Assignment 1 Report

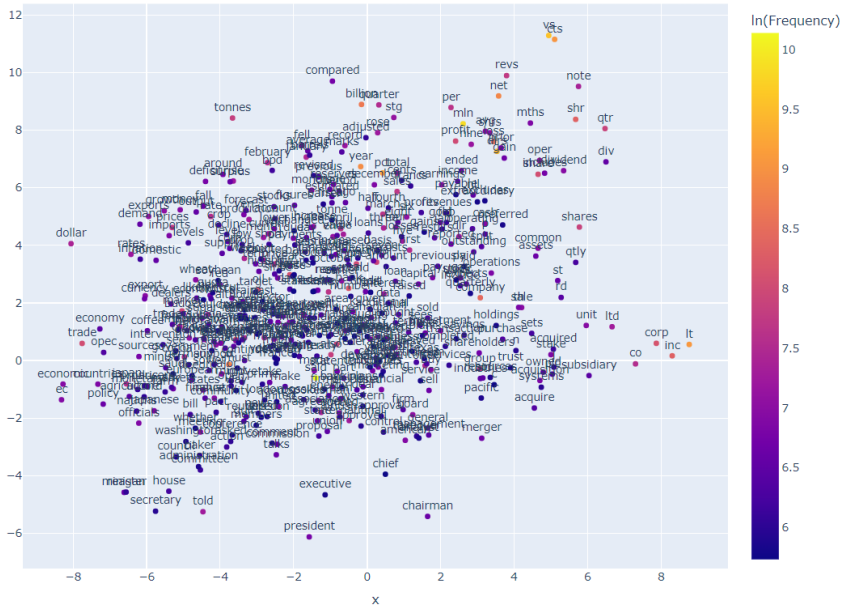
Name: Hongzhao Tan

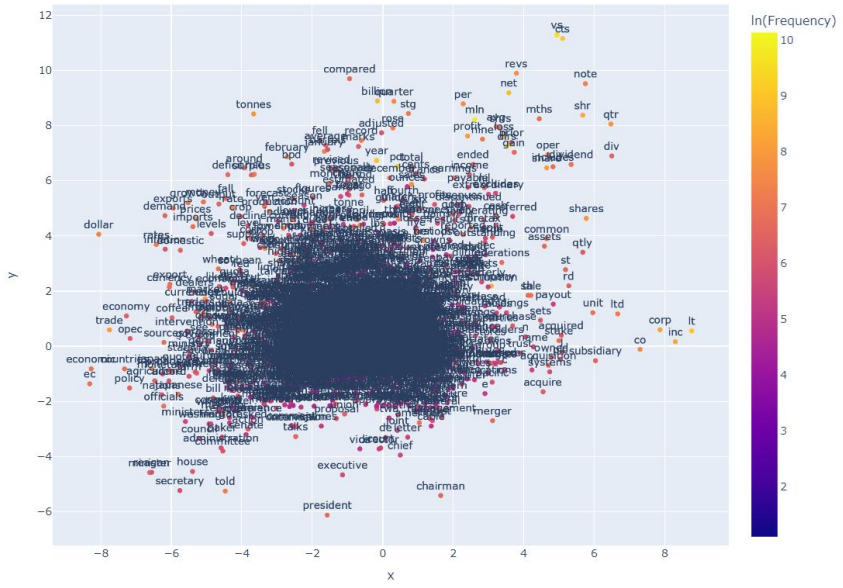
MacID: Tanh10

Student #: 400136957

Question 1:

