Examine The Raw Data

Data-Set tweets.tsv

When looking at the data-set we can notice that

The following describe the distribution among the “account” in the raw data

Account == @realDonaldTrump , 3901 (number of records that satisfy the constraint)
Account == @PressSec , 13 (number of records that satisfy the constraint)
Account == @POTUS , 1 (number of records that satisfy the constraint)

Trump switched to a secured iPhone in April 2017, so we look now on the distribution over the data until April 2017, and from April 2017

Date of tweeting $\geq$ April 2017 , 2 (number of records that satisfy the constraint)
Date of tweeting < April 2017 , 3913 (number of records that satisfy the constraint)

The distribution of the devices on the raw data are

Web device , 293
Instagram device , 3
Iphone device , 1104
Android device , 2508
Other device , 10

The following assumption is necessary for the labeling part:
start with the well-accepted assumption that @realDonaldTrump tweets coming from an Android phone were written by Donald Trump himself, and that during the election, those posted on an iPhone came from someone else (i.e. his staff). In that asummption in mind i cleaned all records that are satisfying the following constrain:

Training Data =

$$\sum_{1}^{3915} Recored_i \mid Recored_i[Account] == @realDonaldTrump \text{ and } Recored_i[date] < April\ 2017 \text{ and } Recored_i[device] \text{ is in } [iphone, android]$$

Now we need to label the data, so i change the device column to “target” and
Iphone = NOT Trump
Android = Trump

Extract features for modelling

When we look at the row data we have the following information, first the date and time when the tweet posted, second the text of the tweet.

Feature based time:

The time data look < year - month - day Hour:Minutes:Seconds >

One feature is based on the day in the week the tweet was posted, Monday=0, Sunday=6
The underlying assumption here is that in the weekends (not a working time), Trump will post more tweets, because his staff are not working.

The other features are based on the time in the day:

Work-time, and not working time, the underlying assumption is again, Donald's staff will post more tweets in work-time and less tweets on nights, the feature is a one hot feature (binary)

Work-time is defined as, 6AM-6PM
Not Work-time is defined as, 6PM-5AM

Feature based text:

A choose some basic features based on the text of the tweet:

- Hash tags
- References to accounts
- Urls
- Time
- Exclamation mark
- quotes

All the above features are based on the occurences of the feature in the given tweet text.

TF-IDF:
A more detailed set of features can be built by considering the frequency of words and phrases in the corpus. I constructed these using the Tf-Idf (Term-frequency Inverse-document frequency) method as implemented in python sklearn. I was looking on unigrams and bigrams frequency. To not over fitting the model i bounded the tf [0.03-0.5]

For the pre-processing of the tokinzing of the data, i decided to remove stop words, stem the words and to use WordNet Lemmatizer.

Regex cleaning:
I lower case all the text, i transform each url link, time, number to a generic form. Clean all the punctuation.

The flow process of the cleaning part:
Rwa Text **-->** Regex cleaning **-->** remove StopWords **-->** stem the words **-->** make POS TAG for the WordNet **-->** Use WordNet Lemmatizer.

Noticed that the TF-IDF was performed on the train-set each time in the 10 fold cross validation, to prevent overfitting on the test.

One thing it worth a mention, in a more deeper analysis and constructing feature, we can make use of the stop word, because different people use different stopword on their writing, tortical we can catch the pattern in the stop word of Donald Trump.

Sentiment analysis:
I use the Textblob for sentiment analysis.TextBlob is an open source text processing library written in Python. It can be used to perform various natural language processing tasks such as part-of-speech tagging, noun-phrase extraction, sentiment analysis, text translation, and many more. TextBlob stands on strong shoulders of NTLK, which is the leading platform for building Python programs to work with human language data. I proforme the sentiment analysis on the clean tweet. I used the subjectivity and polarity.

POS
I had a little experience with the nltk pos modules, it for some inspection of the result, i notice that the nltk module performed badly on the corpus, one explanation is that the the nltk train on a different corpus e.g. The New York Times, and not on a corpus that was extracted from the web.

