AIWR – UE20CS332

Assignment – 1

1) Justification for preprocessing techniques used

Tokenisation:

Tokenization is the process of breaking up a text into smaller units, called tokens, which are typically words or phrases. It is used to prepare text for analysis, such as in natural language processing, machine learning, and data mining. Tokenization helps to standardize and organize text data, making it easier to work with and extract meaningful information.

Performing tokenization is important for creating a search engine because it helps to break up the text into individual units, such as words or phrases, which can be indexed and searched more efficiently. By tokenizing the text, the search engine can quickly identify and match relevant keywords or phrases from the search query to the corresponding tokens in the indexed text, resulting in more accurate and relevant search results.

Stopword Removal :

Stopword removal is important for creating a search engine because it eliminates common, non-informative words like "and" and "the" from the text, allowing the search engine to focus on more meaningful and relevant words. This can improve the accuracy and efficiency of the search engine, as it reduces the number of irrelevant results returned and helps to prioritize the most important words in the query.

Casefolding :

Casefolding is important for creating a search engine because it normalizes text by converting all letters to lowercase or uppercase. This reduces the complexity of the search process and ensures that searches for the same word in different cases return the same results. This can improve the accuracy and consistency of the search engine, making it more user-friendly and easier to use.

Lemmatization:

Lemmatization is important for creating a search engine because it reduces words to their base or dictionary form, known as a lemma. This helps to group together different forms of the same word, such as "run" and "running", improving the accuracy and relevance of the search results. It also reduces the dimensionality of the search space, making the search process faster and more efficient.

Stemming:

Stemming is important for creating a search engine because it reduces words to their root or stem form, which helps to group together different variations of the same word. This can improve the recall of the search engine by ensuring that all variations of a word are included in the search results. It also reduces the complexity of the search space, making the search process faster and more efficient.

2) Justification for choosing an appropriate datastructure

There are several data structures that can be used when creating a search engine , the one we are using here is *Inverted index*

Inverted Index: This is the most common data structure used in search engines. It is essentially a mapping of terms to the documents that contain them. Each term in the index is associated with a list of documents in which it appears.

How it works?

The code you provided is a Python function that takes a list of tokenized tweets as input and returns an inverted index as output. The function works by iterating over each document (tweet) in the corpus and then iterating over each word in the document. For each word, the function checks whether it already exists in the inverted index dictionary. If the word is not yet in the dictionary, it adds an empty list for the word. Then,

the function appends the document index (tweet index) to the list associated with the word in the inverted index dictionary.

Once all the tweets have been processed, the function goes through the inverted index dictionary and removes any duplicate document indexes that may have been added during the iteration process. This step ensures that each document is only associated with a given word in the inverted index once.

Finally, the function returns the completed inverted index dictionary.

The main reasons for using inverted index would be its *speed, reduced storage space, Ranking, Handling multiple queries, Flexibility*

Implementation of Boolean query:

The Boolean model implementation is necessary because for information retrieval boolean operators, such as AND, OR, and NOT, to express complex queries and retrieve relevant documents.

A boolean model is a powerful tool for precise and specific queries, but it may not always be necessary or appropriate for all types of search engines.

The implementation is Boolean query for search engine using a Inverted index. The boolean_query function first parses the query using the parse_query function. It then evaluates the postfix notation by iterating over the tokens and performing the appropriate operation for each token. If the token is a query term, it retrieves the set of document IDs from the inverted index. If the token is an operator (AND, OR, NOT), it pops the appropriate number of operands from the stack and performs the operation. The result is pushed back onto the stack. Finally, the function returns the set of document IDs that satisfy the query.

3) Justification for choosing a typical similarity scoring scheme:

a) Likelihood language model : A likelihood language model similarity scoring scheme is a common choice for natural language processing tasks, such as information retrieval and text classification. It is built on the probabilistic language modeling paradigm, which calculates the likelihood of witnessing a set of words in a particular situation.

Here are some justifications for choosing a likelihood language model similarity scoring scheme:

i. Intuitive interpretation: Likelihood language model similarity scores have an intuitive interpretation as the probability of observing a given query given a document. This makes it easy to understand the relevance of the document to the query.

ii. Generalizability: Likelihood language model similarity scoring is a general approach that can be used for a variety of tasks, including information retrieval, text classification, and question answering.

iii. Flexibility: Likelihood language model similarity scoring can be adapted to different types of language models, including n-gram models, neural network models, and probabilistic context-free grammars. This makes it flexible and adaptable to different applications.

iv. State-of-the-art performance: Likelihood language model similarity scoring has been shown to achieve state-of-the-art performance in several natural language processing tasks, including information retrieval and text classification.

Overall, a likelihood language model similarity scoring scheme is a reliable and effective method for comparing the similarity

between documents and queries in natural language processing applications.

b) Retrieve Text using similarity index: A Retrieve Text using similarity index similarity scoring scheme is a popular technique used in information retrieval systems to match a user's query with relevant documents or texts.

The justification for choosing this approach can include the following reasons:

i. Flexibility: A similarity index scoring scheme can be applied to a wide range of text data, including documents, web pages, and social media posts. This flexibility makes it a popular choice for search engines and recommendation systems.
ii. Efficiency: Similarity indexing algorithms can efficiently identify relevant documents or texts by comparing the query to a pre-processed index of terms or phrases. This process can be done quickly, even with large datasets.
iii. Accuracy: Similarity indexing algorithms can accurately match user queries to relevant documents or texts, even when the text is unstructured or contains errors or typos.
iv. Ranking: Similarity indexing algorithms can rank the relevant documents or texts in order of relevance, making it easier for the user to find the most relevant information quickly.
v. Personalization: Similarity indexing algorithms can be customized to the user's preferences and search history, providing more personalized search results.

Overall, a Retrieve Text using similarity index similarity scoring scheme is a powerful tool for information retrieval systems, providing a flexible, efficient, accurate, and personalized way to match user queries with relevant documents or texts.

c) Semantic Matching: Semantic matching similarity scoring schemes are often chosen in natural language processing and information retrieval tasks where the meaning of the text is more important than its surface form. Such schemes aim to capture the semantic similarity between two pieces of text, even if the wording is different.

   One justification for using a semantic matching similarity scoring scheme is that it can improve the accuracy of information retrieval systems. For example, when a user enters a query in a search engine, a semantic matching algorithm can help identify relevant documents that may use different words to describe the same concept. This can lead to more accurate and relevant results.