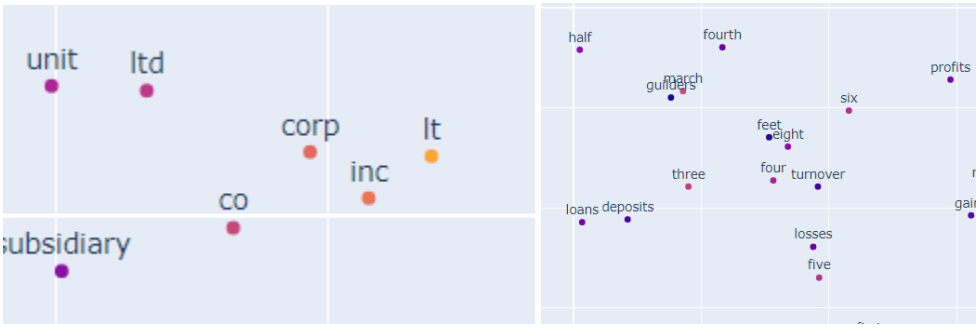
After creating word vectors using Gensim Word2Vec model and Reuters corpus, reducing the dimension of the vectors down to 2 and finally visualizing the 2-dimensional word vectors, following are the annotated scatter plots that has been respectively created using Plotly for the 500 most frequent words in the corpus and all the 14273 words that have appeared in the corpus after preprocessing.





As shown in the two scatter plots above, the points in the plots are distributed similarly to a Gaussian distribution (the points are denser as we are getting closer to the center of the distribution) with a bit of correlation between the 2 dimensions that the word vectors have been reduced to.

By density, there is not any cluster of words can be found in the scatter plot other than and because of the single giant dense cluster of words at the center. However, there are still some patterns can be found from the plots. By observing the colors of the word points which indicate the number of occurrences of the corresponding words in the corpus, we can see that most of the words whose corresponding points are placed away from the center of overall distribution are having high numbers (hundreds or thousands) of occurrences in the corpus. There are also some groups of words that are semantically related with their points located relatively close to each other. For instances, the points of words 'ltd', 'corp' and 'inc' which usually take place at the end of companies' names, and the points of words 'three, 'four', 'five', 'six' and 'eight' which are words of numbers, are located close to each other.



Question 2:

6 analogy tasks have been conducted on the pre-trained 100-dimensional GloVe word vectors that was trained using "Wikipedia 2014" and "Gigaword 5" corpuses. The tasks and their results are as followed:

```
In [24]: find_closest_word((glove_dict['king']-glove_dict['man']+glove_dict['woman']),glove_dict)[:5]
Out[24]: ['king', 'queen', 'monarch', 'throne', 'daughter']

In [25]: find_closest_word((glove_dict['king']-glove_dict['father']+glove_dict['son']),glove_dict)[:5]
Out[25]: ['king', 'prince', 'son', 'nephew', 'throne']

In [26]: find_closest_word((glove_dict['woman']-glove_dict['singular']+glove_dict['plural']),glove_dict)[:5]
Out[26]: ['woman', 'mother', 'girl', 'pregnant', 'boy']

In [27]: sim_rank = find_closest_word((glove_dict['woman']-glove_dict['singular']+glove_dict['plural']),glove_dict).index('women')
         print("The vector for 'women' is the number {} most similar vector in GloVe to 'woman' - 'singular' + 'plural'".format(sim_rank+1))
         The vector for 'women' is the number 11 most similar vector in GloVe to 'woman' - 'singular' + 'plural'

In [28]: find_closest_word((glove_dict['find']-glove_dict['present']+glove_dict['past']),glove_dict)[:5]
Out[28]: ['past', 'find', 'back', 'away', 'trying']

In [29]: sim_rank = find_closest_word((glove_dict['find']-glove_dict['present']+glove_dict['past']),glove_dict).index('found')
         print("The vector for 'found' is the number {} most similar vector in GloVe to 'find' - 'present' + 'past'".format(sim_rank+1))
         The vector for 'found' is the number 726 most similar vector in GloVe to 'find' - 'present' + 'past'

In [30]: find_closest_word((glove_dict['finger']-glove_dict['hand']+glove_dict['foot']),glove_dict)[8:15]
Out[30]: ['toes', 'calf', 'groin', 'bruise', 'wrist', 'ankle', 'toe']

In [31]: sim_rank = find_closest_word((glove_dict['white']-glove_dict['black']+glove_dict['yang']),glove_dict).index('yin')
         print("The vector for 'yin' is the number {} most similar vector in GloVe to 'white' - 'black' + 'yang'".format(sim_rank+1))
         The vector for 'yin' is the number 62 most similar vector in GloVe to 'white' - 'black' + 'yang'
```

Details of the find_closest_word function which was used when conducting the analogy tasks can be found in the submitted Jupyter Notebook.

