SVD, LTA, Word Embedding

Student: 程立維
Student ID: 105502556

Implementation

1. Data Preprocessing

```
for i in newsgroups_train.data:
    train = i.lower()
    for c in string.punctuation:
        train = train.replace(c,' ')
    train = ''.join([i for i in train if not i.isdigit()])
    train = ''.join([ps.stem(i) for i in train])
    docu.append(train)
```

```
vectorizer = CountVectorizer(max_df=0.95, min_df=0.001, max_features=3000, stop_words='english')
```

First I will transform the data into lowercase, so if there are terms like ‘Word’ and ‘word’ they mean the same word. Second, I want to remove all the punctuation. There are usually many punctuations in each document, which will also be trained when I load the data into embedding model. Third, I want to remove all the number by the same reason I deal to punctuations. Then, steaming the words. For example, if there are terms like ‘dog’ and ‘dogs’, I hope it represent the same word ‘dog’ except of two different words. Last but not least, remove the stop words because they occur too frequently.

2. Design of SVD, LDA, Word Embedding Model

   SVD:

   After preprocessing the data, I will transform the data into occurrence matrix. Then use the occurrence matrix to calculate the U, sigma, V^T matrix. But these matrices are too big to represent, so we only take U_2, Σ_2, V^T_2. According to the study, the terms can be represent by $U_2\Sigma_2$ and the documents can be represent by $\Sigma_2 V^T_2$.

   LDA:

   Similar to SVD, I first preprocessing the data then transform data into occurrence matrix. I take the occurrence matrix to train LDA model. But I also take the data to train a Word2Vec model for word embedding. I choose to have 20 topics and show 5 top words of each topics. After getting 20 topics with 5 top

words, I use the word embedding model to visualize each words.

Word Embedding Model:
We don't have to train our own Embedding Model, but we have to load a pre-trained model. I load the 'GoogleNews-vectors-negative300' pre-trained model for this part of assignment.

3. Hyper parameters choosing
   A. SVD:
      CountVectorizer's parameters
         Input: data to form occurrence matrix
         max_df: Ignore the terms that occurred in too many documents.
         min_df: Ignore the terms that occurred in too less documents.
         stop_words: Remove all the stop words from all the documents.
         binary: If True, all non zero counts are set to 1.
      Matplotlib's parameters
         marker: Use different shape to represent document and terms.
         c: Use different color to represent document and terms.
   B. LDA:
      CountVectorizer's parameters
         Input: data to form occurrence matrix
         max_df: Ignore the terms that occurred in too many documents.
         min_df: Ignore the terms that occurred in too less documents.
         Stop_words: Remove all the stop words from all the documents.
      Matplotlib's parameters
         marker: Use different shape to represent document and terms.
         c: Use different color to represent document and terms.
      LatentDirichletAllocation's parameters
         n_components: number of topics.
         max_iter: maximum number of iterations.
      Word2Vec parameters
         sentences: data we want to train.
         size: the dimension of the word-vectors.
         window: how many words before and after to look up to.
         min_count: If the word appeared less than min_count then we won't train this word.
   C. Word embedding model:
      load_word2vec_format parameters
         fname: The file path used to save the vectors in.

binary: If True, the data will be saved in binary word2vec format.

limit: Sets a maximum number of word-vectors to read from the file.

