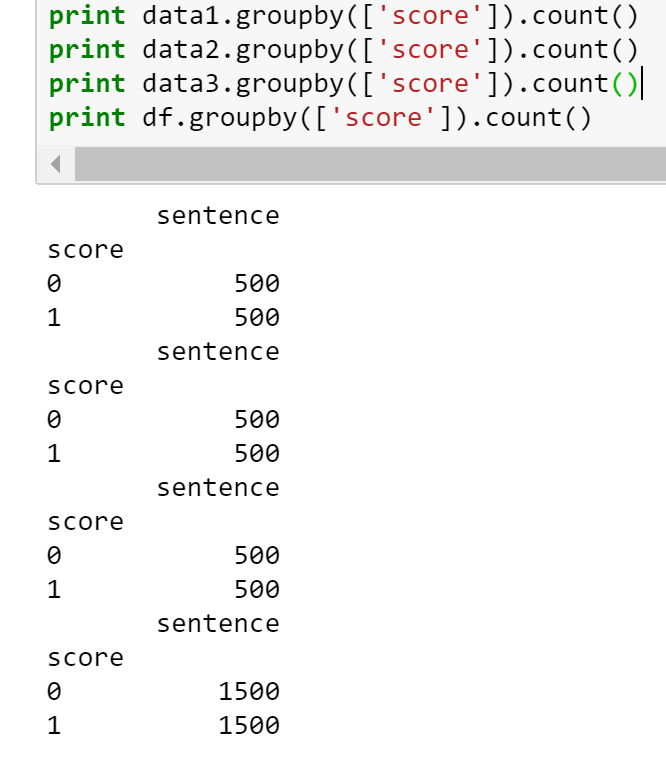
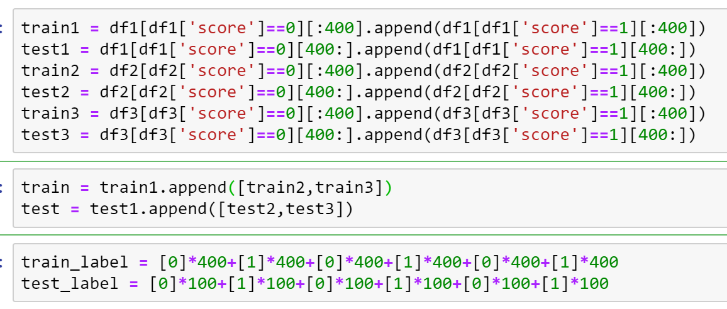
Part1:

1. Download Sentiment Labelled Sentences Data Set. There are three data ﬁles under the root folder. yelp\_labelled.txt, amazon\_cells\_labelled.txt and imdb\_labelled.txt. Parse each ﬁle with the speciﬁcations in readme.txt. Are the labels balanced? If not, what’s the ratio between the two labels? Explain how you process these ﬁles.



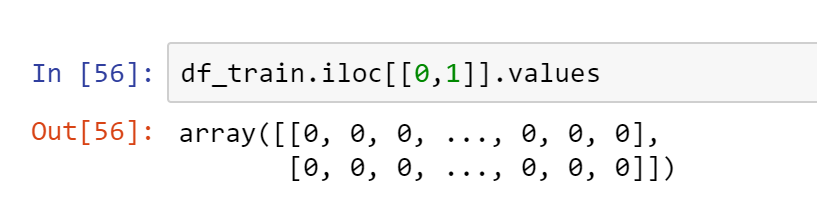
1. Pick your preprocessing strategy. Since these sentences are online reviews, they may contain signiﬁcant amounts of noise and garbage. You may or may not want to do one or all of thefollowing. Explain the reasons for each of your decision (why or why not).   
   • Lowercase all of the words.   
   • Lemmatization of all the words(i.e.,convert every word to its root so that all of “running,” “run,” and “runs” are converted to “run” and and all of “good,” “well,” “better,” and “best” are converted to “good”; this is easily done using nltk.stem).   
   • Strip punctuation.   
   • Strip the stop words, e.g., “the”, “and”, “or”.   
   • Something else? Tell us about it.

