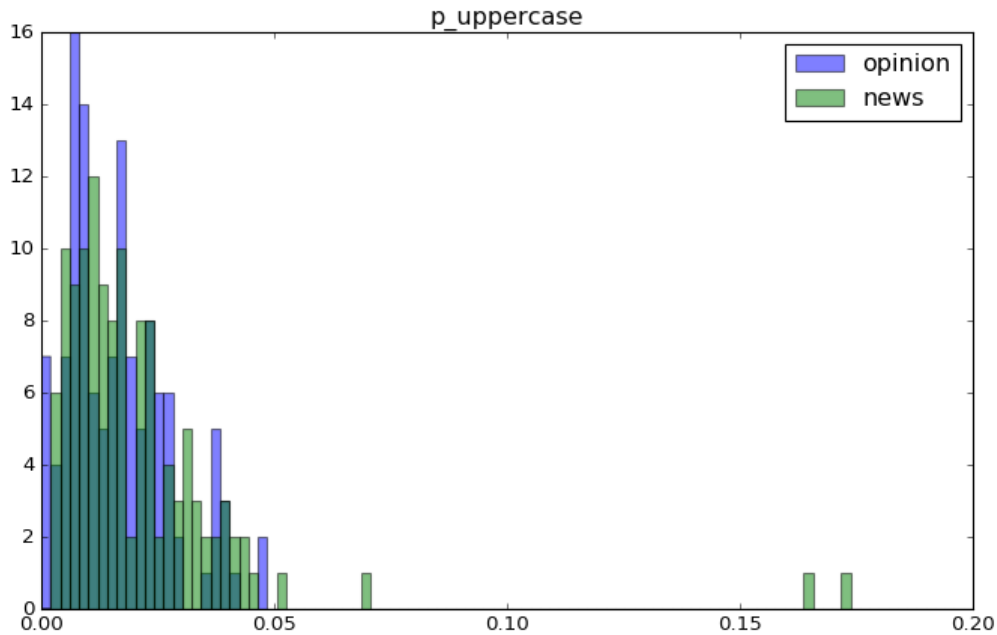
Bibliography

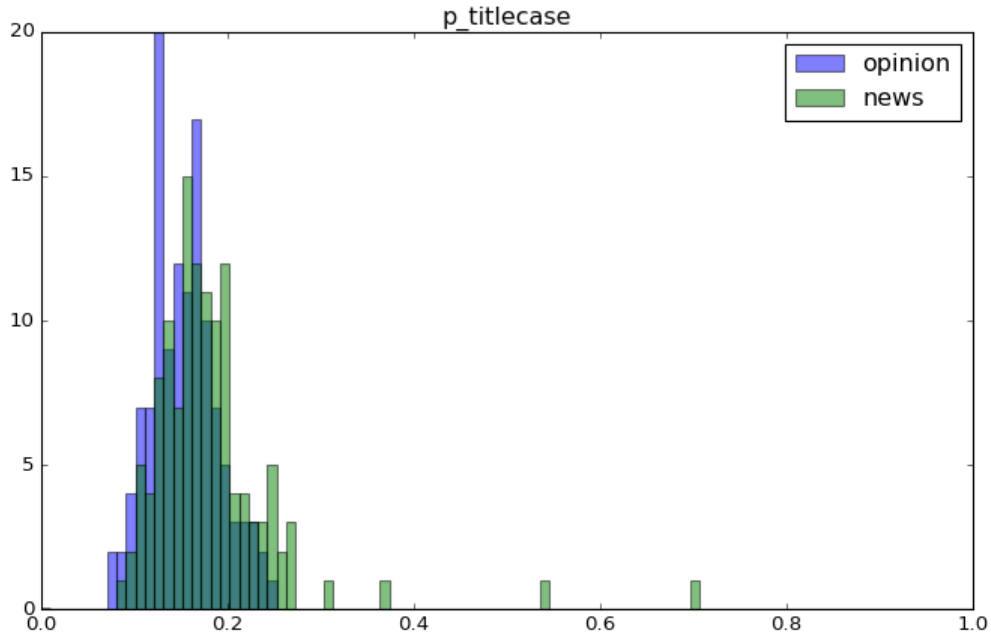
Kamal, Ahmad. "Subjectivity Classification Using Machine Learning Techniques for Mining Feature-Opinion Pairs from Web Opinion Sources." Accessed March 16, 2016. http://arxiv.org/ftp/arxiv/papers/1312/1312.6962.pdf.