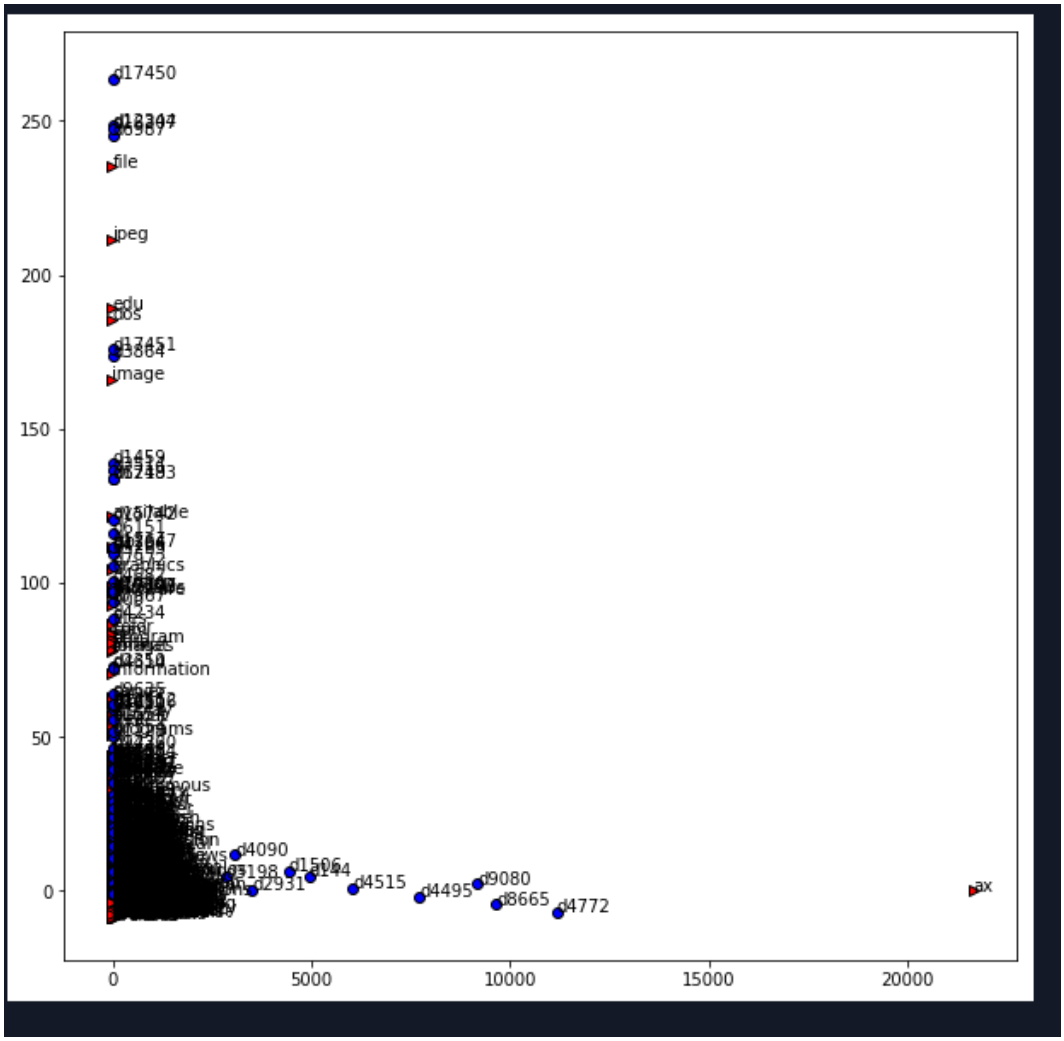
Matplotlib's parameters

marker: Use different shape to represent document and terms.