Normalize the data-
I normalize the data use the simplest formal [0,1] :

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

Feature Selection:
For features selection i used the chi2 test, and i also try using the info gain ratio.

Classifiers

- Logistic Regression (L=2)
- SVM with linear kernel
- SVM with polynomial kernel
- SVM with sigmoid kernel
- Xgboost + Lasso (L=1) **(not from sklearn import xgboost)**

All classifiers used in the default arguments because of lack of time to parameter tuning them.

The setup for the experiments

Because the lack of time i had i used only 6 setups :

	Setup #1	Setup #2	Setup #3	Setup #4	Setup #5	Setup #6
Stemming	true	false	false	false	false	true
Lemmatization	true	true	false	false	false	true
Tf-Idf	true	true	true	false	false	true
Stop words	true	true	true	true	true	false
Feature Selection	true	true	true	true	false	true

On each of the setups i used k=10 cross validation, all the features extracted from the data before the cross validation except the tf-idf to prevent over-fitting.

Results:

The result are shown on Table 1 to 6 for each configuration of the parameter shown in the setup section. The value in each metric is the mean value over the 10 k folds. Table 1 represent the configuration of all parameters are equal to true, that means that the preprocessing stage included, removing stop words, using the Porter Stemming Algorithm and WordNet shows, in the stage of extracting features included tf-idf features, the tf range was between [0.03-0.6] percentage of the corpus. At the end i did also feature selection based chi test. Table 1 shows that the highest accuracy score was achieved by the Xgboost algorithm, while the lowest accuracy score achieved by the SVM with a sigmoid kernel. We can see a same behavior with the F score, the higher is the Xgboost while the lowest score made by SVM with a sigmoid kernel. Figure 1 is shows the accurese score in depending on the different classifiers. Anther behavior that we can see from the data that the linear kernel outperform the sigmoid and the polynomial one, that can apply that our data can be separable linearly, i.e. If your data is linearly separable (linear frontier between the two classes hyper-plane's regions) it is to be expected that the linear kernel would have the best performance, this behavior shows in the data of Table 1. Lets look at the Logistic Regression (LR) and linear SVM, LR finds a classifier which maximizes the conditional likelihood of the training data. SVM maximizes the margin between points closest to the classification boundary. We can see that they both perform almost the same, LR is a bit higher. Linear SVM (without using any kernel function) perform usually as same as Logistic Regression as both are structurally similar. SVM usually does a better job as it is a large margin classifier. However, SVM may perform worse than Logistic Regression when the dataset is small, thus data points near the decision boundary(Support Vectors) may not be true representation of the actual decision boundary, thus may form false maximum margin classifier boundary. When looking at our data size, about 3700 records, it’s relatively small amount of data to compering to big machine learning tasks. So it can explain why the LR did a little bit better than linear SVM. and LR can handle better more variety of features, than the SVM. The best result came from the Gradient boosting, also known as Xgboost, the motivation for using it, came from the Kaggle challenges, Gradient boosting does very in the Kaggle competition well because it is a robust out of the box classifier (regressor) that can perform on a dataset on which minimal effort has been spent on cleaning and can learn complex non-linear decision boundaries via boosting. Most of the arguments can be applied to regressors as well.

The same behavior that explained above is preserved over all setup (1 to 6), that apply that the most significant feature where does who i extracted based on counting and time. The n-grams feature (n={1,2}) did not contribute to the learning process.

One main drawback is the time that took the Xgboost to train is significantly larger than all the other.

Limitation

Data representation:
I didn't have enough time to experience with one-hot representation, one-Hot-Encoding has a the advantage that the result is binary rather than ordinal and that everything sits in an orthogonal vector space. The disadvantage is that for high cardinality, the feature space can really blow up quickly and you start fighting with the curse of dimensionality. In these cases, I typically employ one-hot-encoding followed by PCA for dimensionality reduction. I find that the judicious combination of one-hot plus PCA can seldom be beat by other encoding schemes. PCA finds the linear overlap, so will naturally tend to group similar features into the same feature

parameter tuning:
because the luck of time i had, i also did a minimal exploration with the variety of the parameters that can be tuning, in the features part e.g. tf-idf, for example maybe only using tf on tweets will achieve a better result then using the tf-idf, and in the classification algorithms e.g. LS.

