IR Assignment 2

Question 1:

1: Pre-processing

Both the data files and query are pre-processed using the following methods in order to be ready for models.

1. Removal of accented characters(if any)
2. Expanding contractions
3. Removal of special characters
4. Converting data to lower case
5. Tokenization of data
6. Number converted to English language words
7. Removal of stop words
8. Removal of single characters
9. Lemmatization
10. Stemming

2: Generating various Data Structure for usage

1. Master Dictionary for corpus

   Format : {docName:[tokens list]}

2. Master Dictionary for titles

   Format : {titleName:[tokens list]

3. DocName-DocID Mapping and same format for title also

   Format : {docID:DocName}

4. Dictionary with docID

   Format : {docID:[tokens list]}

5. Inverted Index

   Format : {term:{docID:term frequency}}

3. Assumptions

1. Number to words will convert only pure numeric terms eg. ‘1234’ is converted to ‘One thousand Two Hundred Thirty Four’.
2. Decimal values are not retained and only their ceiling value is taken.
3. Stemming after lemmatization is done to improve the results as lemmatization was not able to give root word of each of the vocab words and also stemming has saved space for some of the words by cutting the suffix’s.
4. Libraries used are :

```
import os
import email
import pickle
import math

import nltk
import re, unicodedata
import contractions
import inflect

from nltk import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer,PorterStemmer
```

Methodology

After the above steps are performed successfully, the scoring based on 3 different measures was done and the K documents are retrieved based on the query and K value given the user with some other details.

1. Jaccard Score
   First the count of intersecting terms of query and document is calculated and the Jaccard score. For each document, score is saved in a dictionary and then dictionary is sorted in non-increasing order to retrieve K documents.
2. TF-IDF Matching Score
   For each document, query terms TF-IDF is calculated and them summed up to get document TF-IDF score. For each document, score is saved in a dictionary and then dictionary is sorted in non-increasing order to retrieve K documents.

3. Cosine Similarity
   In this, query and document TF-IDF vectors are created first. Then cosine similarity of each document with respect to query is calculated using the well-known formula. For each document, score is saved in a dictionary and then dictionary is sorted in non-increasing order to retrieve K documents.

Question 2:

Steps followed are:

1. Loaded the English dictionary
2. Taken query from the user.
3. Make a list of those query words which are not present in the dictionary.
4. For each query word in above list:
   1. For each dictionary word:
      a. Calculate edit distance with specified cost of operations using dynamic programming and memorizations
      b. Store the result in a dictionary
   2. Sort the dictionary based on edit distance in increasing order
   3. Display top 10 word suggestions.