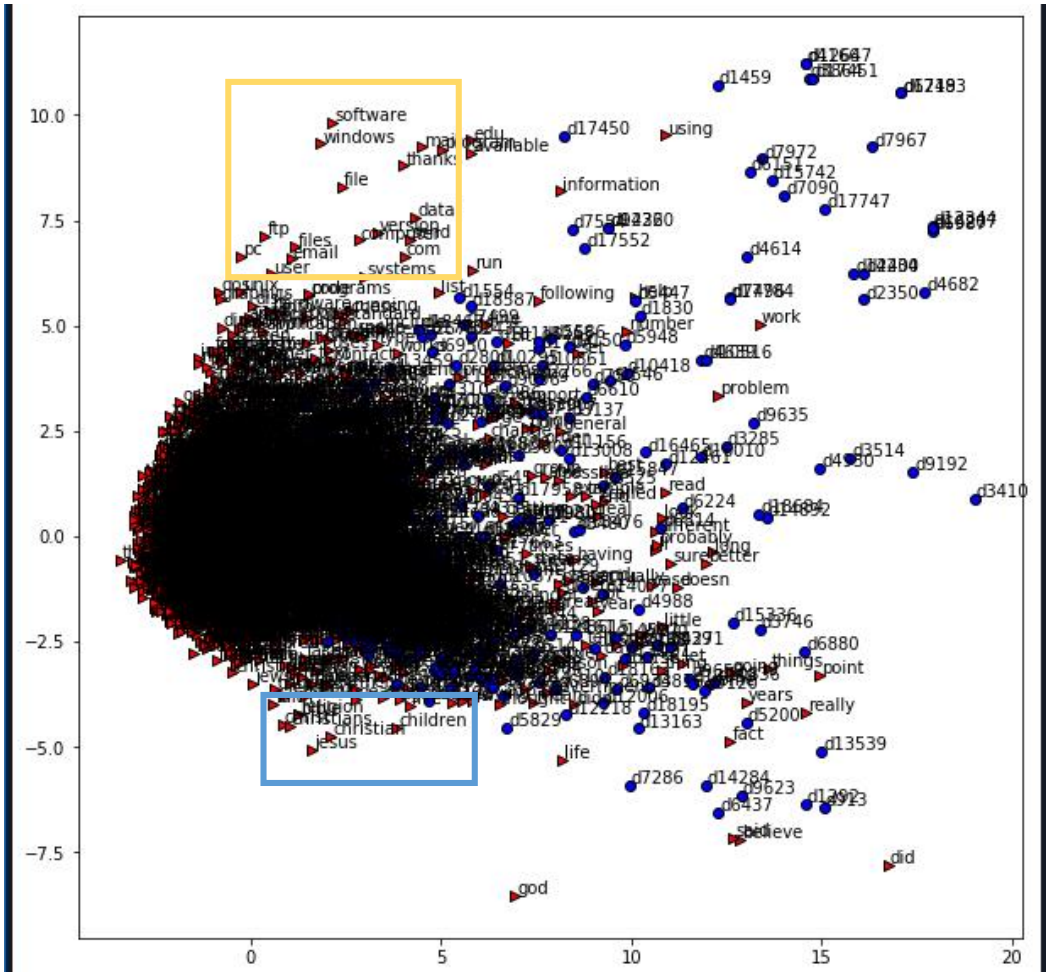
c: Use different color to represent document and terms.

Experiment for the 3 Task

SVD: So at first, I try to visualize the document and terms vector and appears some problems. As you can see most of the vectors are on a line, but several vectors are far away from them. (blue for documents and red for terms)