Answer:

1. Lowercase all the words: I will do that because uppercase and lowercase doesn't effect the meaning. (Except some abbreviations, which is rare.)
2. Lemmatization of all the words: I will do that because these words have almost same meaning.
3. Strip punctuation: I will do that since punctuation don't have much meaning/information.
4. Strip the stop words: I will do that since these stop words don't have much meaning/information.
5. What else: rule out the number words like "one" "two" "three", "1", "2", "3", which we think somewhat meaningless.
6. Split training and testing set. In this assignment, for each ﬁle, please use the ﬁrst 400 instances for each label as the training set and the remaining 100 instances as testing set. In total, there are 2400 reviews for training and 600 reviews for testing.
7. Bag of Words model. Extract features and then represent each review using bag of words model, i.e., every word in the review becomes its own element in a feature vector. In order to do this, ﬁrst, make one pass through all the reviews in the training set (Explain why we can’t use testing set at this point) and build a dictionary of unique words. Then, make another pass through the review in both the training set and testing set and count up the occurrences of each word in your dictionary. The ith element of a review’s feature vector is the number of occurrences of the ith dictionary word in the review. Implement the bag of words model and report feature vectors of any two reviews in the training set.

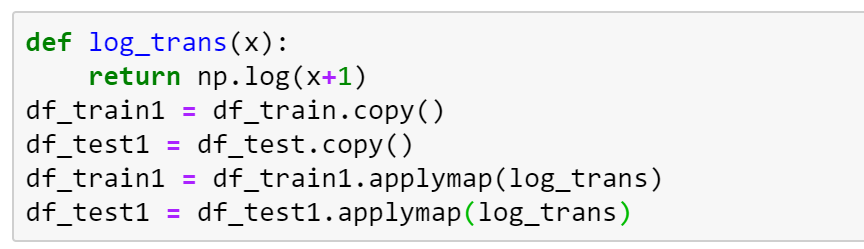
Explain why we can’t use testing set during the step of feature extraction:   
Answer: Will result in biased performance estimates.  
To get an unbiased performance estimate, the test data must not be used in any way to make choices about the model, including feature selection.  
Also since including testing data into feature selection will generate more features. It will probably cause some level of overfitting.

Report feature vectors of any two reviews in the training set (see code output)



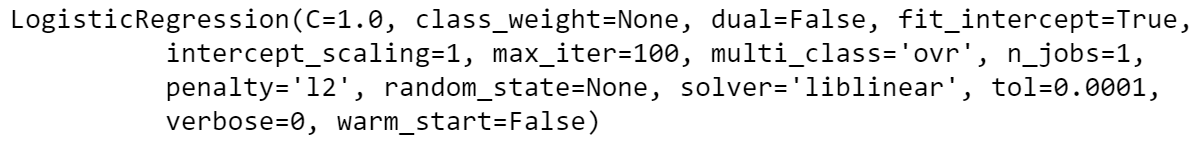
1. Pick your post processing strategy. Since the vast majority of English words will not appear in most of the reviews, most of the feature vector elements will be 0. This suggests that we need a postprocessing or normalization strategy that combats the huge variance of the elements in the feature vector. You may want to use one of the following strategies. Whatever choices you make, explain why you made the decision.  
   • log-normalization. For each element of the feature vector x, transform it into f (x) = log(x+1).  
   • l1 normalization. Normalize the l1 norm of the feature vector, x'=x/|x|.   
   • l2 normalization. Normalize the l2 norm of the feature vector, x'=x/||x||.   
   • Standardize the data by subtracting the mean and dividing by the variance.

