A

Report On

TEXT DOCUMENT PROCESSING FOR CLASSIFICATION

Department of Computer Science and Technology

Submitted by:

Mortha Sai Sriram	**Id No:**510517056
Sajal Soni	**Id No:**510517061
Bhanu Pratap	**Id No:**510517074
Paras Jain	**Id No:**510517077
Shashwat Srivastava	**Id No:**510517083

Head of the Department

Dr.Sulata Mitra

Mentor

Dr.Asit Kumar Das

Acknowledgement:

We would like to express our gratitude towards our mentor, Dr.Asit Kumar Das, Department of Computer Science & Technology, for guiding us throughout this semester in making this mini project. Also we thank our respected Head of the Department, Dr.Sulata Mitra, and all other professors, for giving us this opportunity of learning something different through this mini project.

This work would surely help us in some way or the other in the future.

CONTENTS

INTRODUCTION:

Natural language processing (NLP) is a sub-field of Artificial intelligence that is focused on enabling computers to understand and process human languages, to get computers closer to a human-level understanding of language. Computers don't yet have the same intuitive understanding of natural language that human do. They can't really understand what the language is really trying to say. In a nutshell, a computer can't read between the lines. That being said, recent advances in Machine learning(ML) have enabled computers to do quite a lot of useful things with Natural language!

Deep Learning has enabled us to write programs to perform things like language translation, semantic understanding, and text summarization. All these things add real-world value, making it easy for you to understand and perform computations on large blocks of text blocks without the manual effort.

Text Processing

- Text processing means the manipulation of text, especially the transformation of text from one format to another.
- In computing, the term text processing refers to the discipline of mechanizing the creation or manipulation of electronic text.
- This can further be used in text retrieval and data mining.
- The goal of text processing is to discover relevant information in text by transforming the text into data that can be used for further analysis. Text processing accomplishes this through the use of a variety of analysis methodologies.
- One such method to do so is natural language processing. It is a method used to program computers to process and analyze large amounts of natural language data.

Why Text Processing???

- The task of retrieving data from a user-defined query has become so common and natural in recent years that some might not give it a second thought.Query retrieval can be described as the task of searching a collection of data, be that text documents, databases, networks, etc., for specific instances of that data.
- One such way which helps in the task of retrieving data is by finding the tf-idf of all the words in the given corpus of documents,which we have implemented in this project.
- TF-IDF is an efficient and simple algorithm for matching words in a query to documents that are relevant to that query. From the data collected, we see that TF-IDF returns documents that are highly relevant to a particular query. If a user enters a query,then using TF-IDF we may return the most relevant document to the query given.This method is used by various search engines for providing us with the results we get on any search query.

NLTK (Natural Language Toolkit) In PYTHON

- It provides a suite of libraries and programs for symbolic and statistical Natural Language Processing (NLP) for English written in the Python programming language. Developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the 'University of Pennsylvania'. NLTK includes graphical demonstrations and sample data. NLTK supports classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionalities.

- In our project, we used these libraries extensively for demonstration of how a set of documents are processed for data retrieval and analysis.

- NLTK in python provides a large number of libraries for operations related to text processing. For this demonstration we imported a set of documents. Initially we used a set of 20 documents which we imported from Newsgroup20,a dataset containing around 20000 newsgroup documents.

- We further performed various operations on all the documents such as tokenization, stopwords removal, lemmatization and finally finding the tf-idf of the data in these documents.

Tokenization

- **Tokenization** is the method of breaking up a sequence of strings into pieces such as words, keywords, phrases, symbols and other elements called tokens.
- We break the document into words. It involves removal of all the punctuation marks from the document which leaves us with only the words defined in the document.
- Further, all the words of the document are stored in a list.

Library functions used: -

RegExpTokenizer: This is a tokenizer predefined in python in the package nltk.It removes all the punctuations from the document.

Tokenize: It takes all the words from the document and stores them in a list. This is also defined in the package nltk.

```
import nltk
from nltk.tokenize import word_tokenize,RegexpTokenizer
tokenizer=RegexpTokenizer(r'\w+')
word_tokens=[]
word_tokens=tokenizer.tokenize(data)
```

An example illustrating what this function does:

'This sentence is just an example of word tokenizer, regexptokenizer.'

