IRE Assignment 4

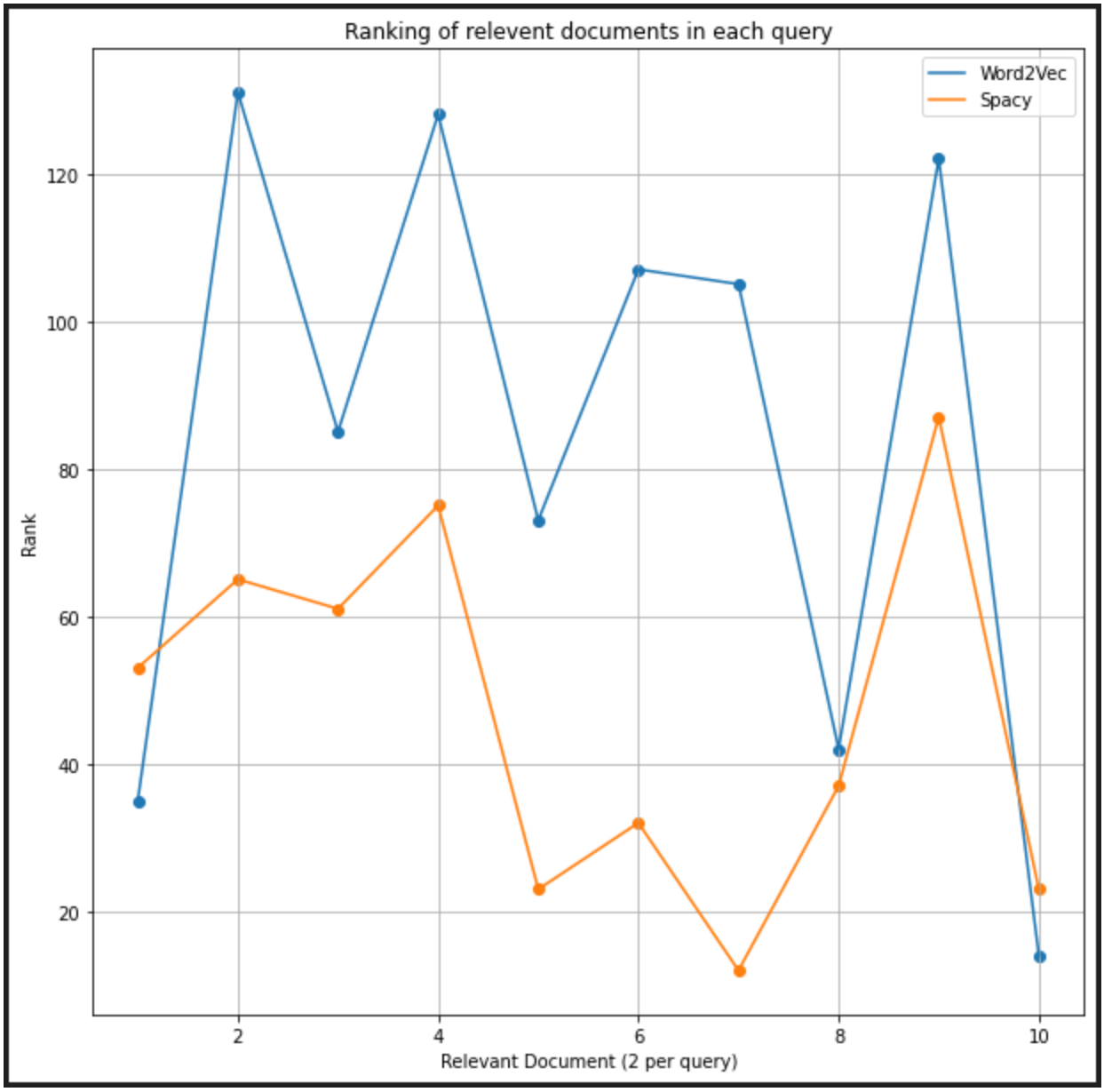
In this assignment, I have installed the following packages: gensim, transformers, torch, tqdm, huggingface, ipywidgets, and spacy modules like en_core_web_sm.

Task 2.1

I am using gensim to train the **word2vec model** after preprocessing the text. I am then using the model to get the query and document vectors and then generating the ranking list based on cosine similarity (from sklearn). Here, my queries are generated by combining the keywords from random .key files in the nasa corpus and the query vectors are obtained by mean pooling the word embeddings within the query. I then convert the results of running 5 queries on the Word2Vec model into a dictionary called “res1”, for further analysis.

