REPORT
IR ASSIGNMENT 2

Aayush Singh 2020009 Hrishav Basu 2020067 Ipsita R 2020379

Question 1.

I. The data provided in assignment 2 is being used and Same Relevant Text extraction is performed as done in assignment one.

II. The same pre-processing methods used in assignment 1 are being applied here. All punctuation marks are initially removed and the text is transformed to lowercase. The remainder of the text is then broken down into individual tokens after the stopwords have been removed. Lastly, the text is cleared of any empty tokens.

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

III. Tf-Idf Matrix Methodology:

a. Preprocessed the given dataset.
b. Returned tokens list of individual files as set after preprocessing.
c. Preprocessed the query, into tokens as set.
d. Calculated the IDF value of each word is calculated using the formula as mentioned below: Using smoothing:- IDF(word)=log(total no. of documents/document frequency(word)+1).
e. Used all 5 weighting schemes for term frequency calculation and report the TF-IDF score and results for each scheme separately.
f. Computed tf-idf for each doc-val pair and stored it.
g. Calculated tf-idf score for the query using the matrix.

IV. Jaccard Coefficient Methodology:

a. Preprocessed the given dataset
b. Returned tokens list of individual files as set after preprocessing
c. Preprocessed the query, into tokens as set
d. Used & for intersection of the sets,(filenumber1 & query) and so on.
e. Used | for union of the sets,(filenumber1 | query) and so on.
f. Using formula to calculate the jaccard coefficient, calculated it and stored it in a dictionary, for individual files.

g. Used Counter for retrieving the topmost 5 documents.

Binary-

Pros:

a. Simplicity: The binary weighing scheme is simple to implement and understand. It requires only a single binary value to represent the presence or absence of a term in a document.
b. Robustness: The binary weighing scheme is robust to the length of the document. It gives equal weightage to all terms that are present in a document, regardless of their frequency.
c. Computationally efficient: The binary weighing scheme is computationally efficient and requires less memory compared to other weighing schemes.

Cons:

a. Loss of information: The binary weighing scheme does not take into account the frequency of a term in a document. It treats all terms that appear in a document as equally important, which can result in a loss of valuable information.
b. Lack of discrimination: The binary weighing scheme does not discriminate between terms that appear in different documents. It treats all terms that appear in a document as equally important, which can result in reduced ability to distinguish between different documents.
c. Inability to capture document length: The binary weighing scheme does not consider the length of the document. Thus, it may not be suitable for applications that require consideration of the document length.

Raw Count-

Pros:

a. a.Captures term frequency: The raw count weighing scheme captures the frequency of a term in a document. This allows for more accurate representation of the document's content and can lead to better results in some applications.
b. Simple to implement: The raw count weighing scheme is simple to implement and understand. It only requires counting the number of times a term appears in a document.
c. Applicable to short documents: The raw count weighing scheme is applicable to short documents where the binary weighing scheme would fail to provide enough information

Cons:

a. Susceptible to document length: The raw count weighing scheme is susceptible to the length of the document. Longer documents tend to have

higher term frequencies, which can lead to bias towards longer documents in some applications.

b. Susceptible to high frequency terms: The raw count weighing scheme can give undue importance to high frequency terms, which can lead to inaccurate representation of the document's content. Common words such as "the" or "a" can appear frequently in many documents but do not necessarily represent the content of the document.

c. Not normalized: The raw count weighing scheme is not normalized and does not take into account the overall frequency of a term in the corpus. This can result in a lack of discrimination between terms that are important in one document but not in another.

Term Frequency-

Pros:

d. Reflects importance of terms within documents: By using the term frequency, the weighting scheme is able to reflect the importance of a term within a document. This can help to rank documents based on their relevance to a query.

e. Simple and easy to implement: The TF weighting scheme is relatively simple and easy to implement. It only requires counting the number of times a term appears in a document.

Cons:

a. Does not account for term importance across the collection: While the TF weighting scheme reflects the importance of a term within a document, it does not take into account the importance of the term across the collection as a whole. This can lead to issues when ranking documents.

b. Susceptible to document length bias: The TF weighting scheme is susceptible to document length bias, which means that longer documents will have higher term frequencies, even if the terms are not particularly important to the document's content. This can lead to inaccurate ranking of documents.

Log Normalisation-

Pros:

a. Reduces the effect of document length bias: Log normalisation reduces the effect of document length bias, which means that longer documents will no longer have higher term frequencies, even if the terms are not particularly important to the document's content.

b. Smoothes the distribution of term frequencies: The log function compresses the range of term frequencies, resulting in a smoother distribution of term frequencies. This can help to mitigate the impact of extreme values and outliers.

c. Reflects diminishing returns: Log normalisation reflects the idea of diminishing returns, which means that the impact of additional occurrences of a term decreases as the frequency of the term increases.

Cons:

a. May not work well with very small term frequencies: Log normalisation may not work well with very small term frequencies, as the logarithm of very small numbers approaches negative infinity. This can lead to issues when ranking documents.

b. Can be sensitive to base of logarithm: The choice of base for the logarithm can affect the final scores and ranking of documents. Different bases may result in different weighting schemes, which can impact the effectiveness of the TF-IDF weighting scheme.

c. Does not address stop words: Log normalisation does not address the issue of stop words, which are words that are common in the language but do not carry much meaning. This can lead to inaccurate ranking of documents if stop words are given too much weight.

