CS539 - NLP - Homework 1

Nuttaree Busarapongpanich

1 Getting Used to Handling Language Data

1.1

In the `Vocabulary.py` file, I have implemented the tokenize function splitting a string into multiple words and return as an array of words. I am using nltk package to lemmatize the word. I am starting by removing punctuation and capitalization. After that, I split the string to an array and lemmatize each word from that array to be my token. I found that the lemmatization of nltk package will be more accurate if we specify the POS-tag in the lemmatize function. For example,

The lemmatization without POS-tag:

```
> vocab.tokenize("The quick, brown fox jumped over the lazy dog on the rocks.")
['the', 'quick', 'brown', 'fox', 'jumped', 'over', 'the', 'lazy', 'dog', 'on', 'the
  ', 'rock']
```

We can see that the word ***jumped*** is **not** changed to ***jump*** as we expect.

The lemmatization with POS-tag:

```
> vocab.tokenize("The quick, brown fox jumped over the lazy dog on the rocks.")
['the', 'quick', 'brown', 'fox', 'jump', 'over', 'the', 'lazy', 'dog', 'on', 'the',
    'rock']
```

We can see that the word ***jumped*** is changed to ***jump*** as we expect.

1.2

Please see in the code for this task.

1.3

In this part, I apply the frequency cutoff to avoid features of words that rarely occur. The more features the more complex of the program. The low occurrence word will have less support, so we can get rid of them. Moreover, some low frequency words would even be typo which could affect our model. If we do not remove these words, we possibly get overfit to the model.

For this assignment, I only keep words that have frequency more than or equal to 50, which yields around 91.5% coverage of the whole corpus. The reason I use freq=50 is it cover more than 90%, which is high enough to not lose important words (words that show up frequently) and not too high to ruin the performance of the algorithm. After running the program, this 91.5% word coverage gives the reasonable result and satisfies the expectation.

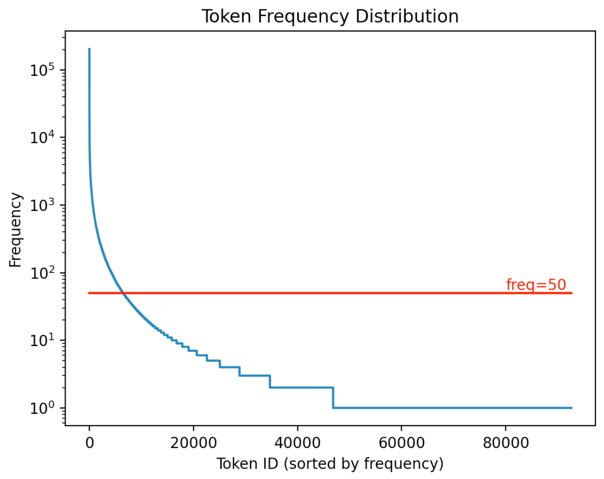

Figure 1: Token Frequency Distribution graph

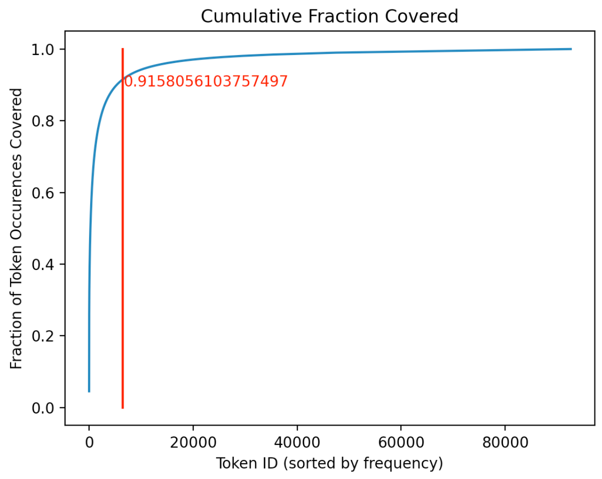

Figure 2: Cumulative Fraction Covered Graph

2 Frequency-Based Word Vectors – PPMI

2.1

Q: What are the minimum and maximum values of PMI (Eq. 1)?