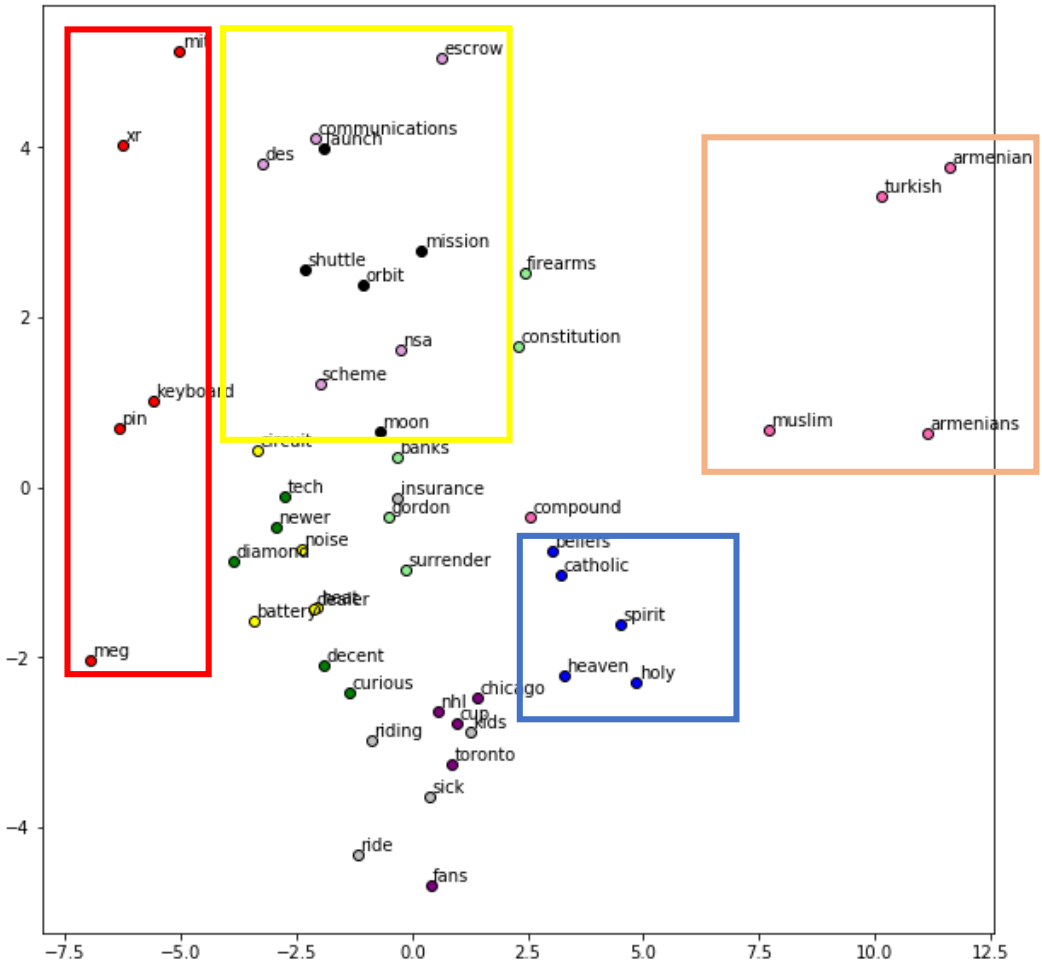


Later on, I try use the parameter 'binary=True'. According to the sklearn website, it said that we can set the parameter 'binary' to True for discrete probabilistic models. And it seems to work better than the original one. We can also see some relative words are put together. For example, inside the top yellow square we can find: software, window, data, email, file, etc. And the bottom square we can find: children, jesus, christian, etc.

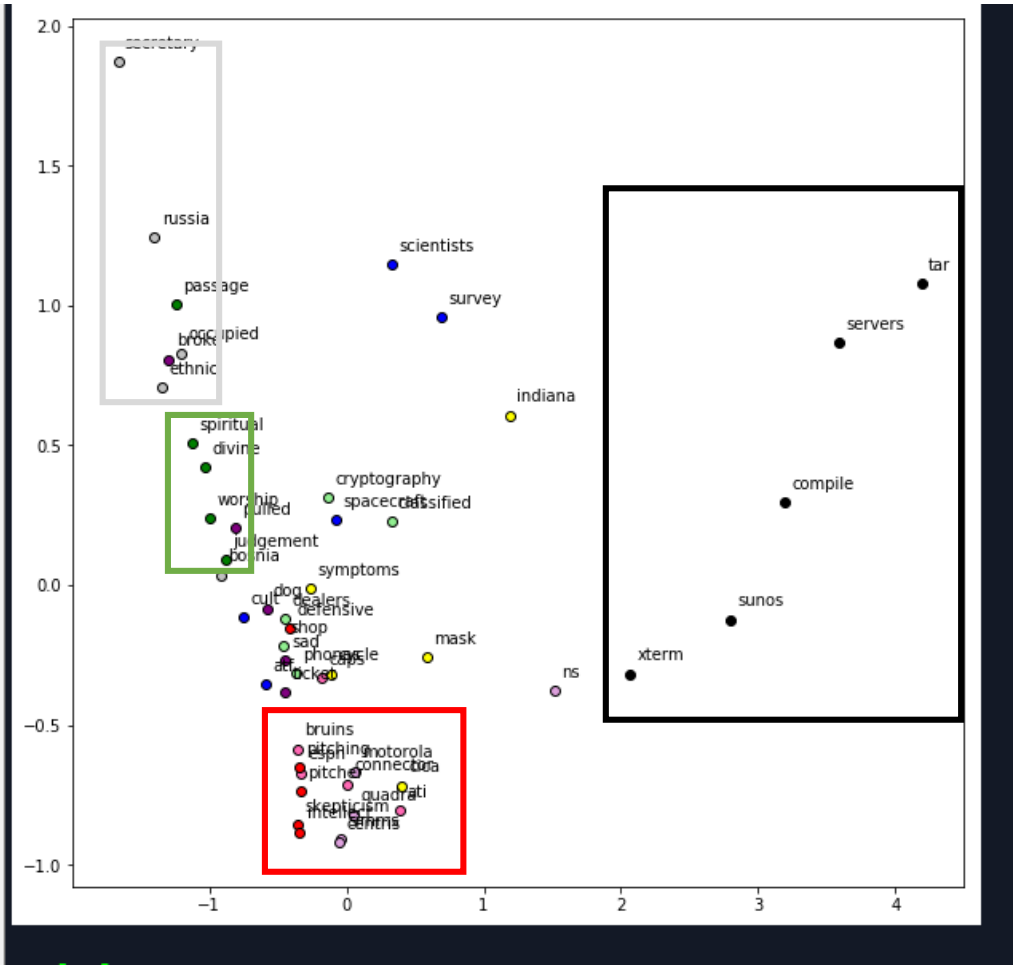


LDA:

In this task, I had to train two models. One for LDA, and another for word-embedding. I choose to use 10 topics, because it is better to see the grouping. 20 topics will be too much and hard to recognize the group because it is too messy. So, the difficult part of this task is because I have to use the top words for each topic from LDA, and represent by word-embedding model in 2-D space. First, I try to use Word2Vec as word embedding model and the 2-d space looks like this.



So I also try to use SVD's word vector to represent the top words of each topics. And I think that this is a great way, too. Because you can see some same color vectors gather together.



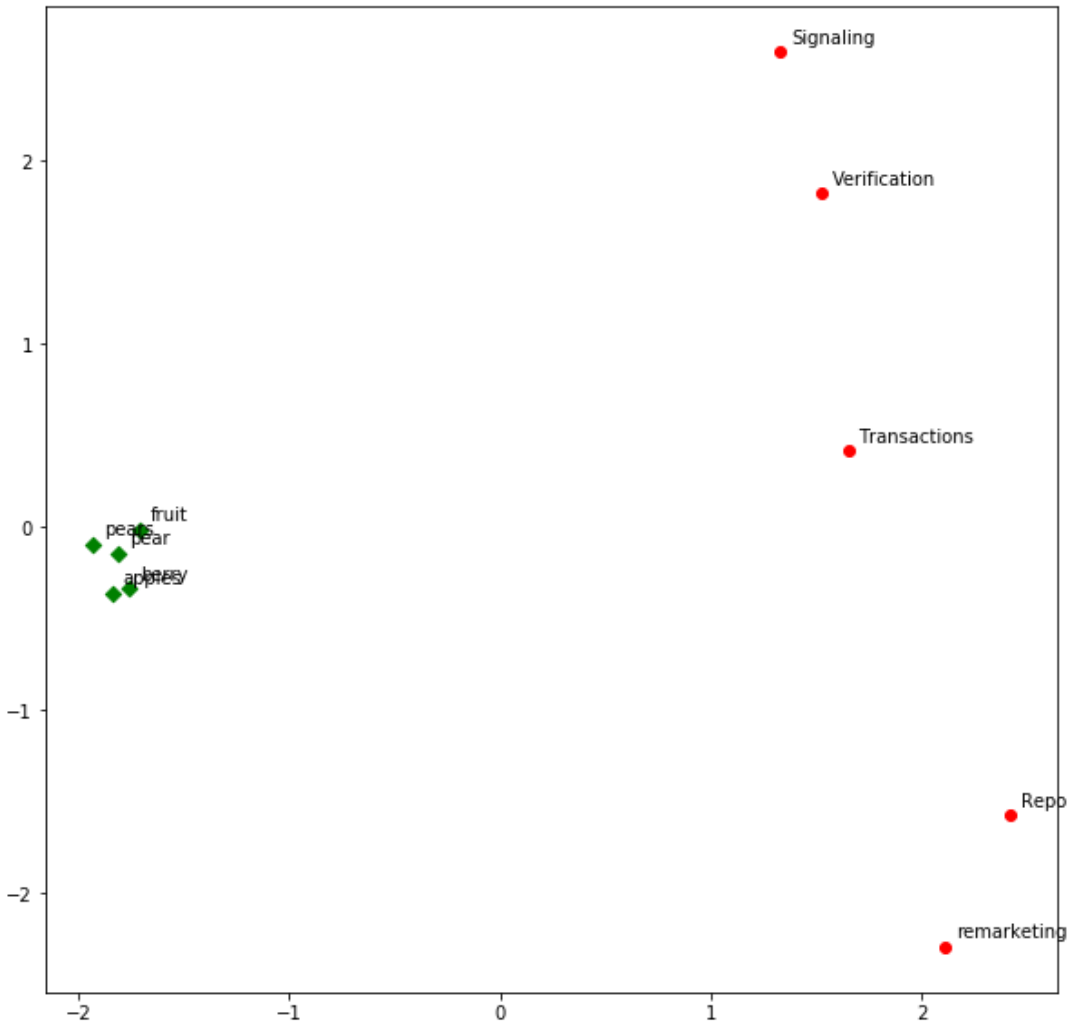
Furthermore, I try to use different value for max_df. I found out that if I take max_df=0.9, there will be same word in different topics. So I reduce the value of max_df to 0.1, which will not happen the same problem again.

