# ML BASED SOLICITATION IN FEDERAL TRANSCRIPTS

BITS SSZG628T: Dissertation

By

**DILIP PRASAD J**

2020MT12208

Dissertation work carried out at

**DXC TECHNOLOGY, CHENNAI**

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE**

**PILANI (RAJASTHAN)**

**APR 2022**

# ML BASED SOLICITATION IN FEDERAL TRANSCRIPTS

BITS SSZG628T: Dissertation

By

**DILIP PRASAD J**

2020MT12208

Submitted in partial fulfillment of M.Tech. Software Systems

Dissertation work carried out at

**DXC TECHNOLOGY, CHENNAI**

Under the supervision of

**Dr Hans Beck**

**DXC Technology, Germany**

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE**

**PILANI (RAJASTHAN)**

**APR 2022**

*CERTIFICATE*

This is to certify that the Dissertation entitled "ML BASED SOLICITATION IN FEDERAL TRANSCRIPTS" and submitted by Dilip prasad Jayakumar having ID-No. 2020MT12208 for the partial fulfillment of the requirements of Masters of technology in software systems degree of BITS, embodies the bonafide work done by him/her under my supervision.

Hans

_____

Signature of the Supervisor

Place : _Germany_____

Date : __20/04/2022 _____

Dr Hans Beck

Advisor, Data Science

Germany

_____

Name, Designation & Organization &Location

# Acknowledgements

I would like to express my deep gratitude to Dr. Hans Beck my research supervisor, for their patient guidance, enthusiastic encouragement, and useful critiques of this research work. I would also like to thank Mrs. Hema Priya, for her advice and assistance in keeping my progress on schedule.

Finally, I wish to thank my wife and parents for their support and encouragement throughout my study.

Dilip prasad Jayakumar

# ABSTRACT

Federal archive has evolved from Middle Ages to modern period, where archives are retained proofs of political and genealogical claims based on the authenticity. Archives do not neutrally store documents; rather, objects captured through archival practices are transformed into knowledge. In the creation phase of archives, records growth is expounded by modern electronic systems. Records will continue to be created and captured by the organization at an explosive rate as it conducts the business of the organization.

To perform search any specific transcripts without any pointers from the humongous set of files one can consider as cumbersome or almost impossible task. However, this have been the case for long where people are dedicatedly assigned to perform this as a full-time job. With the advent of technology and cheaper hardware has paved way to innovative and resource intensive computations executing complex algorithms to achieve what was once unimaginable.

Currently, the search operation on the German transcripts is done using Regex to perform a word-by-word search in all the files linearly. Where the time taken to complete the search increases based on the number for files which is inefficient. Additionally, the search only works with straight forward approach and will not be able to list mentions for related synonyms.

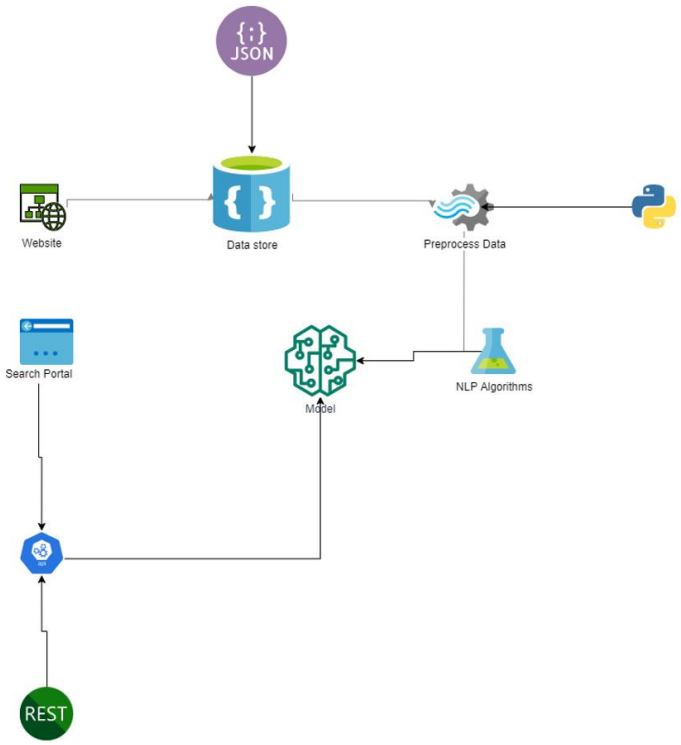
So, in this dissertation we will implement Natural Processing Techniques to not only understand the transcripts provided in German language but also perform innovative search mechanism. Where we could search by specific speaker and list all their mentions in all the archival documents and search with related synonyms that could be a potential match for that mention.

