First Step: preprocessing the data:

After preprocessing I save the preprocessed documents in the corresponding new documents called preprocessed. for preprocessing I delete punctuations, covert each letter to its lower case and delete stop words. (Just copied from the previous home works)

I also created the inverted index and posting list for calculating tf and idf later. (Just copied from the previous home works).

I also created a list for storing all the queries in for easier accessing.

This is the corpus that contains all the terms. In this corpus similar terms could be repeated because we use this corpus to count frequency of a term in all documents i.e., D_c

Step 2: Implementing Okapi BM25:

in this step I have implemented the okapi NM25 as asked in the question. to do so, I used a helper function which does the computation part of the BM25 method and in the main function I get all the documents once and calculate the score of each document with the query based on the BM25 with basic weighting.

the reason we used default values for k1 and b i.e., k1 = 1.2 and b = 0.75 is that is realistic because users tend to write short queries and also documents have variable length and with these values, we can handle short and long documents together.

Step 3: Implementing the Language Model:

Here I have implemented the language model using Jelinek Mercer smoothing method with the default value lambda = 0.5. Like BM25 it uses a helper function for the main computation and in the main function we calculate this score for each document and return the k nearest documents to the query. To avoid runtime error in the calculation I used an if condition to skip terms with zero occurrence in the corpus because calculating the logarithm there could run into an error.

I used the Jelinek-Mercer smoothing method because it can handle the zeros efficiently and it's easy to implement and use. it is also more robust for query with different lengths.

Step 4: Implementing the hybrid method

Firstly, I implemented the vector space model using the previous homework. Then in the hybriModel function I have defined some weights called w1, w2 and w3 to normalize impact of each model in the final score. for getting the correct answers I have divided each document score to the maximum of the that model and also I have subtracted that value from the minimum.

Here's more explanation of why I have used this:
in the vector space model scores are integers and can be high up to even more than 100. In the BN25 model scores are floats and small and finally in the language model scores are negative and have high variance. all together for normalizing i used that formula to get number between 0 and 1 and also tried to make huge difference in the scores between documents. but still in the language model values are close to each other so I gave it more weight for more considering even small differences between scores.
Since the dataset is not judged at all we can't know which documents are relevant and which ones are not relevant, so recall means nothing in this situation, furthermore we cannot compute 11-point interpolation since we cannot compute recall for this dataset. in conclusion without further judgment, we cannot compute any metric for our models.

But here is my own comparison with checking documents with one-view judgment by myself:
most of the results are the same specially about the first and second highest score retrieved documents, but in third and fourth one I think the hybrid method is doing better and return more relevant documents. comparing the result with the both methods in HW2 result are almost similar and but I think in the all differences in this home works results are better in quick looking on documents.

Finally, we can't make a precise comparison between all the models but if we can say approximately the hybrid model is doing better than other models.