Word embedding model:

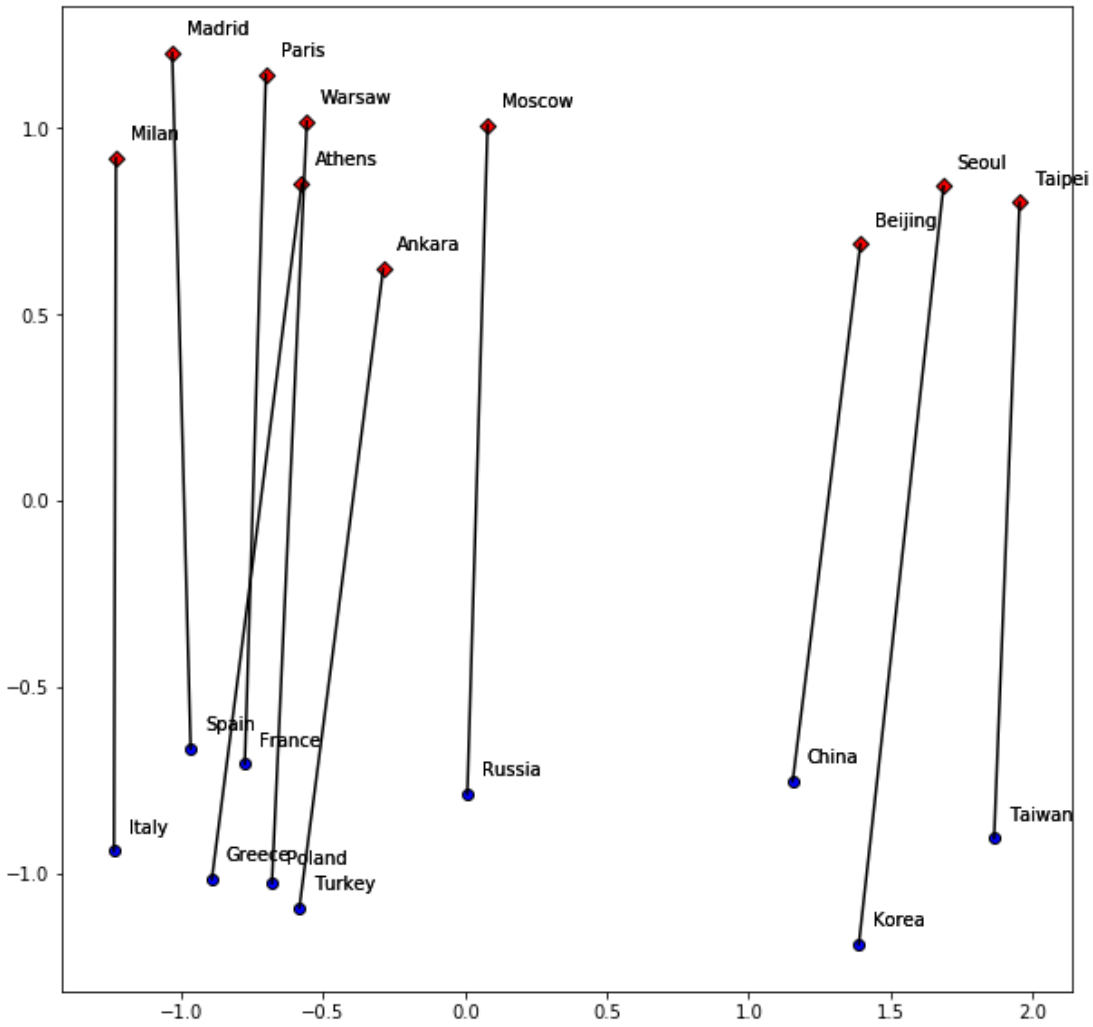
At first I try to load the data, but the memory of my computer isn't big enough to handle this much data so it cause 'memory error'. I use the parameter 'limit' to limit the data size.

```
limit=100000)
```