The Mean Value over the 10 k fold of metric evaluation

id	AUC	Accuracy	F1_score	Precision	RECALL	Time
lasso_xgb	0.838209486	0.89658108	0.9300662155	0.8790839942	0.9876366534	12.8060522
LogisticRegression	0.8239340473	0.8813394629	0.9192663391	0.8731143965	0.9708956175	0.0086714
xgb	0.8590451964	0.9035139953	0.9334903423	0.897520231	0.9728860558	12.8060522
lasso	0.8047924333	0.8735631812	0.9152561364	0.8581272011	0.9808525896	0.0055886
svm_sigmo	0.5435540168	0.6215460167	0.7320034871	0.7212657672	0.7432047809	0.3192492
svm_poly	0.6308968011	0.7721149319	0.858312212	0.7561985336	0.9924191235	0.2641991
svm_liner	0.8128108	0.8793857	0.9190213	0.8628465	0.9832446	0.1463721

	012	506	842	753	215	

Table 1: present the metric score for the first setup

id	AUC	Accuracy	F1_score	Precision	RECALL	Time
lasso_xgb	0.8363813703	0.8954768941	0.9293461601	0.8777246025	0.9876334661	12.462918605
LogisticRegression	0.8202331119	0.8780022308	0.9169547833	0.871089334	0.968089243	0.0085784
xgb	0.8557889649	0.9018496199	0.9324631905	0.8946859205	0.9736796813	12.4525434
lasso	0.8045237838	0.8735716454	0.9152722701	0.8578152507	0.9812494024	0.0058287
svm_sigmo	0.5234821362	0.6049601527	0.7204399441	0.7094987194	0.7320685259	0.3152788
svm_poly	0.6326812882	0.7735076895	0.8591830803	0.7571160405	0.9932191235	0.263636
svm_liner	0.8127974939	0.879391902	0.9189839589	0.8627695487	0.9832430279	0.1422985

Table 2: present the metric score for the setup #2

ID	AUC	Accuracy	F1_score	Precision	RECALL	Time
lasso_xgb	0.8362774324	0.8960278329	0.929823387	0.8773473506	0.9892270916	13.7594181
LogisticRegression	0.8244340011	0.8813386849	0.9192189793	0.8736140336	0.9700940239	0.0088021
xgb	0.8578833397	0.9029630608	0.9331703582	0.8963912817	0.9732812749	13.5131508
lasso	0.8034186769	0.8727398595	0.9147502029	0.8571734627	0.9808573705	0.0056854
svm_sigmo	0.5374723345	0.6194086117	0.7316425625	0.7170039619	0.747238247	0.3200699
svm_poly	0.628418124	0.7704513431	0.8573603844	0.7549274424	0.9920239044	0.270327
svm_liner	0.8128147852	0.8793980792	0.9190542698	0.8628905646	0.9832525896	0.1479657

Table 3: present the metric score for the setup #3

id	AUC	Accuracy	F1_score	Precision	RECALL	Time

lasso_xgb	0.8363829639	0.8954730339	0.9293128445	0.8775588142	0.9876366534	12.8831528
LogisticRegression	0.8245653516	0.8818857549	0.9196024003	0.8732051229	0.9712908367	0.0088425
xgb	0.8607620168	0.9051829918	0.9345871382	0.8979180295	0.974476494	12.8401201
lasso	0.8045261742	0.8735647116	0.9152498643	0.8576600738	0.9812541833	0.0061855
svm_sigmo	0.5213026174	0.60410527	0.7199506104	0.7074349973	0.7332557769	0.323603
svm_poly	0.6251426355	0.768777704	0.8565861623	0.7532821711	0.9928207171	0.2697493
svm_liner	0.812815582	0.8793957665	0.9189834891	0.862698166	0.9832541833	0.134468

