fCSE508 Information Retrieval
Winter 2023
Assignment-2

Due Date: March 23, 2023, 23:59 **Max. Marks:** 100

Instructions:

1. The assignment is to be attempted in groups of max 3 members.
2. Each group member must do at least one task. All members should know the working of all the tasks. This will be evaluated during your code demo and viva.
3. Institute plagiarism policy will be strictly followed.
4. Programming language allowed: Python.
5. Your code should be well documented.
6. You are free to use libraries like NLTK, BeautifulSoup for data preprocessing.
7. You are required to use version control via GitHub:
    a. Make a GitHub repository with the name: **CSE508_Winter2023_A2_<Group_No.>**.
    b. Add your assignment TA as a contributor. The TA assigned (along with their GitHub handle) to your assignment group for this assignment can be found here. c. Contribution of each member will be monitored via git commits.
8. You must make a detailed report with the name **Report.pdf** covering your methodologies, assumptions, and results.
9. Submission:
    a. A zipped folder **CSE508_Winter2023_A2_<Group_No.>** consisting of all your code files, dumped files and **Report.pdf**
    b. A text file **CSE508_Winter2023_A2_<Group_No.>.txt** consisting of the link to your GitHub repository.
10. Only one member from a group needs to submit.
11. You cannot use any exact API/library for the tasks you have been assigned. You have to do the implementation from scratch. For instance, if you have been asked to implement IDF then you can't use API for the same.

Q1. Data Preprocessing, Scoring and Term-Weighting[40 marks]

Dataset Link: Dataset [1400 files]

NOTE: Preprocessing is a must in this question, and is same as did in the last assignment-

(i) Relevant Text Extraction - For each file, extract the contents between the <TITLE>...</TITLE> and <TEXT>...</TEXT> tags and concatenate the 2 strings using blank space. Discard the rest of the text and save the string obtained above in the same file. [Do NOT change filename].
Example:-
Final contents of the file **carnfield003**: *the boundary layer in simple shear flow past a flat plate . the boundary-layer equations are presented for steady incompressible flow with no pressure gradient .*

Perform this on all 1400 files. Print contents of 5 sample files before and after performing the operation.

(ii) Preprocessing
Carry out the following preprocessing steps on the dataset obtained above:

1. Lowercase the text
2. Perform tokenization
3. Remove stopwords
4. Remove punctuations
5. Remove blank space tokens

Print contents of 5 sample files before and after performing EACH operation.

TF-IDF Matrix [25 points]

The aim is to create a TF-IDF matrix for every term in the vocabulary and determine the TF-IDF score for a given query. The TF-IDF consists of two components - Term Frequency and Inverse Document Frequency.

• To compute the Term Frequency, the frequency count of each term in every document is computed and stored as a nested dictionary for each document.
• To determine the document frequency of each term, the postings list of each term is found and the number of documents in each posting list is counted.
• The IDF value of each term is calculated using the following formula with smoothing: IDF(term) = log(total number of documents/document frequency(term)+1).
• The Term Frequency is calculated using 5 different weighting schemes, as listed below:

Weighting Scheme	*TF Weight*
Binary	*0,1*
Raw count	*f(t,d)*
Term frequency	*f(t,d)/Pf(t', d)*
Log normalization	*log(1+f(t,d))*
Double normalization	*0.5+0.5*(f(t,d)/ max(f(t',d))*

The following steps should be followed:

1. Utilize the same data as in assignment 1 and perform the same preprocessing steps as previously stated.
2. Create a matrix of size no. of documents x vocab size.
3. Fill in the tf-idf values for each term in the vocabulary in the matrix.
4. Construct the query vector of size vocab.
5. Compute the TF-IDF score for the query using the TF-IDF matrix. Report the top 5 relevant documents based on the score.
6. Use all 5 weighting schemes for term frequency calculation and report the TF-IDF score and results for each scheme separately.

It is essential to state the pros and cons of using each weighting scheme to determine the relevance of documents in the report.

Jaccard Coefficient [15 marks]

The objective is to determine the Jaccard coefficient between a specified query and document, where the formula is provided as follows:

Jaccard Coefficient = Intersection of (doc,query) / Union of (doc,query).