Explanation to the results:

Let a, b, x and y be four words and A, B, X and Y be their corresponding word vectors respectively. In general, a analogy task of (X – A + B) = Y can be transferred into a statement of the form 'x is to y as a is to b'. Mathematically, an analogy tasks starts by calculating (X – A + B) where A, B and X are three word vectors of the same length (100 in this case, given the GloVe is 100-dimensional). Then, with a new vector Y-hat that is resulted from (X – A + B), we can compute either the similarity between Y-hat and a pre-determined word vector Y with certain similarity/disimilarity function (e.g., dot product, cosine similarity, etc.), or the similarities between Y-hat and all word vectors we have to check how close is Y-hat to Y comparing to other word vectors by sorting the word vectors according to the computed similarities. Both of these two approaches are to quantify how well the trained word vectors is performing on the analogy task(s) which could infer the word vectors' performance on capturing semantic/syntactic relationships between words.

The first two analogy tasks (shown in cells 24 and 25 in the screenshot above) that have been conducted are "king – man + woman ≈ queen" and "king – father + son ≈ prince". These two tasks are both to test the word vectors' performance on capturing the semantic relationships between words. The results of the first two tasks are relatively good, since the word vectors of words 'queen' and 'prince' are both the second most similar word vectors to the resulting vectors of "king – man + woman" and "king – father + son" respectively.

The third and fourth analogy tasks (shown in cells from 26 to 29 in the screenshot above), which are "woman – singular + plural ≈ women" and "find – present + past ≈ found" respectively, are to check the word vectors' performance on capturing the syntactic relationships between words. The word vectors have performed relatively badly on these two tasks, which showed that the word vectors are not capturing syntactic relationships well, especially for the syntax of tenses of verbs. The vectors of 'women' and 'found' are the 11^{th} and 726^{th} nearest word vectors to the resulting vectors of "woman – singular + plural" and "find – present + past" respectively.

The last two analogy tasks (shown in cells 30 to 31 in the screenshot above) are "finger – hand + foot ≈ toe" and "white – black + yang ≈ yin".

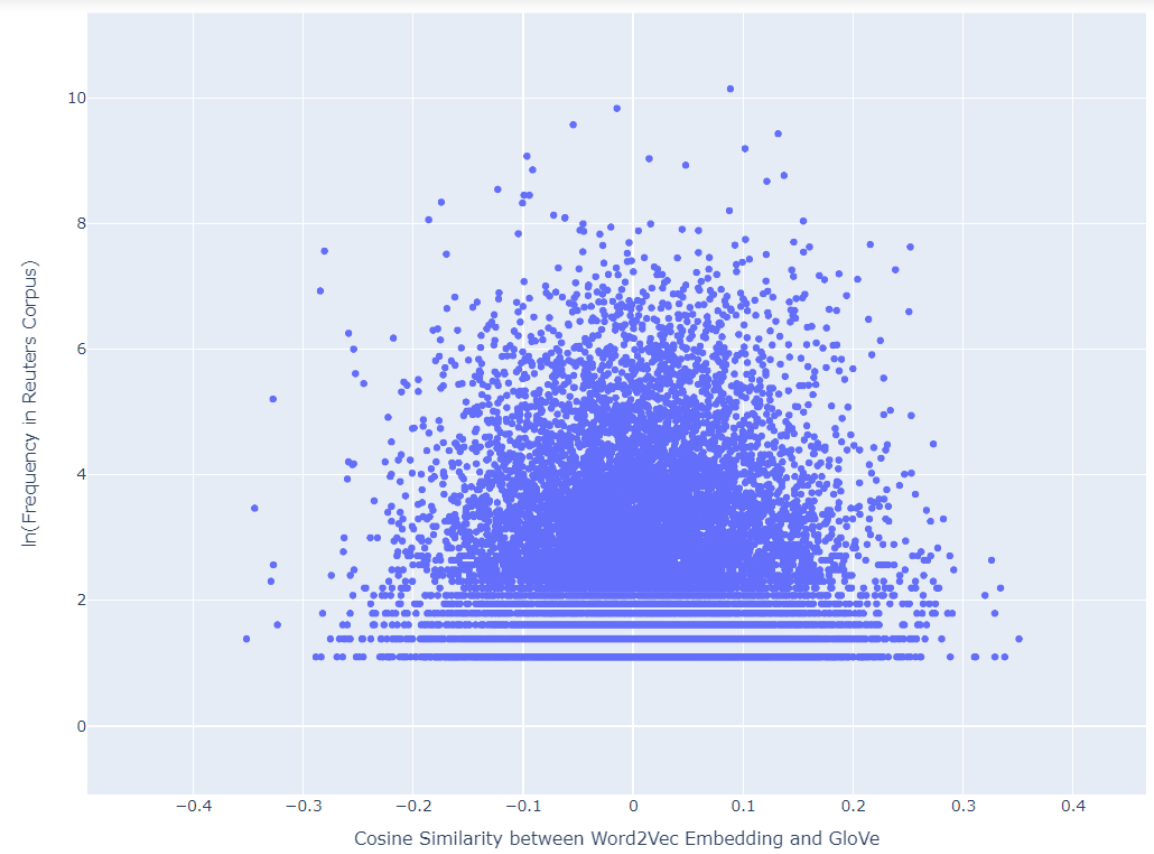
The task of "finger – hand + foot ≈ toe" is to test if word(s) with different meanings would affect word vectors' perfomance on an analogy task. Other than the meaning of a body part, the word "foot" is also commonly used as a unit of length/height. Hence, the vector of "foot" will also encode the semantic information of "foot" as a length/height unit which is completely different from the information of a body part, since the context (set of words that appear nearby) around "foot" usually differs a lot with different meanings of "foot". As a result, the vectors of words 'toes' and 'toe' are the nineth and 15^{th} most similar word vectors to the resulting vector of "finger – hand + foot" respectively, which proves that word(s) with different meanings could negatively affect perfomance on analogy tasks.