Table 4: present the metric score for the setup #4

id	AUC	Accuracy	F1_score	Precision	RECALL	Time
lasso_xgb	0.8391193737	0.8971320188	0.9303811515	0.8795893424	0.987638247	5.58035975
LogisticRegression	0.8226554639	0.8799281995	0.9182480652	0.8725821119	0.969289243	0.0083404
xgb	0.8596953856	0.90406879	0.9338796065	0.8978539531	0.9732940239	11.1548555
lasso	0.8046533532	0.8730114816	0.9147989533	0.8581899593	0.9796653386	0.005864
svm_sigmo	0.5261670954	0.6054965059	0.7199032706	0.7109203569	0.7292733068	0.3237627
svm_poly	0.6256096913	0.7690539472	0.8567387914	0.7535480414	0.9928207171	0.2706282
svm_liner	0.8128221426	0.8793957794	0.9190335493	0.8628552011	0.9832589641	0.1774349

Table 5: present the metric score for the setup #5

id	AUC	Accuracy	F1_score	Precision	RECALL	Time
lasso_xgb_	0.839006631	0.8962994335	0.9297107338	0.8798574637	0.9856446215	4.68100905
LogisticRegression	0.8239912643	0.8846581507	0.9219369884	0.871087383	0.9792669323	0.0083663
xgb	0.85710255	0.90156413	0.93210108	0.89645855	0.97089402	9.356668

	82	89	86	73	39	
lasso	0.8044897182	0.8749574564	0.9163916483	0.8569781336	0.984850996	0.0053501
svm_sigmo	0.5256574888	0.6051809127	0.7196318777	0.7104270086	0.729238247	0.2747898
svm_poly	0.6714265567	0.7965070825	0.8714576793	0.7773413141	0.9916270916	0.2123601
svm_liner	0.8127897644	0.8793911411	0.9189847157	0.8627369914	0.9832525896	0.2284628

Table 6: present the metric score for the setup #6

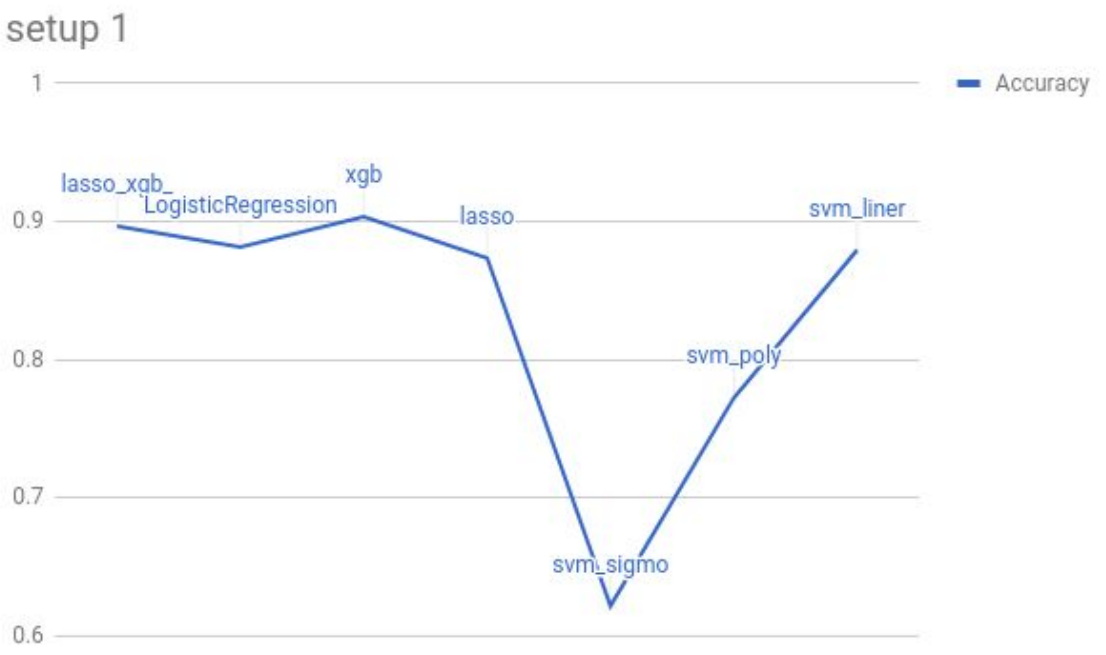


Figure 1: the accuracy score on setup #1 for all the different classifiers.