A: The minimum value of PMI is $-\infty$, while the maximum value of PMI is min[-log p(w_i), -log p(w_j)]. However, when we take a look closely, we can see that when log p(w_i) and log p(w_j) go to the $-\infty$, we will get the upperbound as $+\infty$. Theoretically, when p(w_i) and p(w_j) approach to 0, the limit of -log p(w_i) and -log p(w_j) will go to $+\infty$.

Q: If two tokens have a positive PMI, what does that imply about their relationship?

A: A positive PMI value shows a high association between two tokens (w_i and w_j) so the probability of P(w_i, w_j) is higher than P(w_i)P(w_j). From the equation 1, we can see that the numerator will be higher than denominator. Hence, the PMI value will be positive.

Q: If they have a PMI of zero?

A: From equation 1, the log returns 0 when P(w_i, w_j) = P(w_i)P(w_j). It means that 2 words are statistically independent. In other word, there is no interesting relationship between 2 words.

Q: What about if the PMI is negative?

A: When the equation 1 give the negative result, we know that P(w_i, w_j) < P(w_i)P(w_j). We can imply that either w and c, or one of them occur independently more than mutually.

Q: Based on your answers, what is an intuition for the use of PPMI?

A: PPMI is normally used to measure the association between words, where we get rid of the negative values and replace them with 0. There are some reasons that we do not use the negative values. We are unsure about the cooccurrence in a document. If 2 words does not occur together in this document, how much we can be sure that they will not occur in other documents. We need huge corpora to make the negative values reliable. They might only cooccur less than we expect only for this document. However, if 2 words occur together in one document, they are more likely to occur in other documents as well.

2.2

In this section, I define my context by using k-Window by having k=5; I consider 5 neighbors on either side, so each time, I consider 11 words. We should define the proper window size as if it is too small, it would only denote the grammatical structure. However, if I use the whole document, especially when the document is large, I would get too broad structure. When I apply this size to the program, I got the reasonable result as my clusters make sense as we can see in 2.4.

I have tried k=2, 3 and some others, the best clusters I have got from the graph is where k=5.

2.3

For the `compute_ppmi_matrix` function, I have add a small constant to C to prevent log(0). Please see the code for the implementation of this task.

2.4

Figure 3 shows the whole plot of the visualized word vectors as a result from `build_freq_vectors.py`. After zoom in the graph, we can identify at least 3 clusters. The first cluster is shown in Figure 4, which represent the law category. We can see *prison, lawsuit, fraud, charge, settle, investigation* etc., for example. The next cluster in Figure 5 presents the number category including *many numbers(1,2,3,4,5,...), number as texts(three, four, five, six...), number*, and so on. Figure 6 shows country and city category which contains the word like *german, france, korea, moscow, london, china...*

