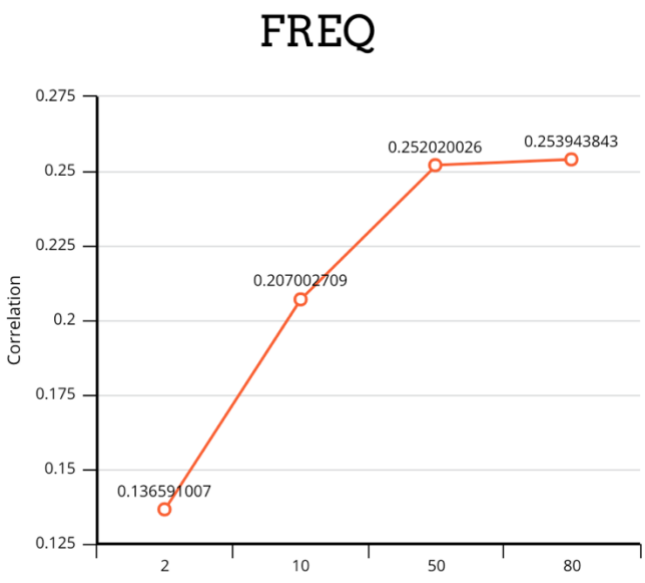
Effect of window size, weighting, and model

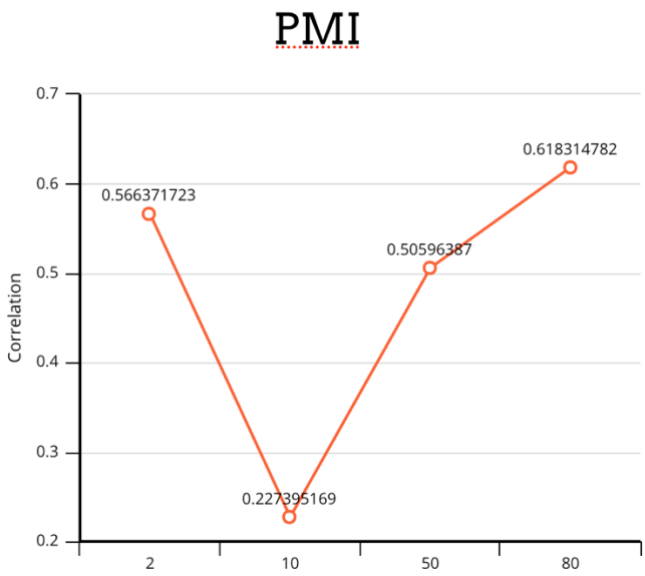
I performed extensive testing on different weighting i.e. FREQ, PMI and different model i.e. word2vec. I calculated correlation at windows_size = 2, 10, 50 and 80.

Increasing the window size resulted in increased correlation value for FREQ which means higher monotonicity of the relationship between two datasets. As per my understanding, increasing the window size allows a larger pool of features to be added to a pool thus increasing the chances of getting higher number of common features between two words.

In the case of PMI, increasing the window size creates a sharp drop in correlation value at windows_size = 10 and then increases gradually. This phenomenon maybe explained by the fact that at windows_size = 10, the PPMI value became zero either due to log(zero) or log(negative).

In word2vec, the drop in correlation can be explained by the fact that increasing the window size causes the dimension of word2vec vectors to increase which throws them farther away from each other.





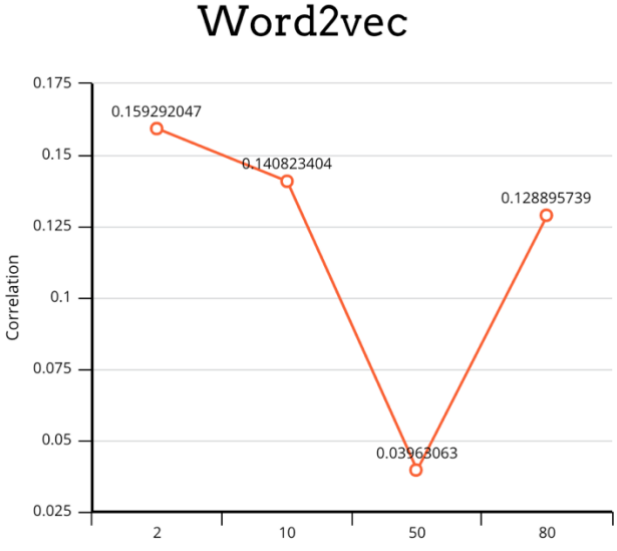
Challenges faced and their solutions

The biggest challenge that I faced was the slow computation in case weighting = PMI.
This was happening because of the repeated computations of *total_sum*, *pw* and *pf* were happening every time *calculate_pmi()* was being called.

```
def calculate_ppmi(matrix, w, f):
    total_sum = matrix.total_sum
    pw = matrix.get_row_sum(w) / total_sum
    pf = matrix.get_col_sum(f) / total_sum
    pwf = matrix[matrix.word_id(w)][matrix.word_id(f)] / total_sum

    return 0 if ((pwf == 0) | (pwf < -1)) else log(pwf / (pw * pf))
```

I solved this problem by storing *pw* and *pf* in a dictionary and looking up in the dictionary first before calculating them again. I also calculated *total_sum* only once in my code and passed it as an argument.



Also, please note that I used the below format in my output file –

4. For each word pair in the file:
    - For each word in the word pair:
        - Print the word and its ten (10) highest weighted features (words) and their weights, in the form:
            - `word feature1:weight1 feature2:weight2 ....`

Source: https://canvas.uw.edu/courses/1222811/assignments/4405673