Part 2

Tpic model
Topic modelling can be described as a method for finding a group of words (i.e topic) from a collection of documents that best represents the information in the collection. It can also be thought of as a form of text mining – a way to obtain recurring patterns of words in textual material.

Pre-Processing Data

The process is as the following:

Trump tweets:

- Lower case all the text tweet
- Change the link to single form
- Change the @username to a single form
- Remove additional white space
- Remove &amp (come for the html)
- Remove hashtags e.g. #word --> word
- Change Number e.g. time to singal form
- Remove punctuation marks
- Remove stop words with extended list
- Do a lemmatization using WordNet

FCC dataset:

- Remove non-letters
- Convert to lower case
- Remove stop words with extend list
- Lemmatize the word using wordnet
- Stem the words using Porter Stemming

Evaluation metric

Perplexity measure

The problem is come from the language community, perplexity is a measurement of how well a probability distribution or probability model predicts a sample. It may be used to compare probability models, but it don't capture the semantic information that we want. For example if we have a corpus that contains only the word “apple”, and our model is always predicts “apple”, and we have a held out test corpus that only contain the word “apple” then the log likelihood will be good, but that is not a good topic model, this measure dont tell us anything of how well the semtic of the corpus is been captured.

Silhouette measure
When one need to evaluate the unsporvise method
the best way to evaluate the results of your clustering efforts is to start by actually examining, human inspection, the clusters formed and making a determination based on an understanding of what the data represents, what a cluster represents, and what the clustering is intended to achieve. In our case the documents that are in the same cluster are talking about similar concepts i.e. topics. Because such inspection is not feasible, i used the Silhouette measurement.

eye-bowling
The process of analysis the topic through a human eye, the problem with that is, the time it takes and the need for an expert in the field.

topic model Results

The main parameters that i try to tune in this task are:

- Number of samples, how much data to train the model
- Number of features that come out from the CountVectorizer or the TfidfVectorizer
- Number of Topics

Trump tweet topic model

I ran all the number of the topic in the range of [2,5,10,20]
And the cluster i ran in the range of [1..10]
Feature i ran on the range of [500,750,1000]

I will analysis a little part of the data because i have not enough time to scan all the outputs with my eyes.

When dealing with small data-set under 5000 rec, the number of samples was assign to the maximal on each of trails.

I train the LDA topic model on the whole data and on ⅔ of the data:
(the ⅔ parameter choose in mistake i set the k to 3 instead of 10, but i think the result are more human meaning full :

#setup_one

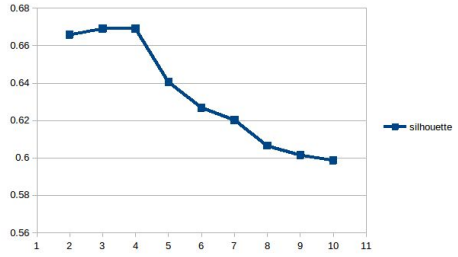
Topic = 2 with , 500 top features from the vectorize process.

Topic # K=2 (the number of topics)	words
Topic #0	hillary clinton job like crook medium cruz look good fail want bad people time run ted need report jeb great
Topic #1	trump great thank poll makeamericagreatagain america big interview new people enjoy donald vote debate win tonight news president watch night

The topic are order form the most important to the least, i decided to show only the TOP 20

The first thing that pop to the eye when we eye bowl the the results is separation between hillary clinton on topic #0 and trump in topic #1. We can also noticed that the hash_tag, #makeamericagreatagain is related to trump and also the world donald is in the second topic, we can also thing word like “fail” “bad” are also reasonable to the context of hillary because all the tweets wrote by trump and his staff so it is a reasonable assumption to connect the words to hillary. We can see that the two topic separated nicely the word and one refers to hillary and the other refers to trump. The mean Perplexity measure over the 3 k fold was Mean train: 342.353044008, Mean test: 865.416166787. I don't refer this metric any father analysis, because the bug in sklearn module.