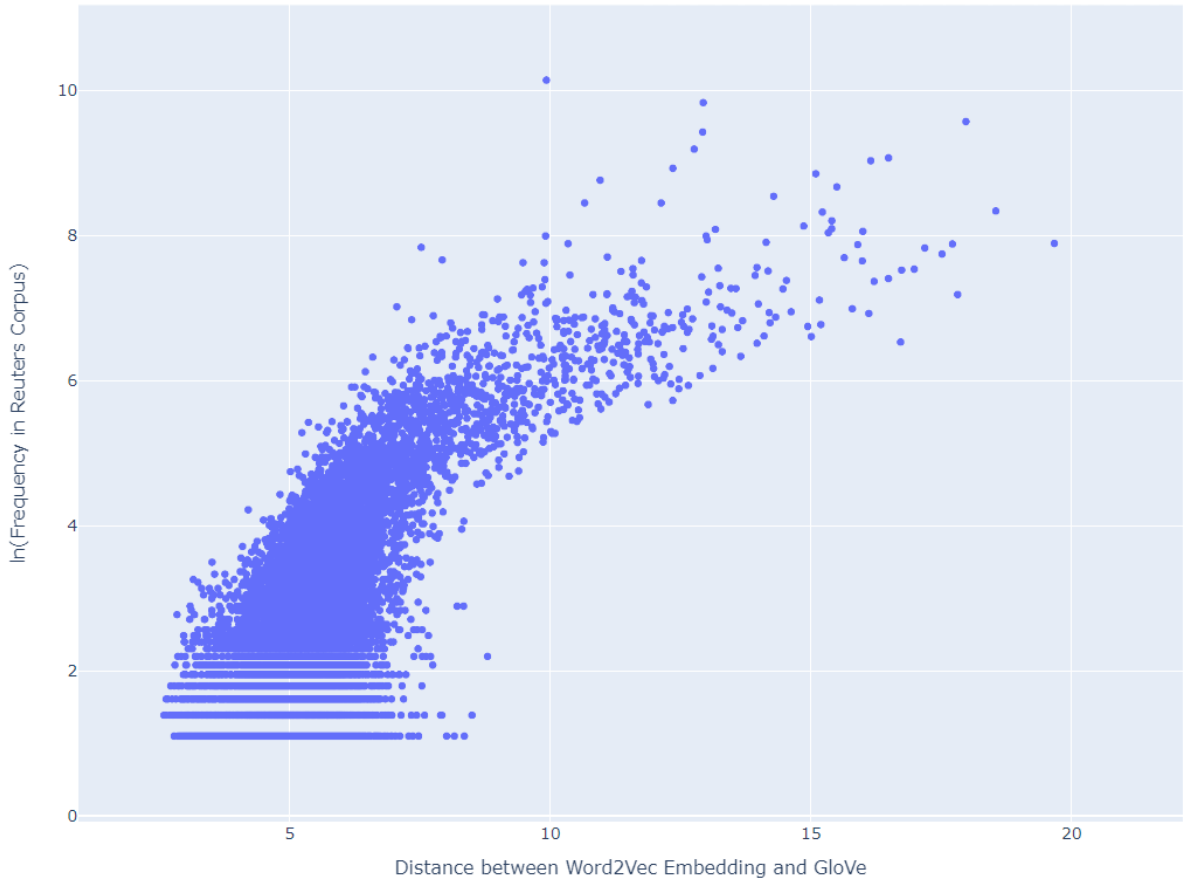
The "white – black + yang ≈ yin" task is to check the word vectors' performance on analogy task with words that are not really English. The result that the word vector of "ying" is the 62^{nd} nearest word vector to the vector resulted from "white – black + yang", shows that word vectors that is trained with English corpus would not encode the semantic information of non-English words as effectively as normal