We will go a step ahead and review at Deep learning approaches to enhance the current NLP abilities to provide better search results.

For this Dissertation we will use nested transcripts from the below URL.
https://www.bundesarchiv.de/cocoon/barch/0000/index.html

# Contents

## High level project architecture




1. Based on our requirements we will approach the solution in a segment-by-segment approach. We will initially crawl the entire website from the given root URL, the intention is to find all the sub-URL or sibling pages where we could find the relevant data. This helps to filter out unwanted data. The crawled URL is queued to azure queue for data extraction.
2. Once we have url's queued to the azure queue, we could trigger the data extraction. Where the data from the given link is capture into a panda's data frame initially and then transferred to a CSV file. The data extracted is normally plain text instead of rich text with images. Even there are multimedia data, we will ignore the same or adapt to recognize in the future releases/ versions. Which could be used based on our convenience at any point in time. This comes handy in case of unplanned downtime or data corruption.
3. After fetching the data into csv, we will do the necessary pre-processing steps required to clean up the text to extract the necessary details. All the preprocessing steps are done using Python, with the help of open-source libraries readily available on the internet. This process could be triggered independently of the previous step. With the help of NLP algorithms, we could perform sentiment analysis on the given text, which helps us extract vital information which will be used during our search.

## NLP phases

There are 5 phases on any given NLP projects and the same will be performed here

1. Lexical & Morphological analysis
2. Syntactic analysis (parsing)
3. Semantic analysis
4. Discourse integration
5. Pragmatic analysis

## Preprocess raw text for Sentiment analysis

Data preprocessing is one of the critical steps in any machine learning project. It includes cleaning and formatting the data before feeding into a machine learning algorithm. For NLP, the preprocessing steps are comprised of the following tasks:

1. Tokenizing the string
2. Lowercasing
3. Removing stop words and punctuation
4. Stemming

## NLP Tasks

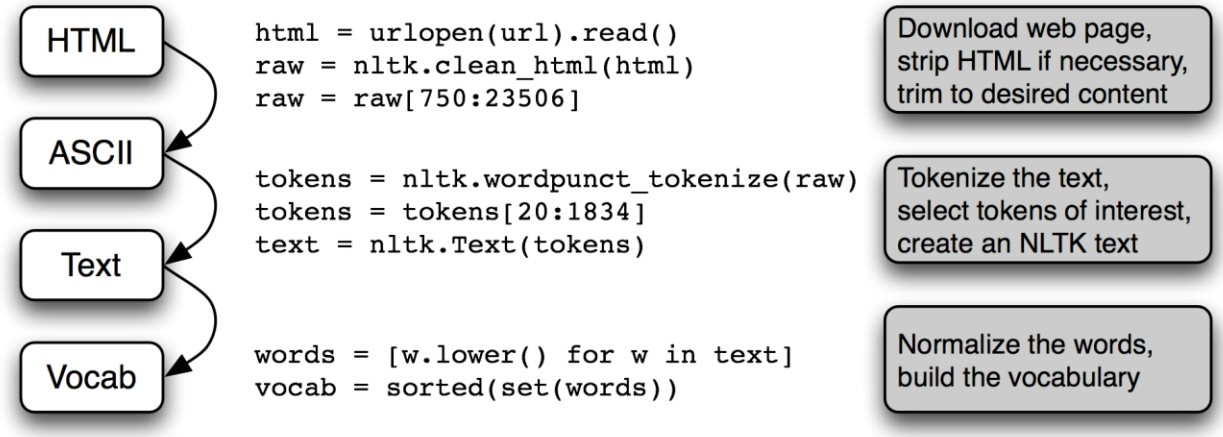
- Using a language library
- Building pipeline object
- Using token
- Part of speech tagging
- Understanding token attributes