When we look at the cluster analysis by the K-means



The x-axis represent the number of the cluster, the y-axis represent the value of the silhouette score. We can see that after C=4 the is a fall in the silhouette value, so it seems reasonable to guess that the topic C=2..4 is a good clustering separation because the topic model was set to 2 topic, and maybe there are two hidden topic in each of the two main topics.

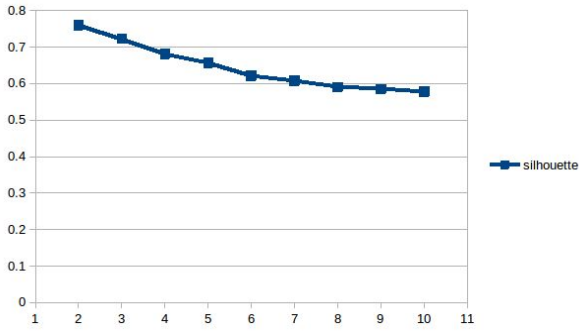
#setup_two

Lets look now at the full data training, that apply that we will not have a train set.

K=2 number of topics and number of feather is None, apply we use all the features in the learning process of the LDA

topic	words
Topic #0:	trump thank great makeamericagreatagain america enjoy interview new tonight morning vote today big poll iowa night watch thanks support join
Topic #1:	hillary president clinton trump poll bad people donald want cruz like win crook job medium debate country know rt time

Here we cannot see any significant concept, within the two topics.



the result of the K-means cluster, we see that the correlation between the topic that set 2 two and the cluster C that set to 2 is high.

#setup_three

In this setup of experiment i set the K=10 and used the 500 top features for the vectorizer process.

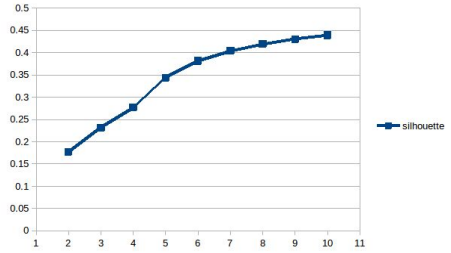
topic	words
Topic #0	interview enjoy tonight morning watch pm nation fox votetrump hope statement face wisconsin meet atpme atpm chris atam press announce
Topic #1	news medium join talk fake fail tomorrow lie election read enemy know apologize night sign story book pay live saturday
Topic #2	candidate dont let marco race law rubio cruz republican crime vote presidential kasich like special look control remember party open
Topic #3	thank great trump soon look jeb night good forward today job bush honor nice americafirst day state nevada wonderful iowa

Topic #4	crowd supporter best massive poll lead bernie far rally big arizona iowa texas love true choice national sad colorado mention
Topic #5	trump america great time ted cruz vote want donald country need say right mr thanks story win change reporter people
Topic #6	job bring help focus wall democrat sell hear obamacare crazy way hit common core penny islamic finally plan create didnt
Topic #7	poll trump new big people win debate rt gop dishonest lead year work come day number cruz carson medium wow
Topic #8	trump president like people thing good american campaign obama doesnt run tough man smart think woman donald tell usa world
Topic #9	hillary clinton makeamericagreatagain crook bad want trump support bernie beat report ad sander email rating endorse fix job medium false

Topic #0, can may suggest that the topic is about the time and interview shows.
Topic #6, may apply the plan of trump, because the word “wall” and “islamic” and “crate”

The other i couldn't interpreter, there nothing meaningful popping from the words, we need to remember that eye bowling need to be performed by an expert of the given field and sadly i don't know much about american politics. The perplexity measures on the mean train-set: 424.682083199 , and mean Test-set: 1595.59871204

I will show the silhouette measure when applying K-means clustering



This suggest that although i didn't see any meaningful topic in the ten topic above the clustering is relative fit the 10 topic when C=10 because the monotonically increasing function. For conclusion i think the best result came from when the K=2, i can't rely on the perplexity measure because the bug.

