Given corpus:

Document1 : today is sunny

Document2 : she is a sunny girl

Document3 : to be or not to be

Document4 : she is in berlin today

Document5 : sunny berlin sunny

Document6 : berlin is always exciting

Part a. 1) :

For Document1 and Document 1 :

Euclidean distance = 1.2908633236550593

Dot product = 0.12162718980527158

Cosine similarity = 0.16475679254915546

2) Given query: 'to sunny girl'. Similarity score:

For Document 1 : 0.294081248262307

For Document 2 : 0.49880205059154453

For Document 3 : 0.3651483716701107

For Document 4 : 0.0

For Document 5 : 0.5163977794943222

For Document 6 : 0.0

Ranking:

Rank 1 = berlin is always exciting

Rank 2 = sunny berlin sunny

Rank 3 = she is in berlin today

Rank 4 = to be or not to be

Rank 5 = today is sunny

Rank 6 = today is sunny

Part b:

Relevant documents while querying the document2 'She is a sunny girl' using vector space model – Ranking below:

0.8988298313517133 : berlin is always exciting

0.3999999999999999 : sunny berlin sunny

0.36104551116967 : she is in berlin today

0.26864618819961844 : to be or not to be

0.06821692667025168 : she is a sunny girl

Relevant documents while querying document1 'She is a sunny girl' using the BM25 model – Ranking below :

4.9349017 : she is a sunny girl

1.3843269 : she is in berlin today

1.2984171 : today is sunny

1.0433689 : sunny berlin sunny

0.45618832 : berlin is always exciting

Approach:

Part a):

The three similarities were computed with the methods getCosineSimilarity, getDotSimilarity, getEuclideanSimilarity. The addDoc method allows to store the documents in a writer and keeps count of the number of documents present in the corpus. Then, the addTerms adds the terms present in the document into the hashset. The function getTermFrequencies returns the vector representation of the document by taking the reader and docId as input parameters.

Part b):

For BM25 model, an index searcher from lucene was employed by setting its similarity to BM25 similarity and the query was created by parsing the string. Further, the lucene's TopScoreDocCollector to compute the score for every document and store it. Next, the scores are compared and the top documents are printed as output.