**CE706 - Information Retrieval 2023**

**Assignment 1**

2202396

# Elasticsearch

*All the code for the Program is done in one single file. First and for most using an online tool the jasonl.gz file is first converted to json file. I have used online tool “ezyZip” for that. The selection of first 1000 dataset points is done outside of the program and are loaded in a new separate File named “1000\_Sample.jason”. The code starts with importing different packages.*

*![A screenshot of a computer

Description automatically generated with medium confidence]()*

*I have imported the nltk library into this program and I’m using it’s built in function Analyzer to perform tasks for Tokenization, Stemming, etc.*

*![Text

Description automatically generated]()*

# Stop Words Removal

*Throughout the program there are separate analizers for different tasks according to my understanding of the task. For stop words removal I’m using built in function for that.*

*![Graphical user interface, application, Word

Description automatically generated with medium confidence]()*

*In order to test my program outputs I have been using the analyzer to figure out final results from my program. The aim is to make sure if it’s printing the right output and removing the stop words which in this example is “the”.*

*![Text

Description automatically generated]()*

# Tokenization.

*It is a really important step which breaks the document into small words. In elastic search we used it with other tasks like analyzer and stemming in order to get best results possible.*

*There are many different ways of tokenization depending on the type of the problem. My approach is to use lowercase as a token.*

*![A picture containing text

Description automatically generated]()*

*![Text

Description automatically generated]()*

# Stemming.

*In stemming we basically reduce the words to their actual root form for example if it’s “running” we will convert it into “run”. Again the implementation of stemming is quite simple and straight forward. Here is a look of my code with the output function*

*![Graphical user interface, application, Word

Description automatically generated]()*

*![Graphical user interface, text, application

Description automatically generated with medium confidence]()*

# TF\_DIF.

*It is used to measure the relevance of the particular word in a document. I’m using the TFDIF equation from the elastic search documentation*

<https://www.elastic.co/guide/en/elasticsearch/reference/current/index-modules-similarity.html>:

**

# Named entity recognition.

*For named entity recognition I implemented in python and for that I am importing spacy with Jason. To solve the problem, I’m importing the NER English model into the program and providing the program with the Jason dataset as I have solved my program directly in python. Once the Jason data file is loaded, it is looped through each time in data and each tie we keep on performing NER on the text data we have in Jason file. And at the end I’m saving the data into the new Jason file. Here is snapshot of the code.*

*![Graphical user interface, text, application, email

Description automatically generated]()*

# Mapping / Indexing

*It is a process of defining the datatype and structure of indexing of the data in the original file. It defines how to data is searched and analysed.*

*![Text

Description automatically generated]()*

# Searching

*In elasicsearch, searching is done using a query, it is a powerful tool which supports a wide range of searches. There are different kinds of queries for different operations. I have tried combination of different things in my queries. Some of these are given below.*

*In my first search I’m using must match, bool which basically combines multiple search querues*

Graphical user interface, text, application

Description automatically generated

*In the second one I simply tried to apply simple query using match, title, which will basically result us files with the same title in the datafile of our 1000 entries that we processed in the beginning.*

*![Text

Description automatically generated with medium confidence]()*

*In the third query, I went into a bit more detail and used bool, must, should and sort all together to make it much more specific. This is my code snapshot*

*![Diagram

Description automatically generated]()*