FCC dataset topic model results:

Here again we can assume from the knowledge on the data that we expect to see maybe two clusters that one belong to the bots and the other belong to the human that wrote the

comment. Intuitively starting with two topics and incersing up will make sense, as in the trump tweet i strat with 2 topic until 20 topic and each time a try to cluster the topic to 2 cluster up to 10 clusters. Again i'm not nearly an expert on the problem of the Net neutrality issue, and unfortunately i didn't have time to read the extra literature about this problem in the U.S.

RESULTS:

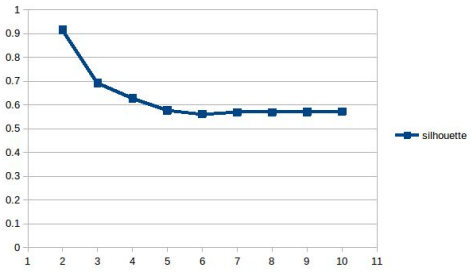
In the following experiment i use tf vectorize, as in contrast to the Trump tweets. I work on rafely on 15 percent of the data 35000 samples. And used the top 850 features form the vectorizer, to reduce the computational time

==========**SETUP #1** ==========

K=2

topic	words
Topic #0	rule fcc access pai internet oppose speed power site ajit cable carrier company want pay house white money ability monopoly charge stand cap american
Topic #1	neutrality net title internet isps ii support fcc rule strong lane open slow company fast pai service oversight business let chairman website isp

I can't notice any union concept in the two of the above topics, it could imply that a bigger K need hear or a butter parameter tuning, in other words i need to increase my number of features, or use tf-idf instead of the idf. The result from the K-mean can implies that they are two main concept here because the samples with C=2 got the highest silhouette score 0.915



X-axis is the n_cluster, and Y-axis is the silhouette score.

==========SETUP #2==========

K=5

The following table shows the top 20 word in each of the 5 topics

topic	words
Topic #0	lane isps slow fast rule website let open block pay throttle clear business small time title fee ii company cable million
Topic #1	internet neutrality net company access service free provider isps allow people consumer business open isp freedom content american corporation regulation

Topic #2	title neutrality net support ii strong oversight isps strongly preserve internet isp regulation rule specifically provider protection protect open maintain
Topic #3	rule fcc pai internet access site oppose cable power ajit carrier want pay company speed house white verizon stand slow sit money employer monopoly
Topic #4	fcc pai chairman plan lawyer american intend famous giant high rule net neutrality censor congress support thank verizon open power censorship isps publicly

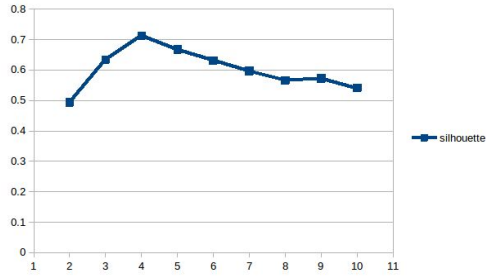


Figure : x-axis is the number of cluster, and y-axis is the silhouette score.

Here i count also found any meaningful interpretation to topics that were generated by the LDA. I try to find topic up until K= 30 but the interpatrion just got harder and the silhouette value when applying k-means got worse (the value decreasing as the K increased) .

Different configuration of the parameters

When i used for training only 3 percent of data and i did stem and lemmatization to the word and instead of only tf, i used TF-IDF, with 3000 top features from the vectorizer,i got a better results. The silhouette score was much more higher on all the cluster i preformed, and some of the topic was seems to talk on a big concept. We can assume that when we want to perform a reduction in the data, it better to remove samples than deal with less features in the LDA model. When i think about it make a lot of sense because if we train on 30% of the data with X features and training on 50% of the data with also X feature it will probably achieve the same result because the same features, when assuming Zipf's law distribution over the words in the corpus.