Explain: I pick the "log-normalization", because intuitively, it is true that the importance of a word increase when it appear twice in one review. However, appearing twice doesn't means importance of this word was doubled. According to our intuitive sense, the contribution to the significance of the word that it appears for the second time is lower than it appears for the first time. That is why I pick log-normalization since this transformation make more sense according to our intuition.



1. Sentiment prediction. Train a logistic regression model (you can use existing packages here) on the training set and test on the testing set. Report the classiﬁcation accuracy and confusion matrix. Inspecting the weight vector of the logistic regression, what are the words that play the most important roles in deciding the sentiment of the reviews? Repeat this with a Naive Bayes classiﬁer and compare performance.

Logistic Regression:



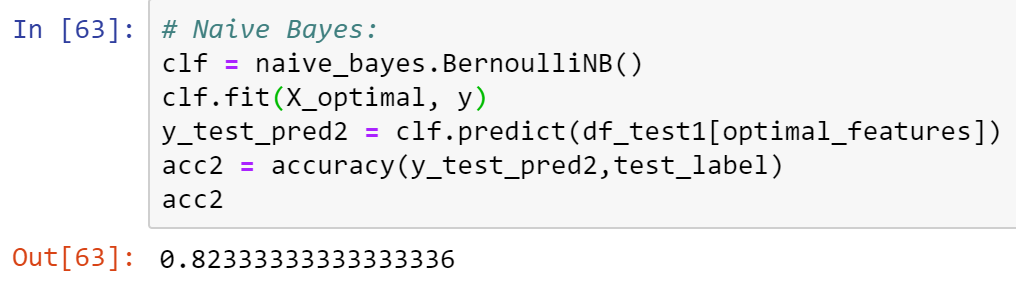
Classiﬁcation accuracy: 0.82333

Confusion Matrix:   
[[260, 40]  
[ 66, 234]]

Words that play the most important roles in deciding the sentiment of the reviews:

[u'bad', u'poor', u'terribl', u'wast', u'slow', u'suck', u'aw', u'disappoint', u'pay', u'stupid', u'break', u'horribl', u'plot', u'start', u'piec', u'hour', u'bland', u'fail', u'avoid', u'rude']  
  
[u'great', u'love', u'excel', u'delici', u'nice', u'amaz', u'fantast', u'best', u'awesom', u'beauti', u'happi', u'comfort', u'good', u'friendli', u'well', u'perfect', u'wonder', u'incred', u'price', u'cool']

Naïve Byes:



Classiﬁcation accuracy: 0.82333

Confusion Matrix:   
[[257, 43]  
[ 63, 237]]

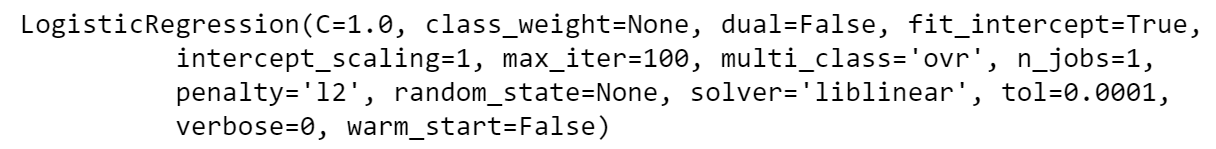
Words that play the most important roles in deciding the sentiment of the reviews:

[u'inexperi', u'disconnect', u'sync', u'unsatisfi', u'sand', u'heist', u'charismafre', u'ticker', u'ticket', u'verg', u'fat', u'reader', u'fals', u'godfath', u'neglig', u'grab', u'humili', u'sashimi', u'unbear', u'dollar']

[u'great', u'good', u'film', u'phone', u'love', u'movi', u'work', u'well', u'like', u'best', u'food', u'make', u'realli', u'place', u'time', u'excel', u'nice', u'go', u'servic', u'price']

1. N-gram model. Similar to the bag of words model, but now you build up a dictionary of ngrams, which are contiguous sequences of words. For example, “Alice fell down the rabbit hole” would then map to the 2-grams sequence: ["Alice fell", "fell down", "down the", "the rabbit","rabbithole"], and all ﬁve of those symbols would be member softhen-gram dictionary. Try n=2, repeat (d)-(g) and report your results.

