**PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE**

**ACADEMIC YEAR: 2023-24**

## **DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT**

**CLASS: B.E. SEMESTER: I**

**SUBJECT: LP-IV**

|  |  |
| --- | --- |
| **ASSIGNMENT NO.** | A3 |
| **TITLE** | **Stop Word Removal** |
| **PROBLEM STATEMENT /DEFINITION** | Write a program for pre-processing of a Text Document: Stop Word Removal. |
| **OBJECTIVE** | * To learn and understand the process of text pre-processing. * To understand stop word removal process. |
| **OUTCOME** | * Implement stop word removal to remove frequently occurring non-important words like a, an, the, in, from, etc. * Implement pre-processing of text document for information retrieval. |
| **S/W PACKAGES AND**  **HARDWARE APPARATUS USED** | 64-bit Ubuntu OS,  8GB RAM(Recommended),  Jupyter Notebook,  Programming Language: Python |
| **REFERENCES** | 1. C.J. Rijsbergen, &quot;Information Retrieval&quot;, (http://www.dcs.gla.ac.uk/Keith/Preface.html)  2. W.R. Hersh, ―Information Retrieval: A Health and Biomedical Perspective‖, Springer, 2002.  3. G. Kowalski, M.T. Maybury. &quot;Information storage and Retrieval System&quot; , Springer, 2005 |
| **STEPS** | Refer to theory, algorithm, test input, test output |
| **INSTRUCTIONS FOR**  **WRITING JOURNAL** | 1. Date  2. Assignment no.  3. Problem definition  4. Learning objective  5. Learning Outcome  6. Concepts related Theory  7. Algorithm  8. Test cases  10. Conclusion/Analysis |

**Prerequisites:**

**Concepts related Theory:**

**Text Pre-Processing:** Text data derived from natural language is unstructured and noisy.

Text pre-processing involves transforming text into a clean and consistent format that can then be fed into a model for further analysis and learning. Text pre-processing techniques may be general so that they are applicable to many types of applications, or they can be specialized for a specific task.

Steps in text pre-processing are:

1. **) Segmentation**
2. **) Tokenization**
3. **) Change Case**
4. **) Spell Correction**
5. **) Stop Words Removal**
6. **) Stemming**
7. **) Lemmatization**

**Segmentation**

Segmentation involves breaking up text into corresponding sentences. While this may seem like a trivial task, it has a few challenges. For example, in the English language, a period normally indicates the end of a sentence, but many abbreviations, including “Inc.,” “Calif.,” “Mr.,” and “Ms.,” and all fractional numbers contain periods and introduce uncertainty unless the end-of-sentence rules accommodate those exceptions.

**Tokenization**

The tokenization stage involves converting a sentence into a stream of words, also called “tokens.” Tokens are the basic building blocks upon which analysis and other methods are built.

Many NLP toolkits allow users to input multiple criteria based on which word boundaries are determined. For example, you can use a whitespace or punctuation to determine if one word has ended and the next one has started. Again, in some instances, these rules might fail. For example, *don’t*, *it’s*, etc. are words themselves that contain punctuation marks and have to be dealt with separately.

**Change Case**

Changing the case involves converting all text to lowercase or uppercase so that all word strings follow a consistent format. Lowercasing is the more frequent choice in NLP software.

**Spell Correction**

Many NLP applications include a step to correct the spelling of all words in the text.

**Stop-Words Removal**

"Stop words" are frequently occurring words used to construct sentences. In the English language, stop words include *is*, *the*, *are*, *of*, *in,* and *and*. For some NLP applications, such as document categorization, sentiment analysis, and spam filtering, these words are redundant, and so are removed at the pre-processing stage.

**Stemming**

The term *word stem* is borrowed from linguistics and used to refer to the base or root form of a word. For example, *learn* is a base word for its variants such as *learn, learns, learning,* and *learned.*

Stemming is the process of converting all words to their base form, or stem. Normally, a lookup table is used to find the word and its corresponding stem. Many search engines apply stemming for retrieving documents that match user queries. Stemming is also used at the pre-processing stage for applications such as emotion identification and text classification.

**Lemmatization**

Lemmatization is a more advanced form of stemming and involves converting all words to their corresponding root form, called “lemma.” While stemming reduces all words to their stem via a lookup table, it does not employ any knowledge of the parts of speech or the context of the word. This means stemming can’t distinguish which meaning of the word *right* is intended in the sentences “Please turn right at the next light” and “She is always right.”

The stemmer would stem *right* to *right* in both sentences; the lemmatizer would treat *right* differently based upon its usage in the two phrases.

A lemmatizer also converts different word forms or inflections to a standard form. For example, it would convert *less* to *little*, *wrote* to *write*, *slept* to *sleep*, etc.

A lemmatizer works with more rules of the language and contextual information than does a stemmer. It also relies on a dictionary to look up matching words. Because of that, it requires more processing power and time than a stemmer to generate output. For these reasons, some NLP applications only use a stemmer and not a lemmatizer.

**NLTK**: NLTK is one of the libraries provided by python for removal of stop words. You can find a list of stop words in the corpus module. To remove stop words, you can divide your text into words and then remove the word if it exists in the list of stop words provided by NLTK.

**Algorithm:**

The algorithm is implemented as below given steps.

Step 1: The target document text is tokenized and individual words are stored in array.

Step 2: A single stop word is read from stopword list.

Step 3: The stop word is compared to target text in form of array using sequential search technique.

Step 4: If it matches , the word in array is removed , and the comparison is continued till length of array. Step 5: After removal of stopword completely, another stopword is read from stopword list and again algorithm follows step 2. The algorithm runs continuously until all the stopwords are compared.

Step 6: Resultant text devoid of stopwords is displayed, also required statistics like stopword removed, no. of stopwords removed from target text, total count of words in target text, count of words in resultant text, individual stop word count found in target text is displayed.

**Conclusion:** Pre-processing of text documents was done and the process of stop word removal was understood using NLTK library in python.

**Review Questions**:

Q1. What are stemming and lemmatization?

Q2. What is count vectorization?

Q3. What is word embedding?

Q4. What do you mean by Lemmatization in NLP?

Q5. What are stop words?