English words, even though the Chinese philosophy concept of YinYang is worldwidely well-known. This could because the words “yin” and “yang” appear much less often than actual English words in an English corpus, and “yin” and “yang” could also appear with different meanings in other contexts such as Chinese names.

Question 3:

To compare the word vectors that was trained with Gensim Word2Vec and Reuters corpus with the pre-trained GloVe word vectors, Cosine similarities and Euclidean distances between the word vectors of each of the common words that have appeared in both Reuters corpus and GloVe, have been computed and plotted against the number of times these common words have taken place in Reuters corpus. Furthermore, two analogy tasks “king – father + son ≈ prince” and “find – present + past ≈ found” have been conducted on the word vectors trained with Word2Vec model to test the vectors’ performance on capturing semantic and syntactic relationships between words. The results were compared with the results from conducting the same two analogy tasks using the GloVe word vectors. The plots and the results of the analogy tasks are as follows.





In [38]:
```
sim_rank = find_closest_word((model.wv['king']-model.wv['father']+model.wv['son']),
                  dict(zip(model.wv.key_to_index.keys(), model.wv.vectors))).index('prince')
print("The vector for 'prince' is the number {} most similar vector trained by Word2Vec to 'king' - 'father' + 'son'".format(sim_rank+1))
```

```
The vector for 'prince' is the number 70 most similar vector trained by Word2Vec to 'king' - 'father' + 'son'
```

In [39]:
```
cosine_sim = cos_sim((glove_dict['king']-glove_dict['father']+glove_dict['son']), glove_dict['prince'])
print("The cosine similarity between the GloVe word vectors 'prince' and 'king' - 'father' + 'son' is {0:0.3f}".format(cosine_sim))
```

```
The cosine similarity between the GloVe word vectors 'prince' and 'king' - 'father' + 'son' is 0.798
```

In [40]:
```
cosine_sim = cos_sim((model.wv['king']-model.wv['father']+model.wv['son']), model.wv['prince'])
print("The cosine similarity between the Word2Vec word vectors 'prince' and 'king' - 'father' + 'son' is {0:0.3f}".format(cosine_sim))
```

```
The cosine similarity between the Word2Vec word vectors 'prince' and 'king' - 'father' + 'son' is 0.827
```

In [41]:
```
sim_rank = find_closest_word((model.wv['find']-model.wv['present']+model.wv['past']),
                  dict(zip(model.wv.key_to_index.keys(), model.wv.vectors))).index('found')
print("The vector for 'found' is the number {} most similar vector trained by Word2Vec to 'find' - 'present' + 'past'".format(sim_rank+1))
```

```
The vector for 'found' is the number 6379 most similar vector trained by Word2Vec to 'find' - 'present' + 'past'
```