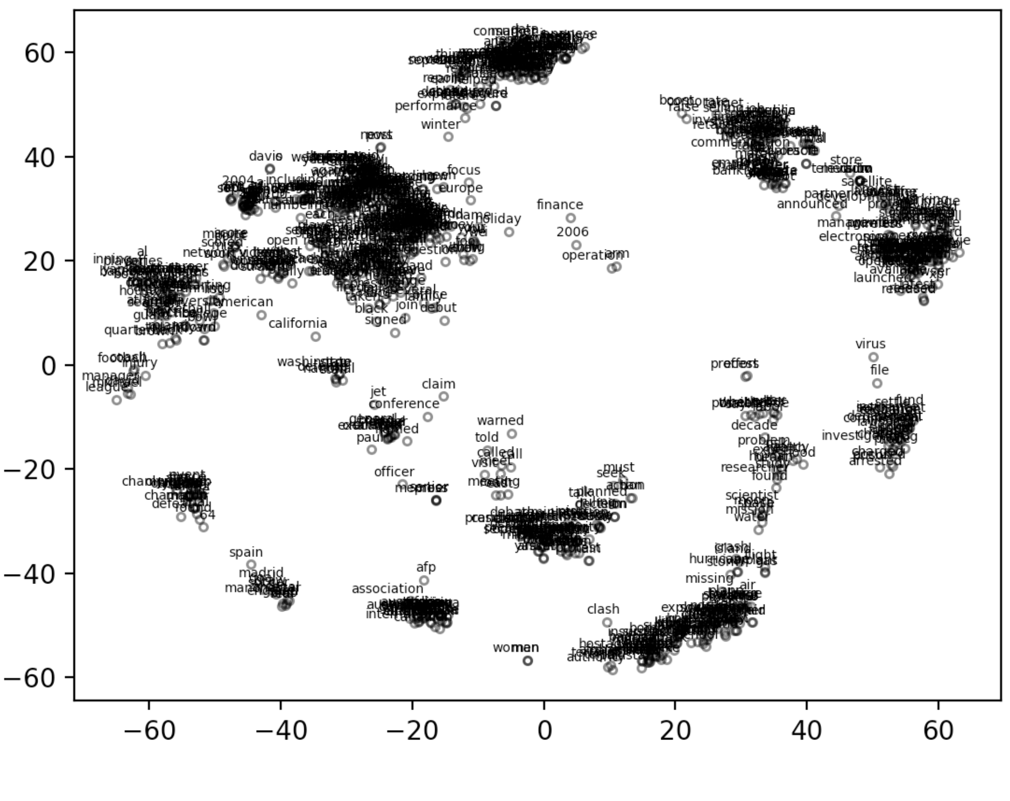

Figure 3: The visualized word vectors

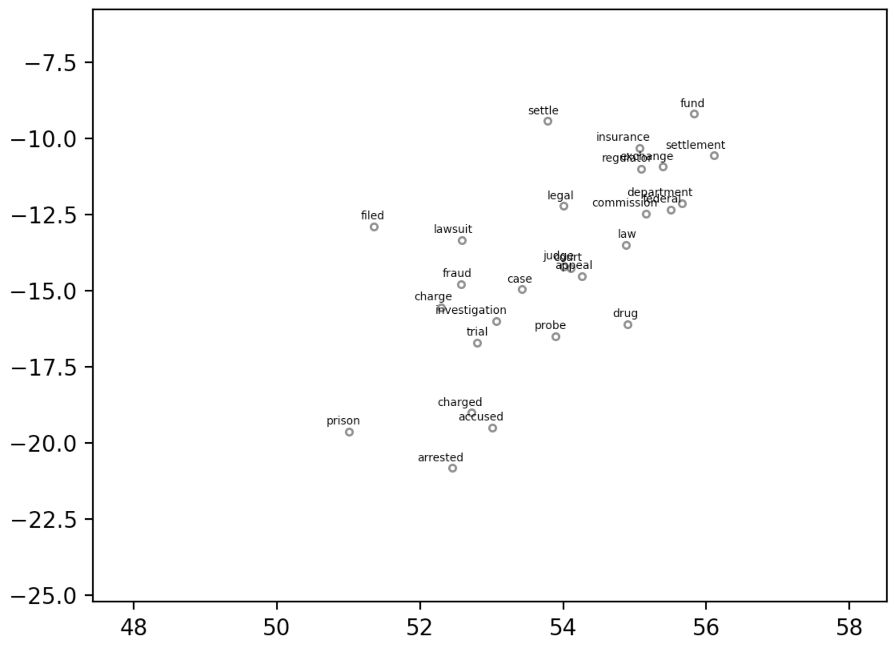

Figure 4: Law category

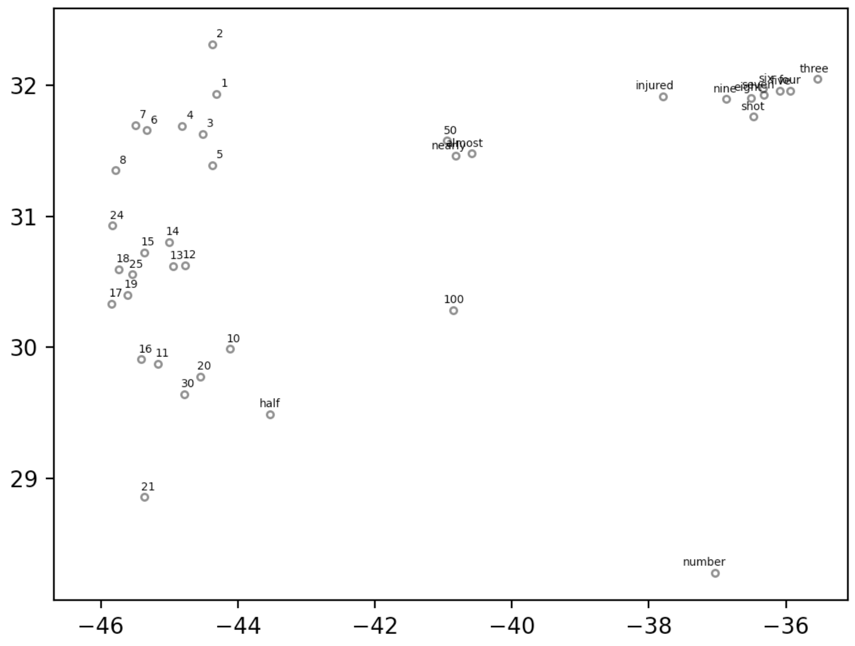

Figure 5: Number category

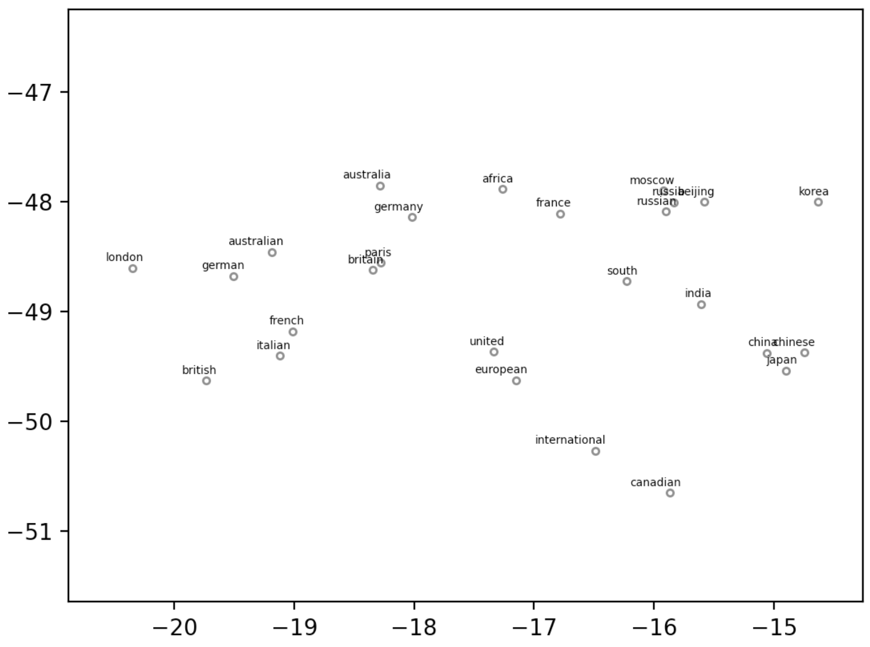

Figure 6: Country and city category

3 Learning-Based Word Vectors – GloVe

3.1

$$J_B = \sum_{(i_m,j_m)\in B} f(C_{i_m j_m})(w_{i_m}^T \tilde{w}_{j_m} + b_{i_m} + \tilde{b}_{j_m} - logC_{i_m j_m})^2$$

$$\nabla_{w_i} J = \sum_{j,(i,j)\in B} f(C_{ij})\tilde{w}_j(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - logC_{ij})$$

$$\nabla_{\tilde{w}_j} J = \sum_{i,(i,j)\in B} f(C_{ij})w_i(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - logC_{ij})$$

$$\nabla_{b_i} J = \sum_{j,(i,j)\in B} f(C_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - logC_{ij})$$

$$\nabla_{\tilde{b}_j} J = \sum_{i,(i,j)\in B} f(C_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - logC_{ij})$$

3.2

Please see in the code for this task.

