IR Assignment : 02

Submitted By :

Group No.-3

Atul	atul17032@iiitd.ac.in
Prashant Jain	prashant18253@iiitd.ac.in
Akanksha Pandey	akanksha20048@iiitd.ac.in
Shivani Mishra	shivani20062@iiitd.ac.in

Question 01:

Methodology :

1. **Read Files:**
   To Read files present in stories folder
   *files_list = glob.glob("drive/My Drive/stories/" + '**/*', recursive=True)*
   Open a particular file
   rfile = open(file_path,'r',errors = 'ignore')
   Read content of that file
   read_file = rfile.read()

2. **Perform following Pre processing Steps :**
   a. Convert the text to lowercase.
   b. Perform word tokenization.
   c. Remove stop words
   d. Remove punctuation marks
   e. Remove extra blank spaces.

3. **To create Positional index:**
   1. The dictionary file_words is used to store the file words mapping
   2. The dictionary pos_index is of type *{token: {doc:[positions]} }* is a multidimensional dictionary that stores a dictionary for

each token which contains the documents and positions for the tokens.

3. Pickle is used to save the positional index and the file mapping in the disk

4. Query Preprocessing:

a. Read query and operation sequence.
b. Apply the same preprocessing on the input query that was applied on files.
c. For each query token we compare its positions with other tokens in the documents. There is a match if the difference between the positions is exactly equal to the relative positions in the query tokens.

.

Question 02:

a. **Jaccard:**

Methodology :

1. **Preprocessing:**
    1. We have done the same **preprocessing** steps as in Question 1.

2. **To find Jaccard coefficient:**
    a. Create tokens for documents and query after preprocessing
    b. Convert them to a set as we don't need duplicate words of a file.
    c. Apply the formula : Jaccard Coefficient = Intersection of (doc,query) / Union of (doc,query)
    d. Sort the values in descending order.
    e. Print files that have top 5 Jaccard coefficients.

3. **Query Processing:**
    We have done the same **preprocessing** steps as in Question 1.

b. **Tfidf:**

Methodology :

1. **Preprocessing:**
    We have done the same **preprocessing** steps as in Question 1.

2. **Finding idf values:**
    a. We have used Smoothing to calculate idf value.
    b. We have used the idf formula for each word of each document.
    c. Formula : IDF(word)=log(total no. of documents/document frequency(word)+1)

3. **Finding tf values :**
    a. **Binary**

        In the binary model, we simply use 0,1 values. 0 denotes absence and 1 denotes presence of a word in a particular document.

    b. **Raw Count**

        In the raw count model, the value of the matrix[doc][token] is equal to the frequency of the token in doc.

    c. **Term Frequency**

        In the term frequency model, the value of the matrix[doc][token] is equal to the frequency of the token in doc divided by the total number of words present in that document

    d. **Log Normalization**

        In the log normalisation model, the value of the matrix[doc][token] is equal to the log(1+f(token,doc)), where f(token,doc) is the frequency of the term in doc

    e. **Double Normalization**

        In the double normalisation model, the value of the matrix[doc][token] is equal to the $0.5 + 0.5 * (f(token, doc) / max(f(t`, doc))$, where f(token,doc) is the frequency of the term in doc and $max(f(t`, doc))$represents the maximum frequency of term t in that doc.

4. **Finding tf-idf values :**

    In the above model, we multiply these with the IDF_scores of each term to get the TF-IDF matrix for each variant, that is, $TFIDF(term,\ doc) = tf(term,\ doc) * IDF(term)$
    So, there will be total 5 TF_IDF matrix

c. Cosine

Methodology :

1. **Preprocessing:**

   We have done the same **preprocessing** steps as in Question 1.

2. **Query Processing**

   A normalised query vector is formed first with the length of vocabulary

   This query vector is regarded as a document and is compared with all other documents using the tf-idf matrices formed in the above part. Each tf-idf value is first normalized and the cosine score is computed.

   $$score = \Sigma \frac{Ai.Bi}{|Ai||Bi|}$$

   The scores obtained are then sorted in reverse order and top k documents can be retrieved.

Question 03 :

Methodology :

1. **Read File :**

   data = pd.read_csv("IR-assignment-2-data.txt", sep=' ', header=None)
   data.to_csv('IR_A2_Q3.csv', index = None)

2. **Find Max DCG :**
   a. Select rows that have quid:4
   b. Sort the data frame on the basis of relevance judgement label.
   c. Find the DCG value for each k (range 1 - 104)
   d. To find DCG, use the formula :

$$DCG_p = \sum_{i=1}^{p} \frac{rel_i}{log_2(i+1)}$$

   e. Select Maximum DCG value

3. **Possible number of files that can be made :**

4. **To find nDCG :**
   a. To find the nDCG value, use the formula :

$$nDCG_p = \frac{DCG_p}{IDCG_p}$$

   Where IDCG is ideal DCG.
   b. Use the same formula for finding the value at 50 and for the whole dataset.

5. **Plot Precision Recall Curve :**
   a. Take the 1st and 75th column in a list.

b. Remove substring “75:” from the strings that are present in the selected column.
c. Convert the numbers present in the 75th column to float from string.
d. Sort relevance judgement label on the basis of 75th column in descending order.
e. Use Formula:
   I. Precision = Retrieved relevant documents / Total number of documents
   II. Recall = Retrieved relevant documents / Total number of relevant documents
f. Use matplotlib to plot the graph.

Assumptions :

1. We have removed all the unnecessary files while reading. Total File count after reading was : 467.
2. Queries will be space separated
3. We have done preprocessing for finding Jaccard Coefficient.