Logistic Regression:



Accuracy: 0.636666667

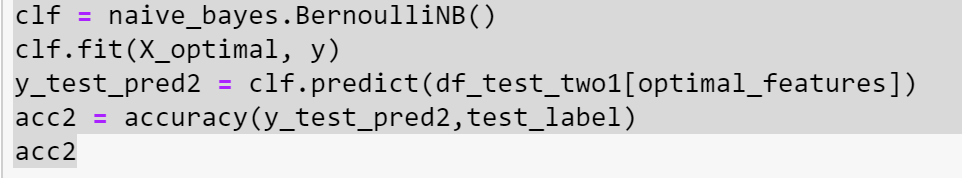
Confusion Matrix:  
[[269, 31]  
[187, 113]]

Words that play the most important roles in deciding the sentiment of the reviews:

[u'wast time', u'disappoint', u'wast money', u'custom servic', u'ever go', u'poor qualiti', u'stay away', u'piec junk', u'horribl', u'rat', u'bad phone', u'bad film', u'bad ever', u'buy product', u'realli bad', u'wait wait', u'make mistak', u'even bad', u'act bad', u'anytim soon']

[u'work great', u'highli recommend', u'great phone', u'great product', u'food good', u'realli good', u'easi use', u'great food', u'great film', u'reason price', u'good price', u'food delici', u'great servic', u'love place', u'film great', u'work fine', u'pretti good', u'well make', u'love', u'good product']

Naïve Bayes:



Accuracy: 0.63666666666666671

Confusion Matrix:  
[[270, 30]  
[188, 112]]

Words that play the most important roles in deciding the sentiment of the reviews:

[u'littl els', u'obviou bluegreenscreen', u'low qualiti', u'servic receiv', u'waitress littl', u'alway old', u'entir movi', u'muffin come', u'see ok', u'redeem would', u'belowpar script', u'uneasi bad', u'never treat', u'superfici movi', u'work balanc', u'ear hurt', u'headset time', u'side restaur', u'feel cheat', u'time serv']

[u'work great', u'highli recommend', u'work well', u'sound qualiti', u'film great', u'great phone', u'great food', u'go back', u'great product', u'come back', u'batteri life', u'great film', u'realli good', u'great servic', u'food good', u'ca nt', u'easi use', u'realli like', u'work fine', u'year ago']

1. PCA for bag of words model. The features in the bag of words model have large redundancy. Implement PCA to reduce the dimension of features calculated in (e) to 10, 50 and 100 respectively. Using these lower-dimensional feature vectors and repeat (f), (g). Report corresponding clustering and classiﬁcation results. (Note: You should implement PCA yourself, but you can use numpy.svdor some other SVD package. Feel free to double-check your PCA implementation against an existing one)

1-gram:

10 features:

Logistic regression:

Accuracy: 0.618  
Confusion Matrix:  
[[251, 49]  
[180, 120]]

Naïve Bayes:

Accuracy: 0.6067  
Confusion Matrix:  
[[190, 110]  
[126, 174]]

50 features:

Accuracy: 0.695  
Confusion Matrix:  
[[256, 44]  
[139, 161]]

Naïve Bayes:

Accuracy: 0.626666666667  
Confusion Matrix:  
[[213, 87]  
[137, 163]]

100 features:

Logistic regression:

Accuracy: 0.715  
Confusion Matrix:  
[[246, 54]  
[117, 183]]

Naïve Bayes:

Accuracy: 0.65  
Confusion Matrix:  
[[210, 90]  
[120, 180]]

2-gram:

10 features:

Logistic regression:

Accuracy: 0.515  
Confusion Matrix:  
[[297, 3]  
[288, 12]]

Accuracy: 0.573333333333  
Confusion Matrix:  
[[234, 66]  
[190, 110]]

50 features:

Logistic regression:

Accuracy: 0.525  
Confusion Matrix:  
[[294, 6]  
[279, 21]]

Naïve Bayes:

Accuracy: 0.57  
Confusion Matrix:  
[[233, 67]  
[191, 109]]

100 features:

Logistic regression:

Accuracy: 0.538333333333  
Confusion Matrix:  
[[287, 13]  
[264, 36]]

Naïve Bayes:

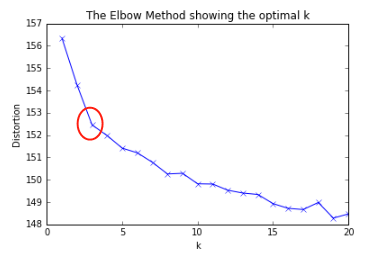
Accuracy: 0.561666666667  
Confusion Matrix:  
[[234, 66]  
[197, 103]]

1. According to the above results, bag of words are better than the 2-gram and PCA for bags of words. Reason might because 2-gram features are less powerful on interpreting the sentimental compare to the bag of words. Besides that, the number of the key features (bag of words) might be much larger than 100 and that is why PCA for bags of words behave not as good as the origin bag of words.

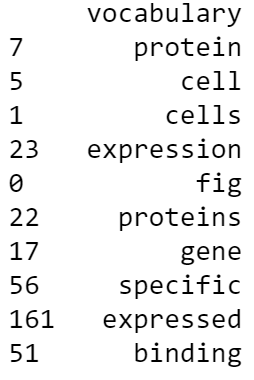
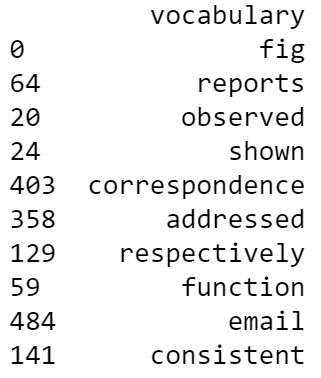
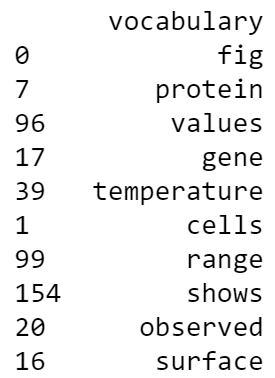
Part 2:

1. 1. Select K:

We select K=3 since according to the inertia of the clustering process, we found that there is a significant elbow at point K = 3 when we plot all the inertia from k = 1 to k = 20.