A higher Jaccard coefficient value indicates greater relevance of the document to the query.
To accomplish this, after preprocessing on the given dataset, create sets of the document and query tokens, and compute the intersection and union for each document and the query.
Finally, present the top 10 documents ranked by Jaccard coefficient value.

Q2.

Naive Bayes Classifier with TF-ICF[40 marks]-

Dataset:
The dataset can be downloaded from Kaggle
https://www.kaggle.com/competitions/learn-ai-bbc/data

In this assignment, you will implement a Naive Bayes classifier with TF-ICF (term frequency-inverse category frequency) weighting scheme to classify documents into predefined categories based on their content. You will use a dataset of news articles and classify them into categories such assport, tech, business, etc.

TF-ICF score for a given term belonging to a class can be calculated as follows:
Term Frequency (TF): Number of occurrences of a term in all documents of a particular class
Class Frequency (CF): Number of classes in which that term occurs
Inverse-Class Frequency (ICF): log(N / CF), where N represents the number of classes (log 10)

You will perform the following tasks:

1. Preprocessing the dataset:
    - The dataset is in CSV format with the following columns: 'ArticleId', 'Text', and 'Category'.
    - Remove any unnecessary columns.
    - Clean the text by removing punctuation, stop words, and converting all text to lowercase.
    - Tokenize the text by splitting it into words.
    - Perform stemming or lemmatization to reduce words to their root form.
    - Implement the TF-ICF weighting scheme.
2. the dataset:
    - Split the BBC train dataset into training and testing sets.
    - Use a 70:30 split for the training and testing sets, respectively.
3. Training the Naive Bayes classifier with TF-ICF:
    - Implement the Naive Bayes classifier with the TF-ICF weighting scheme.
    - Calculate the probability of each category based on the frequency of documents in the training set that belong to that category.
    - Calculate the probability of each feature given each category based on the TF-ICF values of that feature in documents belonging to that category.
4. Testing the Naive Bayes classifier with TF-ICF:
    - Use the testing set to evaluate the performance of the classifier.

- Classify each document in the testing set and compare the predicted category with the actual category.
- Calculate the accuracy, precision, recall, and F1 score of the classifier.

5. Improving the classifier:
    - Experiment with different preprocessing techniques and parameters to improve the performance of the classifier(including different splits like 60-40,80-20, 50-50).
    - Try using different types of features such as n-grams or TF-IDF weights.
    - Evaluate the performance of the classifier after each experiment and compare it with the previous results.
6. Conclusion:
    - Write a brief report summarizing your findings.
    - Discuss the performance of the classifier and the impact of different preprocessing techniques, features, and weighting schemes on the results.

Deliverables:

- Python code implementing the Naive Bayes classifier with TF-ICF weighting scheme.
- A report summarizing your findings and conclusions.
- A README file explaining how to run your code.

Grading Rubric:

- Preprocessing (2 points)
- Splitting the dataset (5 points)
- Training the Naive Bayes classifier with TF-ICF (10 points)
- Testing the Naive Bayes classifier with TF-ICF (10 points)
- Improving the classifier (10 points)
- Conclusion (3 points)

Note:

- Please make sure to cite any sources used in your code or report properly.
- Your code should be well-documented and easily readable.
- Your report should be well-written and well-structured.

Q3) [20 points] Ranked-Information Retrieval and Evaluation

The given task involves working with the Microsoft Learning to Rank dataset, which can be accessed through the provided link. Dataset- link

The dataset contains various queries and their associated URLs with relevance judgement labels as relevance scores.
To complete this task, focus only on the queries with qid:4 and use the relevance judgement labels as the relevance score.
The first objective is to create a file that rearranges the query-url pairs in order of the maximum DCG (discounted cumulative gain). The number of such files that could be made should also be stated.

Next, compute the nDCG (normalized discounted cumulative gain) for the dataset. This involves calculating nDCG at position 50 and for the entire dataset.

For the third objective, assume a model that ranks URLs based on the value of feature 75, which represents the sum of TF-IDF on the whole document. URLs with higher feature 75 values are considered more relevant. Any non-zero relevance judgement value is considered relevant. Using this model, plot a Precision-Recall curve for the query "qid:4".
The curve should help visualize the trade-off between precision and recall as the model's threshold for relevance is adjusted.