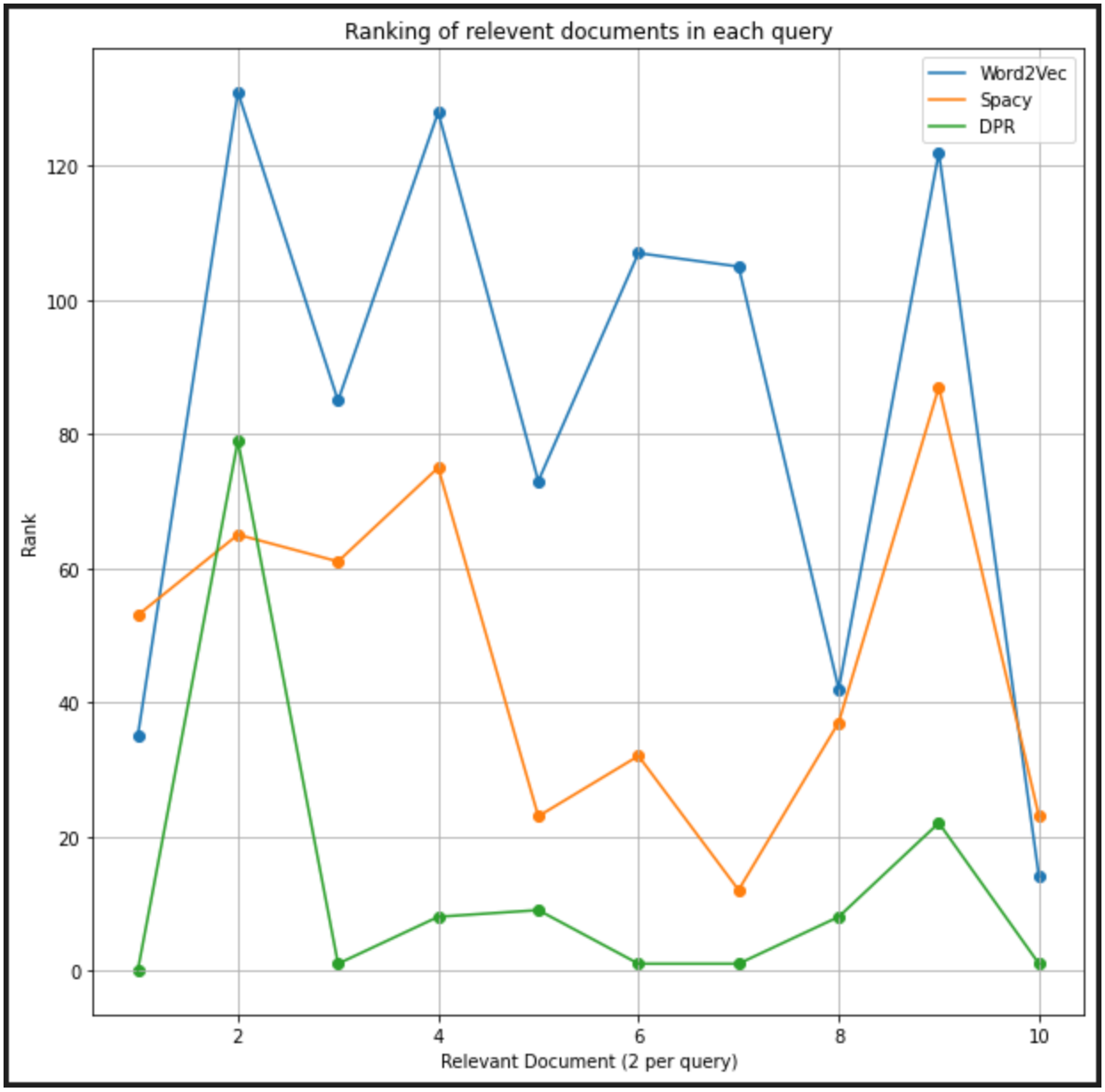
Next, I load the en_core_web_sm model using spacy, and vectorize the same queries and documents using a different vectorizing function. I store the results into another dictionary called “res2”.

I also compare the 2 results by plotting the ranks of relevant documents for each relevant document within each query. In my code, I have considered 2 relevant documents for each query. By observing the plot, we can see that on average, the spacy model appears to be performing better than Word2Vec on these queries. This is because the relevant documents are ranked higher (lower rank number) in the ranking list of the spacy model as compared to that of the Word2Vec model.



Task 2.2

For **dense passage retrieval**, I am using the transformers library to encode the documents and queries to get a fixed length vector for each document and for each query. The model is obtained via huggingface. After this, the similarity scores are calculated (based on cosine similarity) and sorted to obtain the ranked list for DPR. The ranks of relevant documents are stored in a dictionary called “res3”, which is added to the comparison dictionary made in Task 2.1, so that a 3 - way comparison can be performed between the models. The results are plotted as shown below.



From the plot, we can see that DPR performs the best, overall, for the given set of queries.

An observation to note from the results, is that even though Word2Vec is a powerful model, since it is getting trained on the nasa corpus it might not be performing as well as DPR. Computationally, DPR takes the most time, while Word2Vec runs relatively quickly. Also, in Word2Vec, we are performing mean-pooling of word embeddings to get the query embedding, while in DPR, we directly generate 1 vector for the query.