3.3

In this part, I run `build_glove_vectors.py` with the learning rate 0.1 for 35 iterations. There are 4,047,913 non-zero entries in the count matrix. The loss starts from around 1.04 and decreasing quickly at the very first iterations and go down slowly at the later iteration. Eventually, the loss in the very end iterations decrease very slow (keep slower and slower, which almost stable). The loss converges to 0.03 at the iteration 35.

Figure 7 shows the visualization from `build_glove_vectors.py`. The graph looks like the donut shape. When I zoom in the graph, it gives the similar clustering behavior as in `build_freq_vectors.py`. We can see from Figure 8 as it gives the cluster related to numbers.

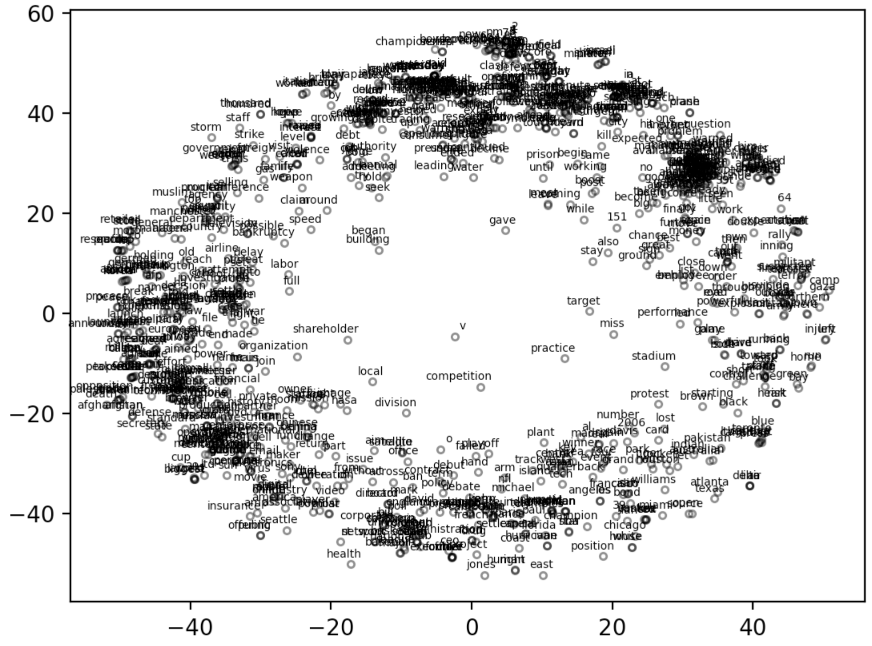

Figure 7: The visualization from build_glove_vectors.py

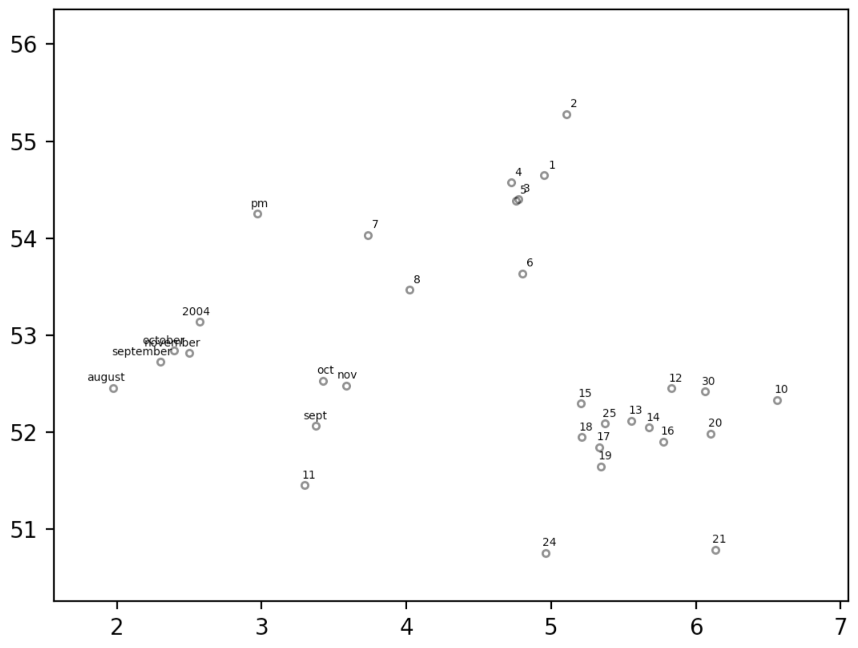

Figure 8: Number category from GloVe

4 Exploring Learned Biases in word2vec Vectors

4.1

```
>>> analogy("man", "dad", "woman")
man : dad :: woman : ?
[('mom', 0.792), ('mother', 0.764), ('daughter', 0.718), ('husband', 0.678), ('
    father', 0.672), ('Dad', 0.668), ('grandmother', 0.664), ('grandma', 0.648), ('
    Mom', 0.642), ('mum', 0.641)]
```

This output satisfy the analogy as a man will always be a dad and a woman will always be a mom. The 2nd result also shows the word mother which is always a woman as well.

```
>>> analogy("man","uncle","woman")
man : uncle :: woman : ?
[('aunt', 0.802), ('mother', 0.777), ('niece', 0.768), ('father', 0.724), ('
    grandmother', 0.722), ('daughter', 0.719), ('sister', 0.701), ('husband', 0.698)
    , ('granddaughter', 0.686), ('nephew', 0.671)]
```

For this output, when people think of uncle no matter what culture it is, we can know immediately that uncle will always be a man. Woman can only be aunt in this case. Therefore, the output satisfy the analogy.

```
>>> analogy("triangle", "3", "square")
triangle : 3 :: square : ?
[('4', 0.662), ('2', 0.66), ('5', 0.649), ('1', 0.638), ('6', 0.638), ('8', 0.63),