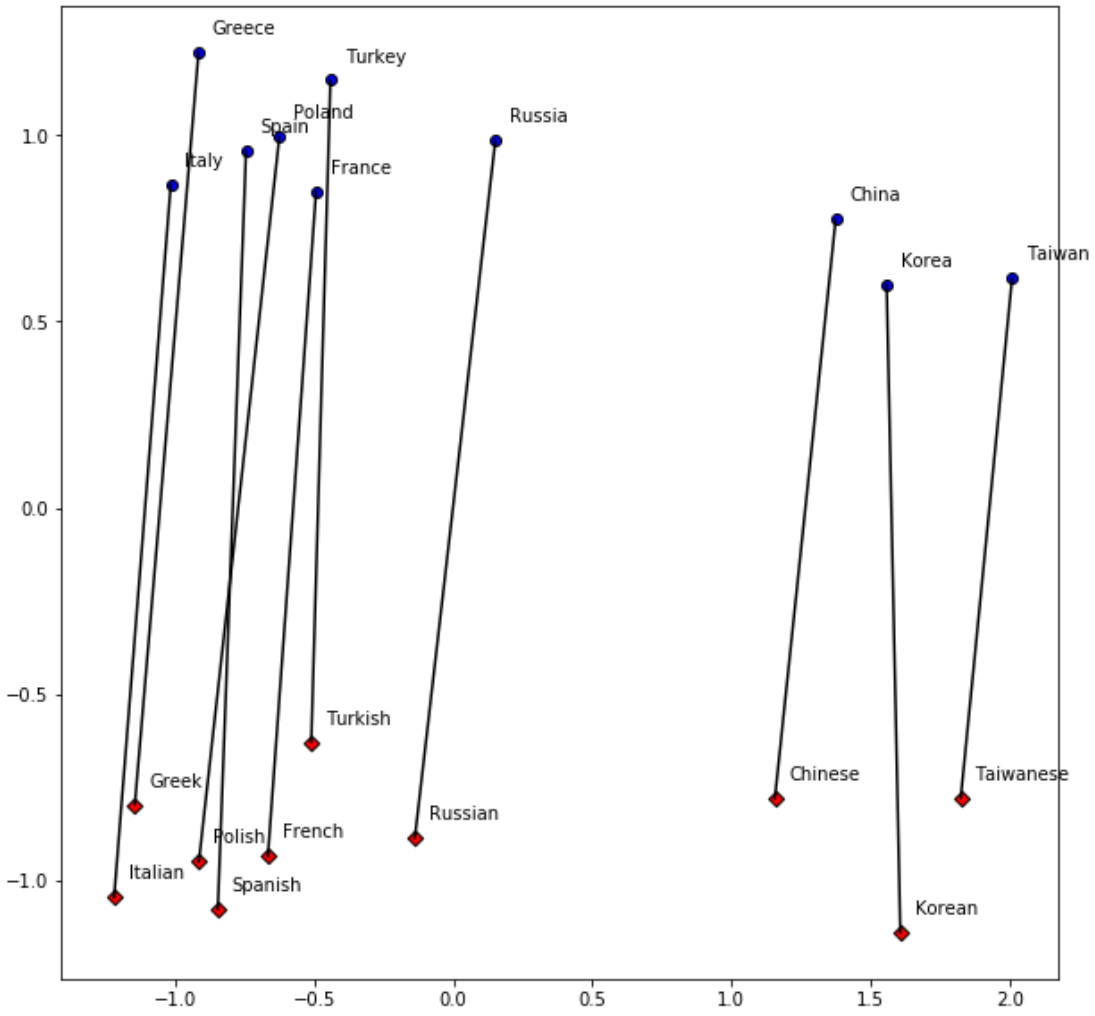
First section is to find the 5 most and least simialr words for apple and try to visualize it. My result is underneath. I use green diamond dot as 5 most similar words and red circle dot as least similar words. As you can see the 5 most similar words gathers in a small place while 5 least similar words are far away.