## NLP Pipeline

There are multiple steps present in an NLP Pipeline

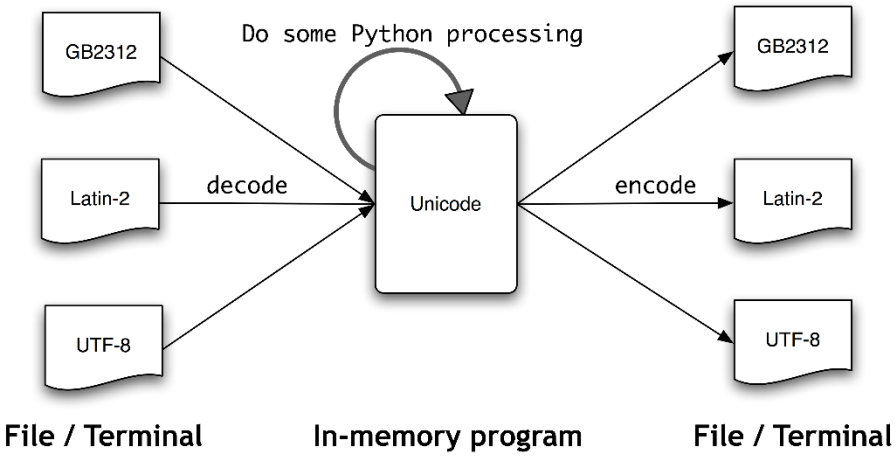
- Grab the HTML contents of the given URL
- then strip of the html tags /markup
- Get the data in text
- Build a Vocabulary with the words

Below is a typical example for a Webpage in english




However, ASCII is normally supported only for English, since we are using German, we need to find different encoding standards. So, we could use Unicode in short UTF standards. Each character is assigned a number, called a code point. In Python, code points are written in the form \uXXXX, where XXXX is the number in 4-digit hexadecimal form.

Since UTF supports only a limited set of characters, we could use UTF-8 which supports multiple bytes that can represent full range of Unicode characters.




## Web Data Scraping

We will scrape the html data from the source website with readily available libraries. One such library is the Beautiful-Soup, with python we could easily extract required text from a complex HTML.

For the Sake for processing the data extracted will be converted to JSON format (JavaScript Object Notation)

Scraping will be done on German Federal Archives - Link

### CSV Data

The extracted data is stored in a CSV file, which will be later used to be loaded into Data frame from pandas library for further processing.

### JSON Data

The Extracted data in JSON format will be categorized based on the topics or links from which it has been fetched from. This information can be considered as meta data.

1. Meeting Date
2. Start and End Time
3. Participants
4. Multiple Discussions in that meeting based on the context

## Storage of Data

Data is Stored in Document DB based Storage server, generally NO SQL DB. Where all these JSON are stored and retrieved. The Storage and retrieval mechanism will be done with the help of Python code. However for the sake of simplicity, we would be storing the files into a csv and use it as source for all steps.

## Preprocess data

The Data retrieved needs to be cleaned and processed before we can apply actual NLP algorithms. This Is a pivotal work that needs to be done

### Stop words

Use the NLTK libraries pre-built stop words to ignore

```
# Import nltk
import nltk

nltk.download('stopwords')

# Get English stopwords and print some of them
sw = nltk.corpus.stopwords.words('german')
sw[:5]
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
['aber', 'alle', 'allem', 'allen', 'aller']
```

## Creating a Corpus of Data

With the multiple conversational / legislative transcripts we could create our own corpus using NLTK (Natural Language toolkit)

Which will also help us project the Frequency distribution of words, to facilitate the validation of the data and we must find if its Balanced.