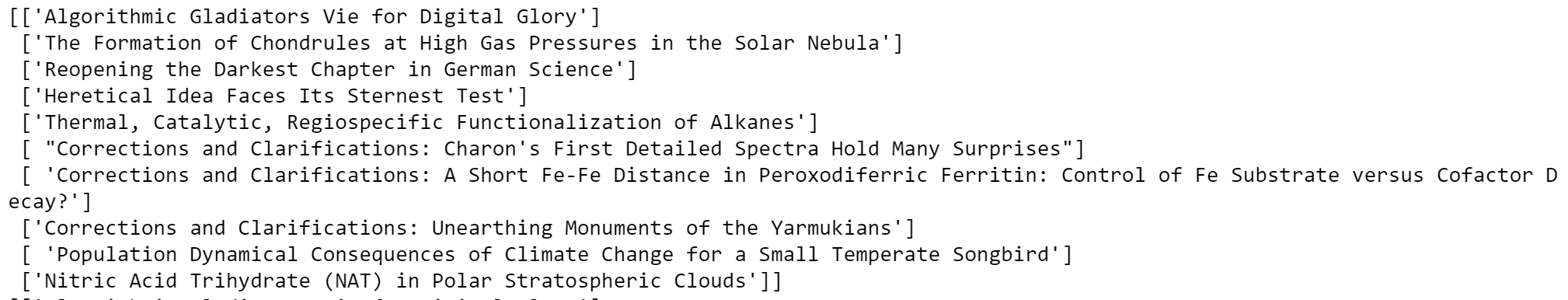


2. Top ten words for each cluster:

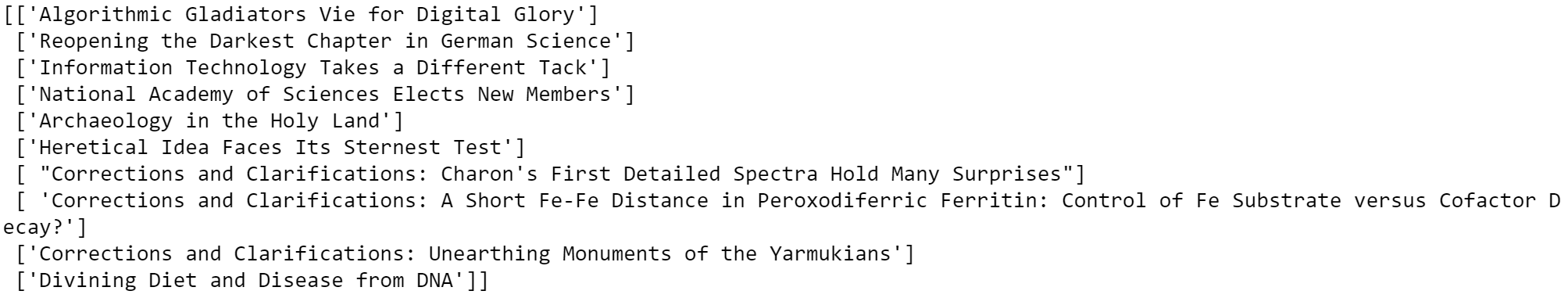


3. Nearest ten documents for each cluster:

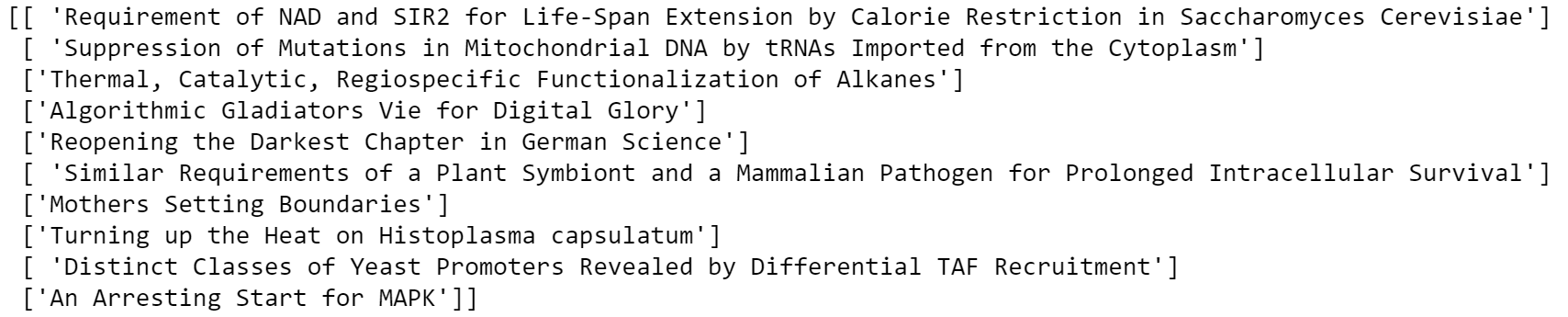
Cluster 1:



Cluster 2:



Cluster 3:

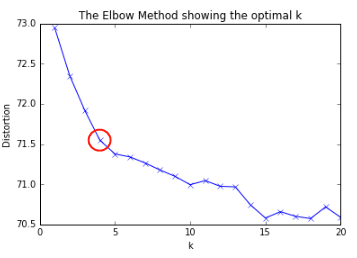


4. Comments for these results:

This algorithm captured the information of the area of topic for each document. So documents, which their topic is most related or in a similar area of study will be clustered into one group. This algorithm might be useful for searching engine or recommendation system, so that user can input several keywords and system can output a group of document that might be related to what user want to get.

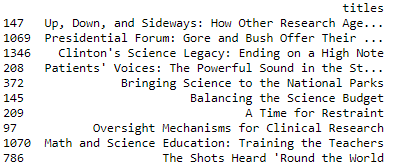
1. 1. Select K:

We select K=4 since according to the inertia of the clustering process, we found that there is a significant elbow at point K = 4 when we plot all the inertia from k = 1 to k = 20.

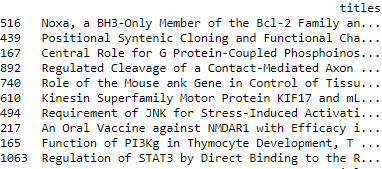


2. Top ten documents for each cluster:

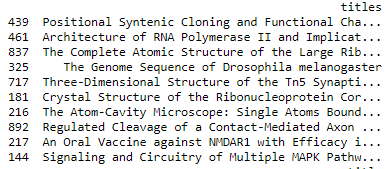
Cluster 1:



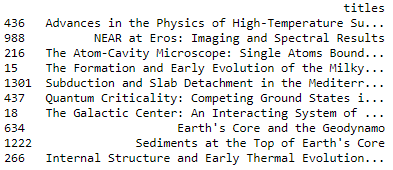
Cluster 2:



Cluster 3:

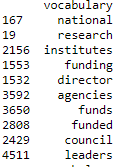


Cluster 4:

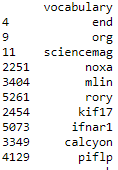


3. Nearest ten words for each cluster:

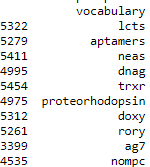
Cluster 1:



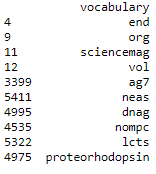
Cluster 2:



Cluster 3:



Cluster 4:



4. Comments for these results:

This algorithm captured the information of the area of topic for each words according to the given documents. So vocabularies that appear similar with similar frequence in all these documents are clustered in one group. By knowing the vector of the vocab frequency among these document, we can know what kind of other words appear with similar frequency in all these documents. It’s hard to come up with a scenario that make this algorithm useful since by knowing the documents list that the words contained, is not that useful to our usual activities. Documents title looks like having much meaningful information than just one words/vocabulary.

Citatition:

1. Removing Punctuation From Python List Items:

<https://stackoverflow.com/questions/4371231/removing-punctuation-from-python-list-items>

1. Python Untokenize a sentence:

<https://stackoverflow.com/questions/21948019/python-untokenize-a-sentence>

1. Pandas read\_csv to deal with un-escaped ‘quotation’ marks.

<https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html>

1. Removing stop words with NLTK in Python:

<https://www.geeksforgeeks.org/removing-stop-words-nltk-python/>

1. Sklearn.cluster.KMeans:

<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

1. Numpy.matrix.mean:

<https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.matrix.mean.html>

1. Searching for k nearest points:

<https://stackoverflow.com/questions/12369484/searching-for-k-nearest-points>

1. Nearest Neighbors:

<https://scikit-learn.org/stable/modules/neighbors.html>

1. Should feature selection be performed only on training data (or all data)?

<https://stats.stackexchange.com/questions/64825/should-feature-selection-be-performed-only-on-training-data-or-all-data>