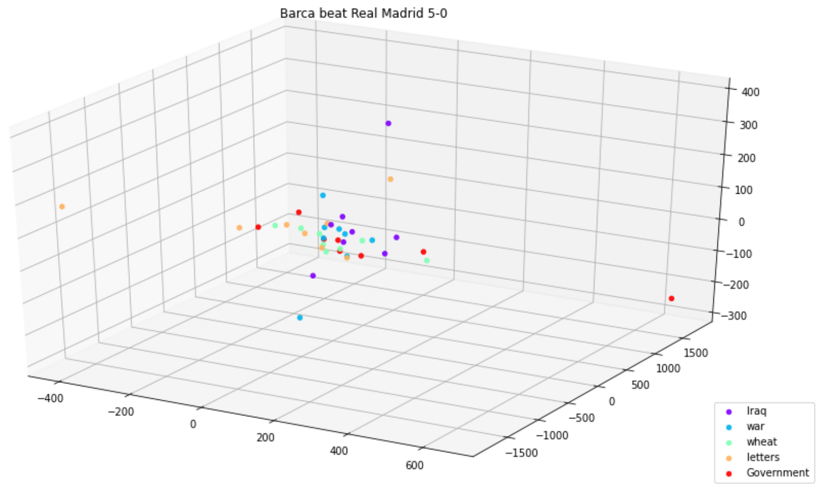
Ques1:

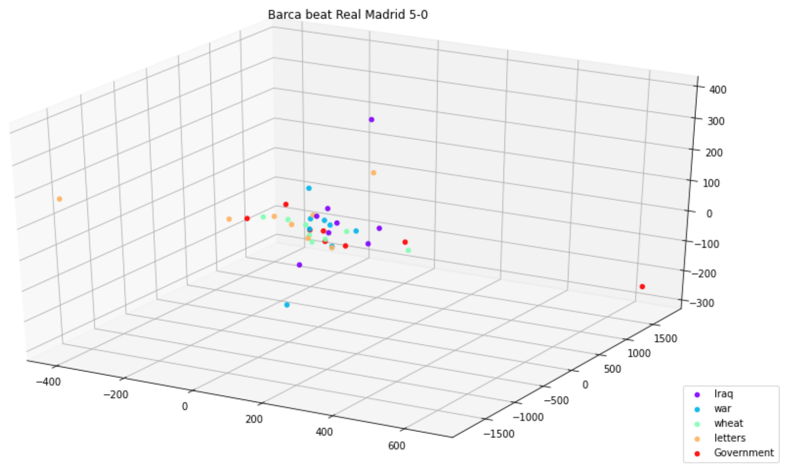
I have implemented a skip gram model to generate word embeddings for a vocab. I have used a negative sampling approach as the softmax(without any optimization) approach takes a LOT of time to train.

My model accepts 2 words as input: one is the selected word(S) and the other is the target word(T). Both are selected at random(almost) from the corpus. If S is within window_size of T, then T is a context word of S(given a score 1). Otherwise T is a negative sample of S(given a score 0).

These words pass through an embedding layer, wherein they get converted to a vector of size vector_length. Dot product of vectors of S and T is then computed. This value is then softmaxed and compared to the truth value(0 or 1). The error is then propagated back into the model for model correction. A high number for epochs has been chosen(2000000) to ensure that maximum number of word combinations are input to the model.



Epoch 1



Epoch 900,000

As can be seen from the figures, cluster formation with increase in epochs can not be seen. This could be due to the fact that as Tsne drops down features to 3 dimensions, a lot of information is lost.

keys = ['Iraq' ,'war', "wheat", "letters", "Government" ]

In an ideal situation, where TSNE was able to capture the essence of the vector, even after scaling down the features, cluster formation would have been observed. This is because words that are related to the same word would have similar vectors, and thus would appear together, forming clusters.
A better method to analyse the workings of the model is the nearest neighbour approach.

At epoch 1 following nearest words are generated for the following keywords:
**Iraq**: **rejects**,scaffold,scattered,isolate,shifted,vibrations,understanding
**war**:Mark, grades, 116, **Americans,** Skies, Therefore, spirals
**Wheat**: **muddy**, merinos, **powder,** powerful, minutes, Link, Defence
**Letters**: **log**, landmark, stadiums, savings, inflows, crustaceans, tightly
**Government**: service ,Apple, Items, happy, conceded, SMART, reactor

At epoch 1900000
**Iraq**: kickbacks ,**Exporter ,Announce ,Veto ,Interests ,shareholders** ,address
**War**: **peace** ,Beach ,**Catastrophe** ,institutions ,**upgrades** ,**subjective** ,**hosts**
**Letters**: **dot** ,handers ,**Posts** ,**Wife** ,Enforcement ,**recovered** ,Flood
**Wheat**: **export** ,**Tonnes** ,**Grain** ,AWB ,**Wheat** ,Cent ,**growers**
**Government**: **Federal** ,Industry ,Farmers ,Drought ,**State** ,**Business** ,**imports**

Clearly it can be seen that as epochs increase, words that are related to each other can be found more in the nearest neighbour approach. This shows that these word vectors that our model outputs are able to capture some essence of the context of the word.

Ques2:

| | 1 Iteration | 3 Iterations |
|---|---|---|
| Relevance Feedback | 0.798 | 0.797 |
| Relevance Feedback(query extension) | 0.799 | 0.798 |

As expected, the solution with query extension performs better, which is due to the fact that we perform extra computation to streamline our query in it.
There isn't any difference in the MAP value as iterations are increased. This could be due to the fact that the model is able to streamline the query in the first iteration itself and thus does not need more iterations.

I expected the Query extension to model to outperform the other model by a higher margin. It doesn't in this case and this could be due to the fact that the model is able to align the query correctly without needing any extra computation as done in the query extension.