Output usingtokenize: ['This', 'sentence', 'is','just', 'an', 'example', 'of', 'word', 'tokenizer', ',' , 'regexptokenizer', '.']

Output using RegexpTokenizer: ['This', 'sentence', 'is', 'just', 'an', 'example', 'of', 'word', 'tokenizer', 'regexptokenizer']

Stopwords

- **Stop words** are natural language **words** which have very little **meaning**, such as "and", "the", "a", "an", and similar **words**.
- We have to remove all the useless data from the documents.So,we remove stopwords from all the documents by using the library functions of python.
- The “stopwords” function defined in nltk.corpus is used for removing the stopwords.
- Stopwords are available for 14 different languages in nltk.corpus.

```
import nltk
from nltk.corpusimportstopwords
stop_words = set(stopwords.words('english'))
filtered_words=[]
for w in word_tokens:
    if w not in stop_words:
filtered_words.append(w)
print(filtered_words)
```

An example illustrating what this function does:

‘This sentence is just an example of stop words’

Output:

[’sentence’, ‘just’, ’example’, ‘stop’, ‘words’]

Conversion to root form

- Every document may have numerous words which could be derived from the same word i.e., the root word. For proper processing of data, we need to reduce all the words to their root form.

Studies	Study
Studying	Study
Studied	Study

- There are two methods by which this can be done.
  1)Stemming
  2)Lemmatization

Stemming

This algorithm works by cutting off the end or the beginning of the word,considering a list of common prefixes and suffixes that can be found in an inflected word. This indiscriminate cutting can be successful in some occasions, but not always, and that is why we affirm that this approach presents some limitations.

Porterstemmer is the library function in python used for stemming.

For example:

Studies	Studi
Studying	Study

Here the stemmed word “studi” has no literal meaning and because of these disadvantages, lemmatization is preferred over stemming.

```
from nltk.stem import PorterStemmer

ps = PorterStemmer()

stem_words=[]

for w in filtered_words:

stem_words.append(ps.stem(w))
```

Lemmatization

This method makes use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word.

Studies	Study
Studying	Study

Lemmatization consumes more time than stemming but the result obtained is optimized precisely and thus is considered better than stemming.

We,in our project have also used lemmatization for obtaining the rootwords. The library function used is **WordNetLemmatizer**.

```
import nltk

from nltk.stem import WordNetLemmatizer

lemmatizer=WordNetLemmatizer()

print(lemmatizer.lemmatize(w))
```

TERM FREQUENCY

- Term frequency is defined as the frequency of all the words in a particular document.
- It indicates the significance of a particular **term** within the overall document.Term frequency can be defined mathematically as:

Term frequency($tf_{i,j}$)= $n_{i,j}/\sum n_j$for ith word in jth document.

```
for w in lem_words:
        count=0
for j in lem_words:
                if w not in final_words.keys():
                if (w==j):
                        count=count+1
        if w not in final_words.keys():
        final_words.update({w:count})
for k in final_words.keys():
        final_words[k]=final_words[k]/i
final.append(final_words)
```

Inverse document frequency:

Inverse document frequency(idf) is a method to calculate the weight of the words across all the the document taken into consideration.

This method helps to evaluate how important a word is to a document in the corpus,i.e how rare the word is in any document.idf decreases as the frequency of the word in a document increases.

Idf for any word is mathematically defined as, idf (w) = log (N/df_w)

N=Total number of documents taken into consideration.

dfw=The number of document in which the word is present.

Code for implementing idf

```
for fi in final:
        for w in fi.keys():
                total[w]=total[w]+1
for w in total.keys():
        total[w]=math.log10(docno/float(total[w]))
for w in total.keys():
        tfidf.update({w:[]})
```

Tf-idf:

- In information retrieval, tf–idf or TFIDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.
- It is often used as a weighting factor in searches of information retrieval, textmining, and usermodeling.The tf–idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general.
- Tf-idf for each word is calculated by multiplying the tf of a word and its corresponding idf.