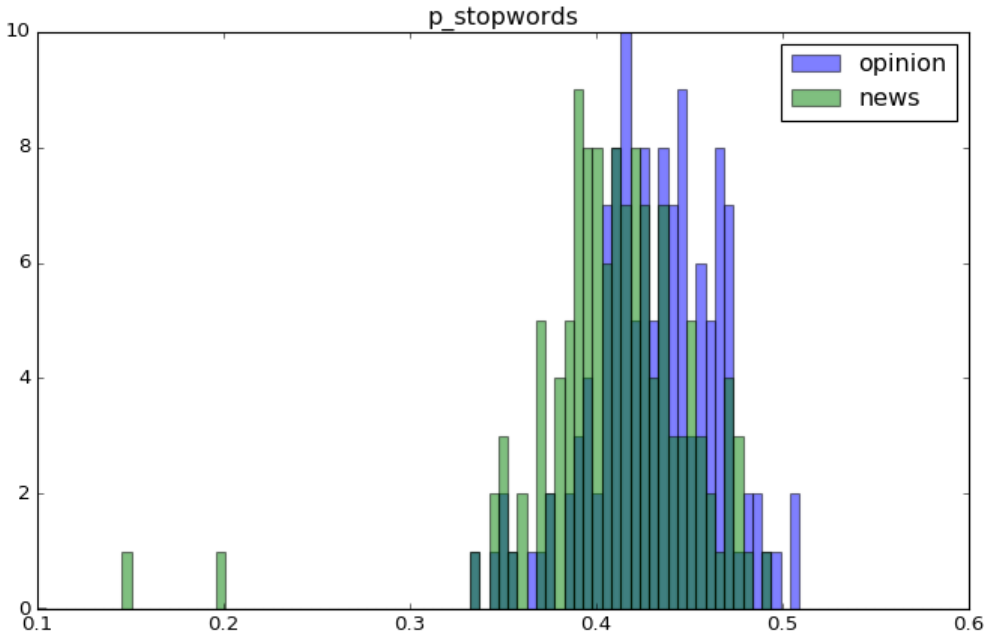
Appendix: Frequency Distribution of Statistics

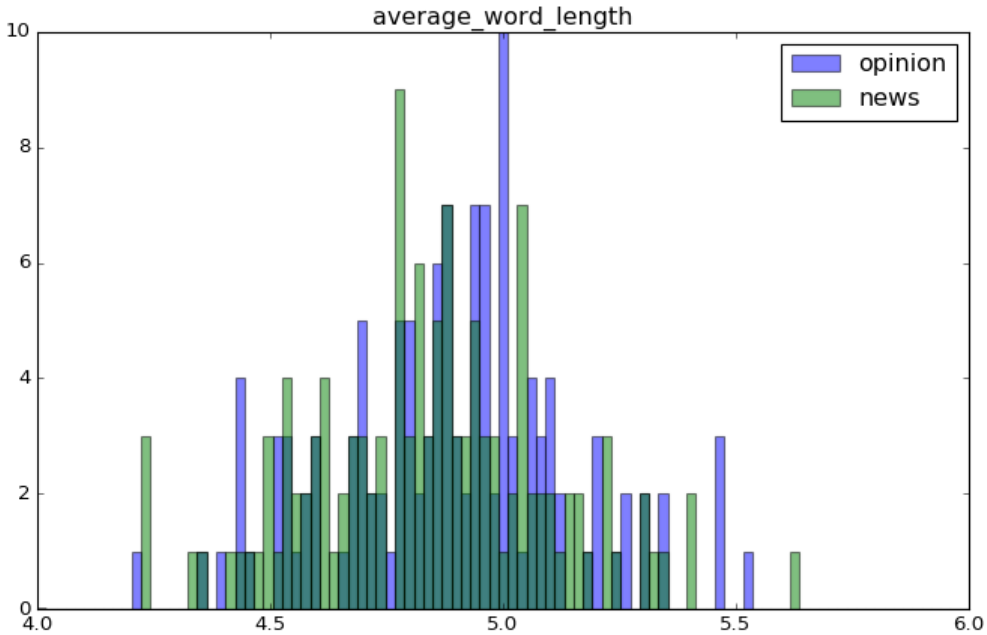





p_uppercase
opinion
news
0
2
4
6
8
10
12
14
16
0.00
0.05
0.10
0.15
0.20


p_titlecase
opinion
news
0
5
10
15
20
0.0
0.2
0.4
0.6
0.8
1.0


p_stopwords
opinion
news
10
8
6
4
2
0
0.1
0.2
0.3
0.4
0.5
0.6


average_word_length
opinion
news
10
8
6
4
2
0
4.0
4.5
5.0
5.5
6.0