In [42]:
```
cosine_sim = cos_sim((glove_dict['find']-glove_dict['present']+glove_dict['past']), glove_dict['found'])
print("The cosine similarity between the GloVe word vectors 'found' and 'find' - 'present' + 'past' is {0:0.3f}".format(cosine_sim))
```

```
The cosine similarity between the GloVe word vectors 'found' and 'find' - 'present' + 'past' is 0.464
```

In [43]:
```
cosine_sim = cos_sim((model.wv['find']-model.wv['present']+model.wv['past']), model.wv['found'])
print("The cosine similarity between the Word2Vec word vectors 'found' and 'find' - 'present' + 'past' is {0:0.3f}".format(cosine_sim))
```

```
The cosine similarity between the Word2Vec word vectors 'found' and 'find' - 'present' + 'past' is 0.320
```

Analysis on the scatter plots:

The first scatter plot above has shown that there is no obvious correlation between the Consine similarties between word vectors trained by Word2Vec model and pre-trained GloVe, and the frequencies of the words in Reuters corpus (note that the frequency is transferred into log scale because the value varies from less than 10 up to tens of thousands). Interestingly, the second scatter plot above has shown a positive correlation between the Euclidean distances between the end points of the Word2Vec-trained word vectors and pre-trained GloVe, and the frequencies of the words in Reuters corpus.

As discussed in Question 1, most of the words whose corresponding end points their word vectors are placed away from the center of the points' distribution, which is close to the origin point of the space of the vectors, are having high numbers (hundreds or thousands) of occurrences in the corpus. This could enlarge the Euclidean distances between the end points of the vectors of highly frequent words in Reuters corpus, because the absolute values of the elements in these vectors and so the length/L2-norm of these vectors will become bigger, while the cosine similarities might stay the same. For example, the Euclidean distances between vectors, $[1,3,5]^T$ and $[4,5,2]^T$, and $[1,3,5]^T$ and $[8,10,4]^T$ are about 4.7 and 10 respectively, while the cosine similarities between these two pairs of vectors are both equal to 0.731.

Analysis on the conducted analogy tasks:

As results of the analogy tasks "king – father + son ≈ prince" and "find – present + past ≈ found" that have been conducted on the word vectors trained from Word2Vec model and Reuters corpus, the word vectors of "prince" and "found" are the 70^{th} and 6379^{th} nearest vectors to the vectors resulted from "king – father + son" and "find – present + past" respectively. These results are much worse than when we conduct the same two tasks on the pre-trained GloVe (second and 726^{th} respectively), as shown in the answer for Question 2.

Nevertheless, if we only compare the consie similarities between "prince" and "king – father + son", and between "found" and "find – present + past", that were computed using the Word2Vec-trained vectors, to the consie similarities that were computed using the pre-trained GloVe, it has shown that the difference was not as significant (0.798 vs. 0.827 and 0.464 vs 0.320 respectively, as shown in the code cells 39, 40, 42 and 43 above). Suprisingly, the consine similarity of task "king – father + son ≈ prince" using Word2Vec-trained vectors was slightly higher than that of using pre-trained GloVe.

This result from comparing these two models' word vectors might due to 2 reasons as follows:

1. As shown in the answer for Question 1, there is a giant dense cluster on the center of the word vectors' end points' overall distribution. Because the endpoints of word vectors are located extremely close to each other, the vectors of words other than the words that participated in the analogy tasks could have introduced a lot of noises to the rank of cosine similarities of the target word vectors "prince" and "found".
2. The corpus which the pre-tained GloVe was trained with was much larger than the Reuters corpus. Solely the Gigaword 5 corpus already contains nearly 10 million text documents, and the pre-tained GloVe has a vocabulary of 400 thousand words, while the Reuters corpus has only about 10 thousand documents and slightly over 14 thousand unique word tokens after cleaning the text data. This tremendous difference on the sizes of corpus and vocabulary

could lead to plenty different contexts/combinations of words around certain word appear much less often or do not even exist in the smaller corpus. Because a word's meaning is given by the context around it, learned by the Word2Vec model and finally encoded in the word vectors. Corpus of smaller size could not be capable of letting the model to learn some words' semantic/syntactic relationships with other words.