## Removing Stop words

```
[11] # Import nltk
     import nltk

     nltk.download('stopwords')

     # Get English stopwords and print some of them
     sw = nltk.corpus.stopwords.words('german')
     sw[:5]
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
['aber', 'alle', 'allem', 'allen', 'aller']
```

```
[12] #Remove German stopwords
     # Initialize new list
     words_ns = []

     # Add to words_ns all words that are in words but not in sw
     for word in words:
         if word not in sw:
             words_ns.append(word)

     # Print several list items as sanity check
     words_ns[:5]
```

```
['kabinettsprotokolle', 'online', '1', 'außen', 'innenpolitische']
```

## Stemming

Out of Porter stemmer and Lancaster stemmer, we will use Porter stemmer which primarily helps in search in alternative forms of words. Porter stemmer correctly handles the word lying (mapping it to lie), while the Lancaster stemmer does not.

We will more precisely use the latest version of the Porter stemmer known as “Snowball stemmer” which supports German and multiple other languages.

## Lemmatization

We would be performing lemmatization as it does not convert lying to lie but converts women to woman. This is useful only if we are compiling a vocabulary of texts and require a valid list of lemmas.

## Word Segmentation

Since most of the text we are handling are properly formatted, there would not be any need to perform this operation.

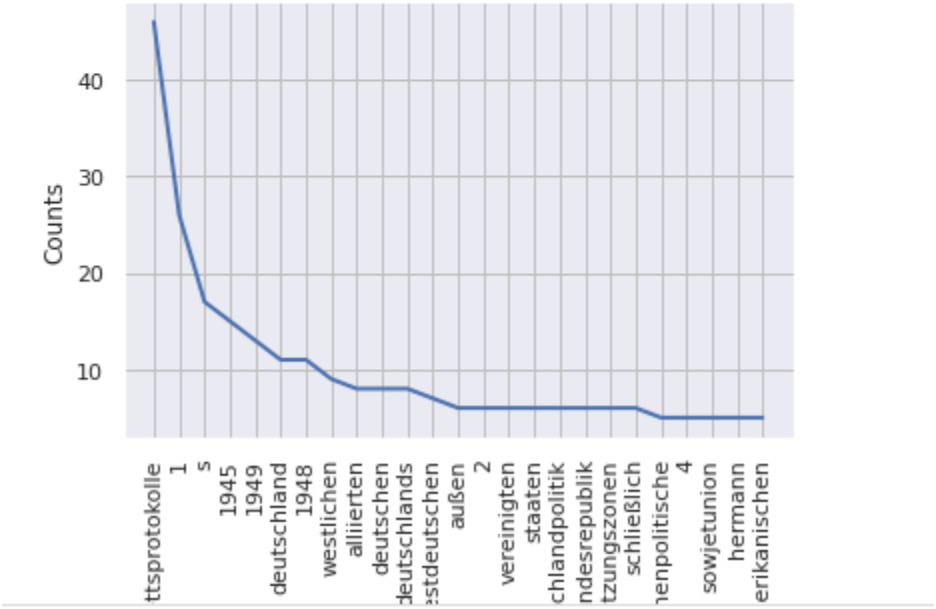
## Frequency Distribution

Let's look at a distribution of work frequency for a subset of data

```
#Get Word Frequency Distribution
import matplotlib.pyplot as plt
import seaborn as sns

#Display Inline chart
%matplotlib inline
sns.set()

#Create freq distribution
freqdist1 =  nltk.FreqDist(words_ns)
freqdist1.plot(25)
```