    ('7', 0.613), ('9', 0.57), ('##', 0.538), ('###', 0.462)]
```

We all know that a triangle has 3 angles and a square will always have 4 angles, which satisfy the analogy.

4.2

```
>>> analogy("windows", "microsoft", "mac")
windows : microsoft :: mac : ?
[('quickbooks', 0.629), ('buy_microsoft', 0.627), ('adobe_photoshop', 0.611), ('
    microsoft_office', 0.604), ('photoshop', 0.587), ('autocad', 0.578), ('turbotax'
    , 0.575), ('windows_xp', 0.574), ('macintosh', 0.572), ('vmware', 0.567)]
```

We all know that Windows is developed by Microsoft. However, when we say about Mac, we expect to get Apple, the company that develop Mac. We can see in the result list, there is no answer saying the word Apple. This is the reason why this output to not satisfy the analogy.

```
>>> analogy("bird","fly","dog")
bird : fly :: dog : ?
[('dogs', 0.429), ('golden_retriever', 0.421), ('yellow_Labrador_retriever', 0.42),
     ('flying', 0.409), ('pooch', 0.407), ('barked_incessantly', 0.395), ('flown',
    0.395), ('flew', 0.394), ('puppy', 0.391), ('border_collie', 0.387)]
```

In this case, when we say that a bird normally move by flying, so we expect a dog should be move by walking. However, no elements in the list show the word related to walk/walking. The result first 3 results are dog itself and the types of dogs. The 4^{th} one is even worse as the word is flying but we all know that the dog cannot fly. Hence, this doesn't satisfy the analogy.

```
>>> analogy("nose", "smell", "eye")
nose : smell :: eye : ?
[('smelling', 0.509), ('smells', 0.49), ('aroma', 0.482), ('odor', 0.475), ('stench
    ', 0.432), ('pungent_smell', 0.43), ('overpowering_smell', 0.428), ('scent',
    0.425), ('unpleasant_odor', 0.422), ('odors', 0.419)]
```

This case, we use nose to smell things, and we use eyes to see or look at things. However, this result list only return the word related to smelling. We cannot see any answer about see or look which is our expected answers. This does not satisfy the analogy.