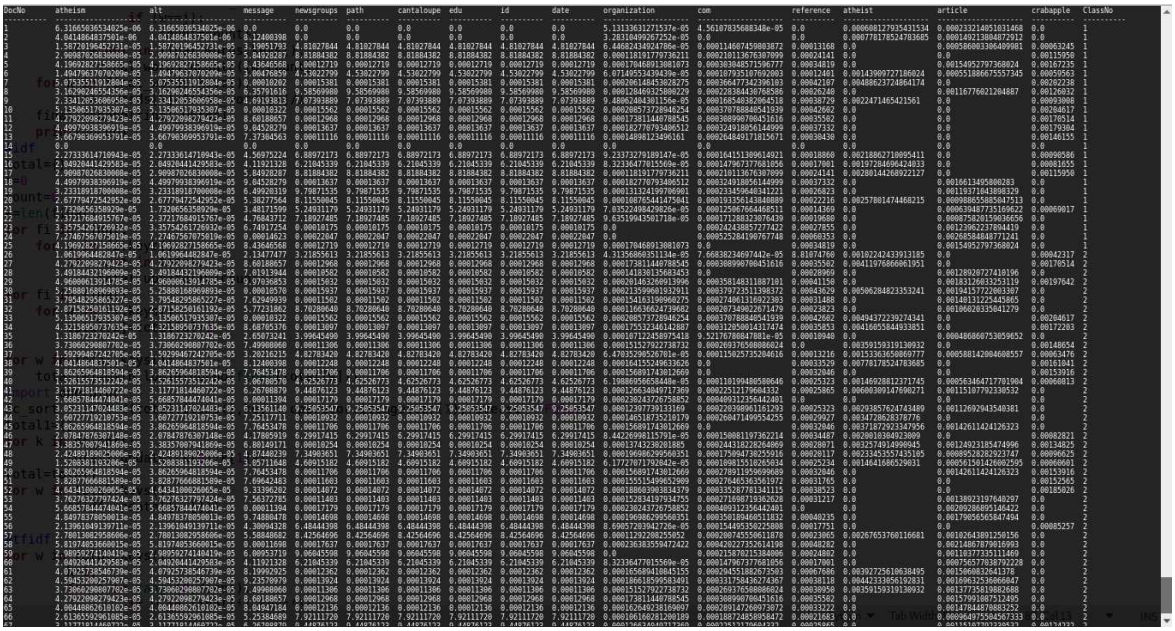
Code for tf-idf:

```
for i in range(0,docno):
        if w in final[i].keys():
            k=(final[i][w])*total[w]
            tfidf[w].append(k)
        else:
        k=0
            tfidf[w].append(k)
```

Database connectivity:

- Having calculated the tf-idf of every word,representation of the evaluated result in the form of tables is necessary.This is done by the use of database.Python offers various database engines and SQLite3 is one such database engine which we have used in our project. It is self-contained, serverless, zero-configuration and transactional.

Output from SQLITE3 Database:



Conclusion:

- Finally we are able to pick up each and every word of some of the documents at a time and make a tabular form in such way that corresponding to words in rows and document number in columns, tf-idf data are placed in the cells.
- So our program finally provides us the tf-idf value of the words present in documents(given to us as input).

WHAT WE WISH TO ACHIEVE:

- Having calculated the tf-idf of the corpus,we finally aim for our project to be able to classify the free documents into their respective categories.This can be achieved with the help of classifiers.

- NLTK helps us in doing so with the help of built-in classifiers.We wish to study different types of classifiers such as:

    - Perceptron.
    - Naive Bayes.
    - Decision Tree.
    - Logistic Regression.
    - K-Nearest Neighbor.
    - Artificial Neural Networks/Deep Learning.
    - Support Vector Machine.

- Using different classifiers,the accuracy of all of them will be measured and hence we will be able to conclude which classifiers are best suited for sorting the set of documents into their respective classes.

REFERENCES

- https://docs.oracle.com
- https://medium.freecodecamp.org
- https://python.org
- https://pythonprogramming.net
- Text mining by David Robinson