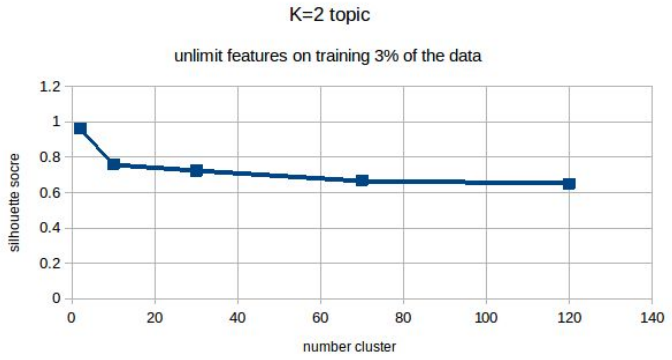
When i defined a k=8 topics using the configuration above, i got the following results:

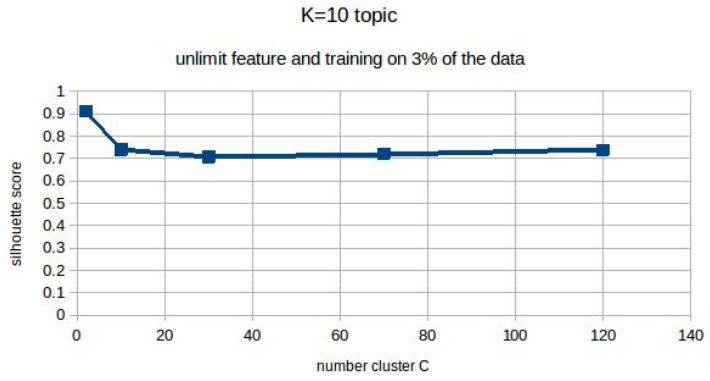
Topic #4	fuck shit crap suck bryce accessibility alex dog luke emperor mug master dick coffee stop rival eye shutdown clayton salary

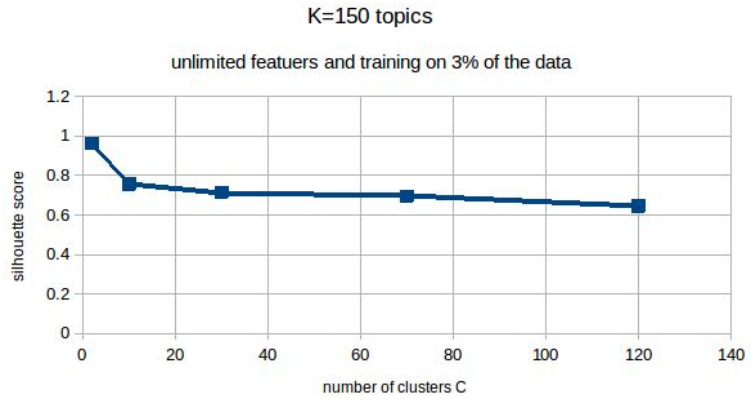
We can see that topic 4 can is belong probably to heater comments or trolls comment. The other topic i couldn't pinpoint some generally concept to them.
As the topics got bigger 20+ it was very hard to find any belong concept. But as the features number got higher the clusters silhouette score get higher also, that means that maybe that the topic were more disjoint from each other so the cluster could separate much more better.

Data shows the following behavior when the when i train on 3% and extracted 3000 features as the topic increased the Cluster silhouette score still behave the same on C=2 it achieve

90% and converge on 70% as the biggest C=100, that implies that the 3000 feature is an anchor. So the last experiment i assign the number of feature None, in other words i didn't bounded the limit on the features.







We can see that the same behavior is occur when applying the number of features in the vectorizer to None (no limit on the features), we can see that in all cases the minimal C achieve better scoring and then C coverage on the area of 0.7.

When looking at the two data sets Trump's tweets and FCC i found the Trump more interpretable than FCC, maybe because i'm more familiar with the filed, and maybe on a big scale of training data over 50% the topic will be more interpretable. I also noticed the weird behavior of the cluster on the FCC, as the topic increased the cluster seems to still prefer the smaller one over the bigger (value of C) when looking on the silhouette scoring. I think that implies that the there maybe בomments they support and opposeת or, computer bots and humans but it seems to be in the area of binary clusters. While in Trump's dataset it is not the cases.

Limitation:
Because the limitation on my resources e.g. server with enough power, i deal with only a relatively small amount of data in the cpurpus.
Basically i can reduce the computational time by reducing two parameters, one is number of sample per training, and the other is the maximal number of feature per training in a given data set.