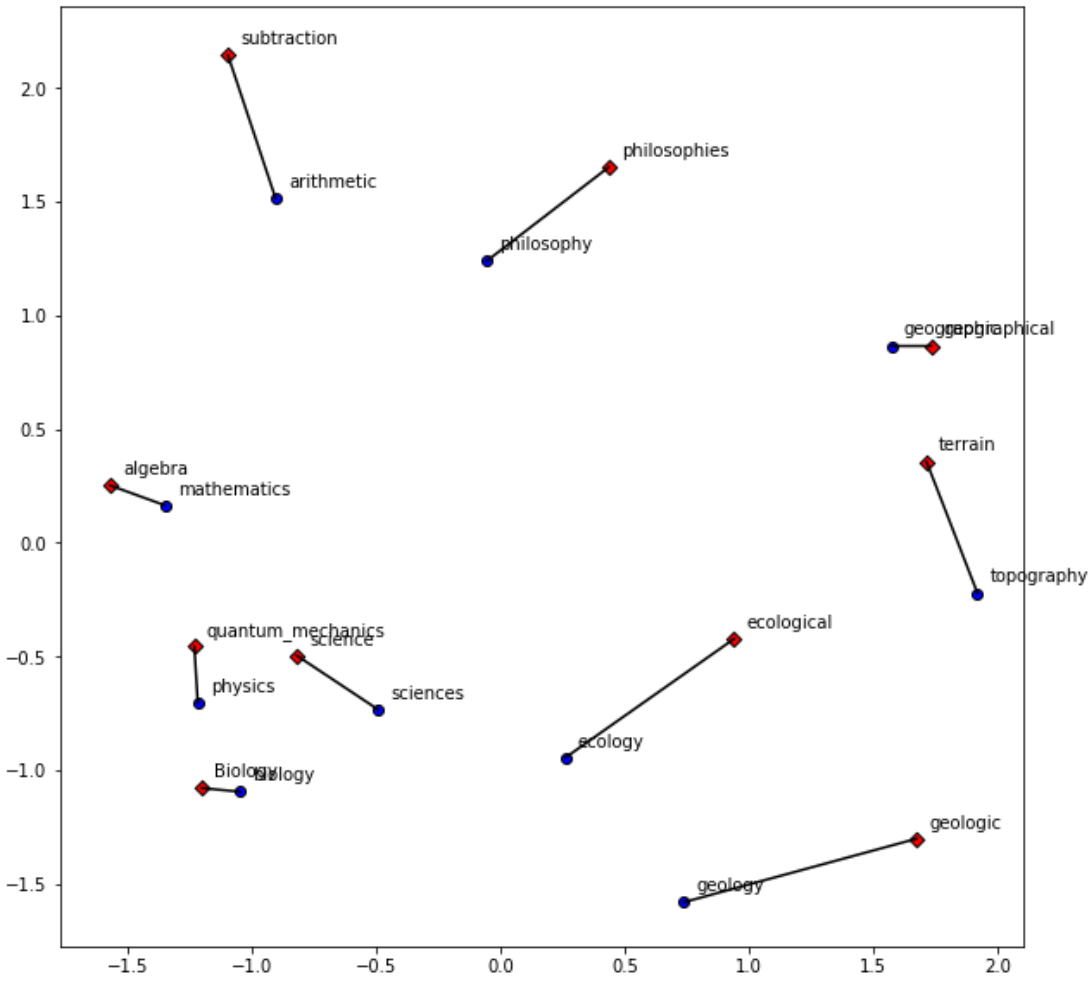
The next section is to find the capital of each country according to the relationship of Japan and Tokyo. I use blue dot as the countries and red dot as the capitals. And use black line to link the city and capital pairs. As you can see, capitals are all on the top of the 2d-space while the countries are all under the 2d space. And the capitals are mostly right above the countries.



Last section is to create 10 word pairs according to another relationship. First I want to use country and language as 10 word pairs and the result is underneath.



But I think this is too simialr to the last one. So I try to use ‘Course’ and ‘subject’ from the relationship of ‘math’ and ‘mathematics’. I use blue dot as course and red diamond dot as subject, then link them up by black line. The result is quite interesting, because it really find the relationship from math and mathematics. Most of the given course have great corresponding subject.



Work distribution

Division: 程立維 100%

Contribution: 程立維 100%