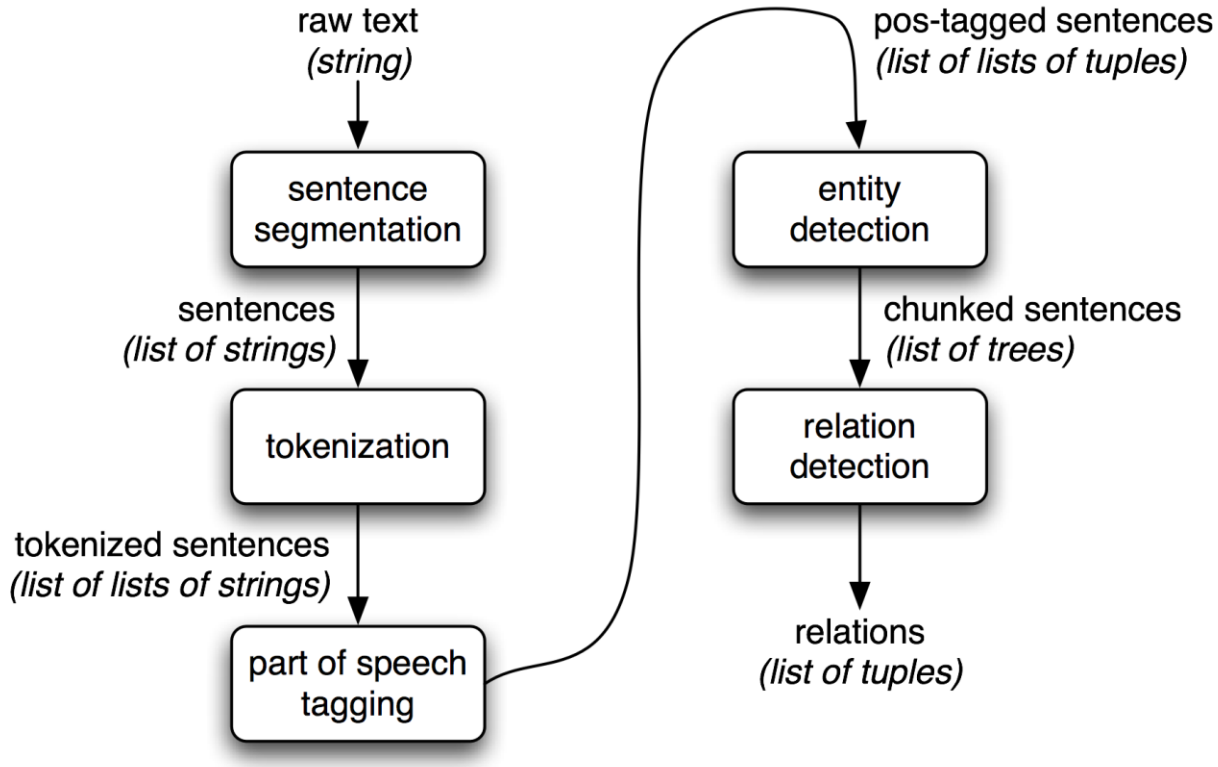



## Gender Recognition

Male and female names have some distinctive characteristics. Names ending in a, e and I are likely to be female, while names ending in k, o, r, s and t are likely to be male. As a part of this we will create a classifier to model these differences more precisely.

## Information Extraction Architecture

We will process using procedures discussed as above, raw text of the document is split into sentences using a sentence segmented, and each sentence is further subdivided into words using a tokenizer. Next, each sentence is tagged with part-of-speech tags, which will prove very helpful in the next steps




## Named Entity Recognition

The goal of a named entity recognition (NER) system is to identify all textual mentions of the named entities. This can be broken down into two sub-tasks: identifying the boundaries of the NE, and identifying its type

Named entity recognition is a task that is well-suited to the type of classifier-based approach that we saw for noun phrase chunking. We will build a tagger that labels each word in a sentence using the IOB format, where chunks are labeled by their appropriate type.