4.3

```
>>> analogy("woman", "homemaker", "man")
woman : homemaker :: man : ?
[('machinist', 0.527), ('carpenter', 0.524), ('lifelong_resident', 0.521), ('
    tinkerer', 0.497), ('retired_schoolteacher', 0.496), ('housewife', 0.485), ('
    auto_mechanic', 0.474), ('handyman', 0.471), ('computer_programmer', 0.469), ('
    mechanical_engineer', 0.463)]

>>> analogy("man", "homemaker", "woman")
man : homemaker :: woman : ?
[('housewife', 0.674), ('registered_nurse', 0.576), ('businesswoman', 0.569), ('
    beautician', 0.545), ('seamstress', 0.535), ('Homemaker', 0.528), ('
    medical_transcriptionist', 0.525), ('Ayesha_Dar', 0.525), ('mother', 0.523), ('
    she', 0.522)]
```

This seems to be a bias towards gender. People will think that women will be homemaker, while men will take a stronger job such as machinist and carpenter. Actually, women can also be machinist and carpenter, especially nowadays. When we map a man with homemaker, a woman still be housewife and nurse. People tend to think that men will always take the stronger job.

```
>>> analogy("men", "pilot", "woman")
men : pilot :: woman : ?
[('flight_attendant', 0.488), ('Roark_Schwanenberg', 0.477), ('Cessna_###R', 0.473)
    , ('Hang_glider', 0.469), ('Pilot', 0.467), ('relaxed_el_Amruni', 0.466), ('
    crop_dusting_helicopter', 0.449), ('piloting', 0.444), ('Buddy_Summerfield',
    0.437), ('Rama_Tello', 0.436)]
```

```
>>> analogy("woman", "pilot", "men")
woman : pilot :: men : ?
[('pilots', 0.558), ('piloting', 0.459), ('Pilot', 0.445), ('Brett_Hotchin', 0.436)
    , ('relaxed_el_Amruni', 0.426), ('unpiloted_aircraft', 0.423), ('copilots',
    0.422), ('DGCA_Pradeep_Tyagi', 0.414), ('nonpilot', 0.411), ('aerobatic_pilots',
     0.401)]
```

For this case, people always think that a pilot has to be a man, and a woman has to be a flight attendant. This is an example of word2vec that reinforces stereotypes about gender roles in career. Women tend to have a stereotype of supporting, while men tend to have a stereotype of directing, leader, and managing. However, many pilots, now, are women, which shows that this analogy contains a bias. Besides, we can see the bias that, when a man is a pilot, a woman will be a flight attendant. In contrast, when a woman is a pilot, a man will still be a pilot.

4.4

Q: Why might these biases exist in `word2vec`?

A: People use text for communication, so text can carries the culture, norm, social discrimination, and personal attitude of people. When the input data of `word2vec` contain this kind information, the algorithm, then, contains biases, since the input has its own biases.

When biases happen in the corpus that this model is trained on, the cooccurrence between 2 words with another word will have different values and one value seem to be a lot more than another. For example, the cooccurrence of (man, pilot) is higher than (woman, pilot). Also the cooccurrence of (woman, homemaker) is higher than (man, homemaker). The model will bias towards the higher occurrence.

Q: What are some potential consequences that might result if `word2vec` were used in a live system?

A: When the bias model is used in the real application, we sometimes get the incorrect result which might lead to many problems. For example, if we use this in the chatbot with the question ”Did the engineer solve the problem?”, and the program response ”Yes, he did” rather than ”Yes, she did.” This lead to a bad user experience, especially if the user expects ”she” instead of ”he”. One another example would be a bias in the resume screen application. If `word2vec` is used in the resume screen application, there could be one situation that the application bias towards men in the engineer role. This could lead to the unreliable of the software.