Double normalization-

Pros:

a. Reduces the impact of high frequency terms: The Double normalization weighing scheme reduces the impact of high frequency terms by normalizing the term frequency with respect to the maximum frequency in the document. This ensures that the weight assigned to each term is proportional to its importance in the document.

b. Provides better discrimination between documents: The Double normalization weighing scheme provides better discrimination between documents by normalizing the inverse document frequency using the logarithm of the total number of documents divided by the number of documents containing the term. This ensures that rare terms are given more weight than common terms, which can lead to better results in some applications.

c. Balanced representation of short and long documents: The Double normalization weighing scheme is balanced in representing short and long documents since it normalizes the term frequency. This makes it

suitable for use in applications that require the representation of short documents, unlike raw count which favors long documents.

Cons:

a. Can lead to zero weights: The Double normalization weighing scheme can lead to the weight of a term being zero if it does not appear in a document. This can result in the loss of valuable information in some application.
b. Computationally expensive: The Double normalization weighing scheme is more computationally expensive compared to the binary or raw count weighing schemes, which can result in slower performance in some applications.
c. Requires tuning of parameters: The Double normalization weighing scheme requires tuning of the normalization parameters, which can be challenging in some applications.

Question 2.

The process for implementing a Naive Bayes classifier for text data involves importing necessary libraries, loading the training and test datasets, converting the text data into TF-IDF vectors using CountVectorizer and TfidfTransformer, training the classifier with the TF-IDF vectors, converting the test data into TF-IDF vectors, using the trained classifier to predict the categories of the test data, and calculating performance metrics using the "weighted" average for precision, recall, and F1 score.

To start, the BBC News Train.csv dataset is loaded using read_csv and unnecessary columns are dropped. Text is cleaned by removing punctuations, converting to lowercase, and removing stop-words. NLTK library is used for tokenization, stemming, and lemmatization. Sklearn library is used for TF-IDF and related work, using CountVectorizer and TfidfTransformer, and these values are stored in a dictionary named tf_icf. The updated BBC train dataset is split into train and test datasets on a 70:30 ratio using the train_test_split function from the sklearn.model_selection library. The resulting train and test datasets are saved in separate files.

To train the Naive Bayes classifier, TfidfVectorizer is used to create a TF-IDF vectorizer, and MultinomialNB is used to train the classifier using the TF-IDF weighted data. The training data is transformed using the vectorizer, and the TF-IDF score for each term in the training data is calculated. The code outputs the initial accuracy and classification report for the classifier.

Different pre-processing techniques and parameters can be experimented with to improve the performance of the classifier.

Experiment 1 involves changing the train-test split to 60-40, 80-20, and 50-50 splits and comparing their performances.
60-40 Split: Accuracy = 0.9531, Precision = 0.9575, Recall = 0.9495, F1 Score = 0.9535
80-20 Split: Accuracy = 0.9624, Precision = 0.9653, Recall = 0.9606, F1 Score = 0.9629
50-50 Split: Accuracy = 0.9539, Precision = 0.9577, Recall = 0.9504, F1 Score = 0.9540

Experiment 2 involves using TF-IDF weights instead of TF-ICF weights.
TF-IDF: Accuracy = 0.9721, Precision = 0.9746, Recall = 0.9706, F1 Score = 0.9726

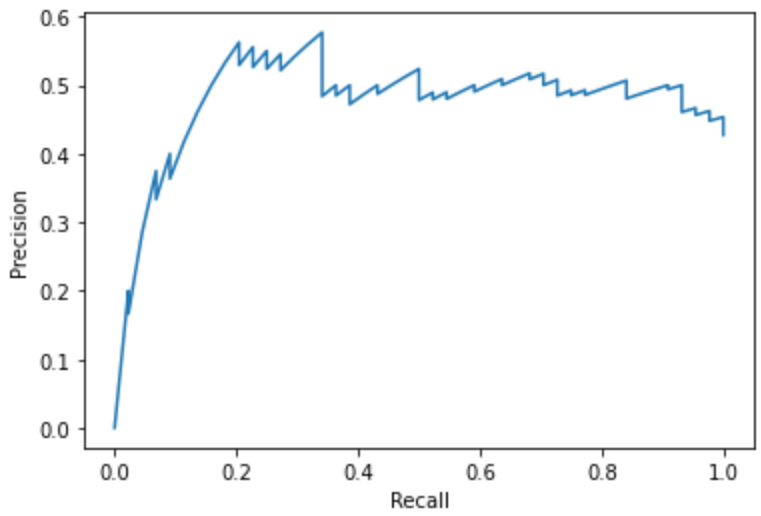
Conclusion:
The results show that the 80-20 split performed the best in all splits, using TF-IDF weights produced better results than using TF-ICF weights..

Question 3.

To begin with, a CSV file was used to create a DataFrame, and the query df[df[1] == "qid:4"] was utilized to filter the DataFrame to only consider queries with qid=4. A unique list of relevance judgment labels was created and stored in a variable named relevance. The unique values in the DataFrame were [0 1 3 2]. Next, the total number of files was determined by rearranging the query-url pairs in order of maximum DCG, resulting in a massive number expressed in scientific notation.

Moving forward, the nDCG was computed for the entire dataset and the top 50 articles, utilizing the relevance labels and feature 75. To obtain this, the DCG was divided by the optimum DCG. The nDCG at 50 was calculated to be 0.35612494416255847,
And nDCG for the entire dataset was calculated to be 0.5784691984582591.

Finally, for queries with qid:4 and relevance labels greater than 0, the relevance labels were sorted based on feature 75. Precision and recall were determined for each step of the sorted relevance labels, and the precision-recall curve was generated.


0.6
0.5
0.4
0.3
0.2
0.1
0.0
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Recall