## TF-IDF

Term Frequency- Inverse Document frequency is a measure used in information retrieval. This score is used to determine importance of a term. When it comes to search engine TF-IDF plays a vital role to get a nearest match or a meaningful match (based on synonym)

Term Frequency – How frequent a word is occurring. If a word appears multiple times, then we could consider it has higher importance /score. This approach does not work always, as sometimes the words may not be frequent or many words could be frequent.

$$TF = \frac{Frequency\ of\ the\ word\ in\ the\ sentence}{Total\ number\ of\ words\ in\ the\ sentence}$$

IDF – Inverse Document Frequency- measures the informativeness of the term t

$$IDF = \frac{Total\ number\ of\ sentences}{Number\ of\ sentences\ containing\ that\ word}$$

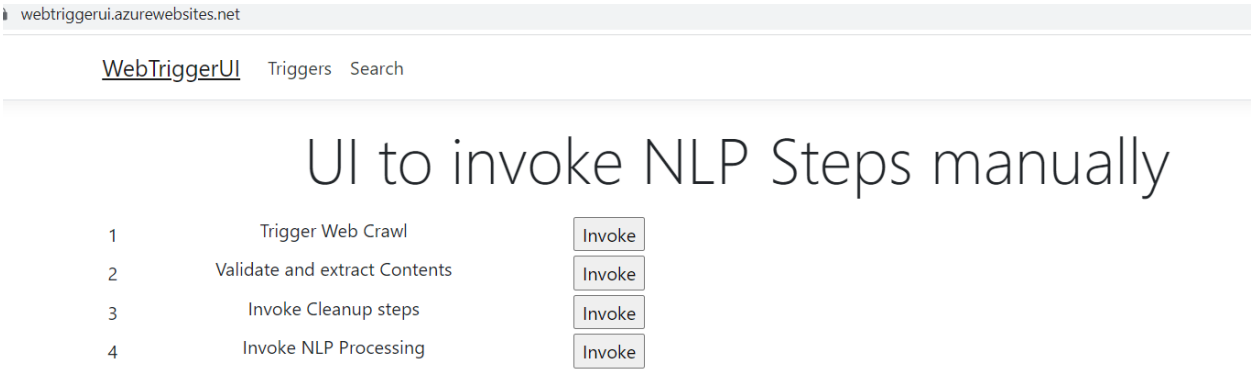
## Model Creation

Finally, after the model is trained based on the collected data, we will readily save the model physical and ship it. This enables us to consume the model and use it to get relevant occurrence in the search results.

## Trigger NLP Steps

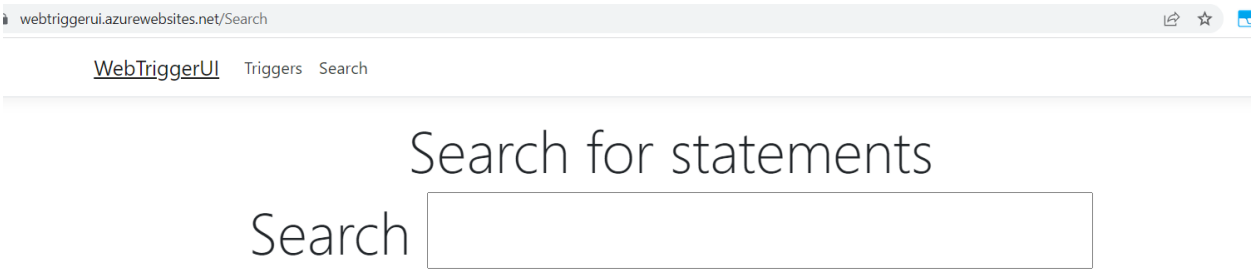
We have created an UI which invokes the python scripts to trigger the process. We could individually process one step after the other.

1. Triggers Web Crawling
2. Extracts and validates the web page contents
3. Invokes cleanup steps
4. Invokes NLP processing steps




## Search

Once the above-mentioned triggered process are completed, we could perform search and find results of our query




## Conclusion

We can crawl a web page in German language and use various data cleanup steps to extract the pivotal data then use NLP techniques to train. Finally, all these steps have paved way to help us search for results in an UI which fetches the relevant details (*which does not have to be a word-by-word match*)

Scope for improvement

- Use no server methodologies by making use of azure on demand or AWS functionless offerings
- Communicate between serverless functions to create a cascading experience
- Create a MLOPS pipeline to continuously train and deploy model for enhancements

## References


[1] https://en.wikipedia.org/wiki/National_archives
[2] https://en.wikipedia.org/wiki/Records_management