ML BASED SOLICITATION IN FEDERAL TRANSCRIPTS

BITS SSZG628T: Dissertation

By

DILIP PRASAD J

2020MT12208

Dissertation work carried out at

DXC TECHNOLOGY, CHENNAI

Submitted in partial fulfilment of **M. TECH** degree program

Under the Supervision of

BECK, HANS

DXC TECHNOLOGY, GERMANY

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE

PILANI (RAJASTHAN)

FEB 2022

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

Dilip prasad

Signature of the Student

Name: Dilip prasad, Jayakumar

Date: 26-02-2022

Place: India

Hans

Signature of the Supervisor

Name: Hans, Beck

Date: 26-02-2022

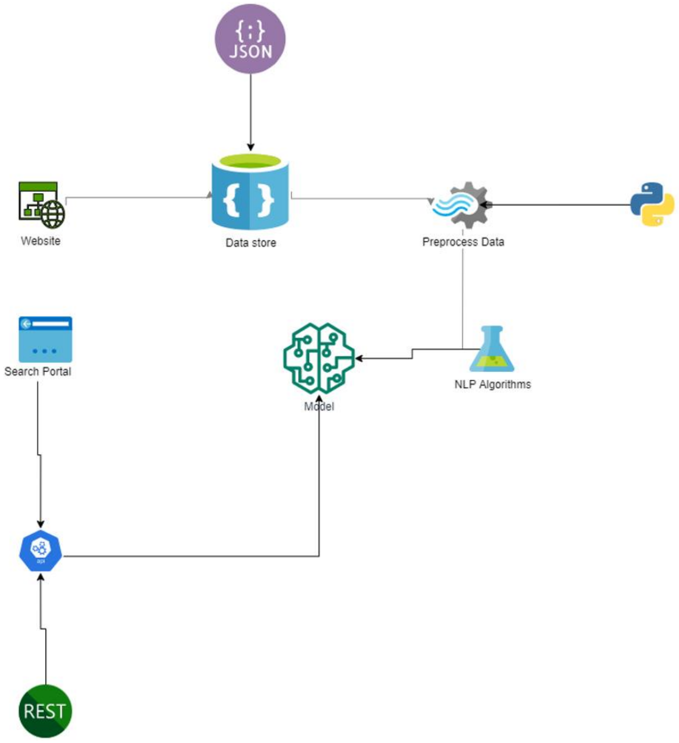
Contents

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:

- Tokenizing the string
- Lowercasing
- Removing stop words and punctuation
- Stemming

High level project architecture



Based on our requirements we will approach the solution in a segment-by-segment approach. We will extract the data from the main website and capture the details in a document DB. 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. After fetching the data, 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.

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.

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

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

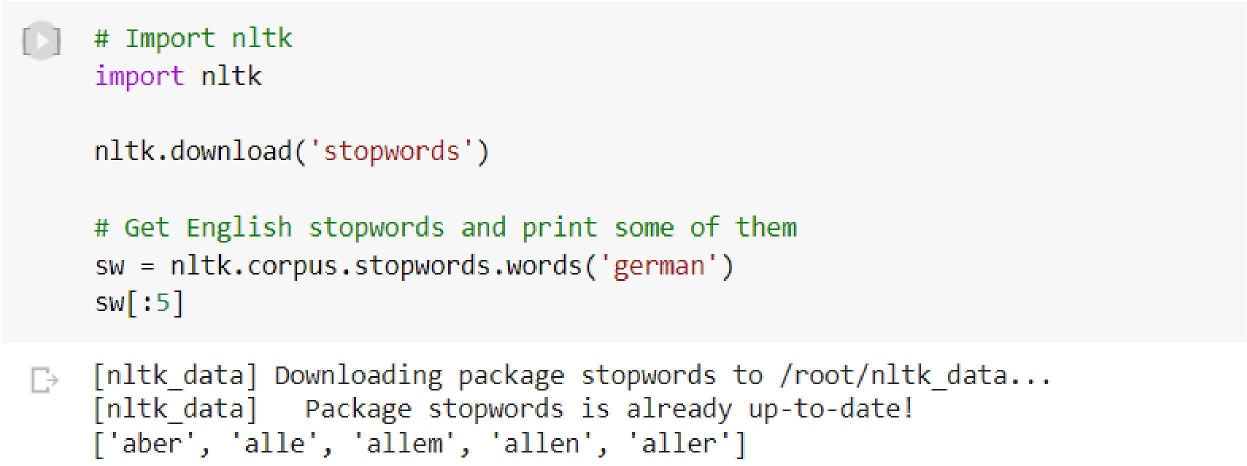
Data is Stored in Document DB based Storage server, generally NO SQL DB. Where all these JSON are stored and retrieved. The Storage and retrieval mechanism will be done with the help of Python code.

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]
```

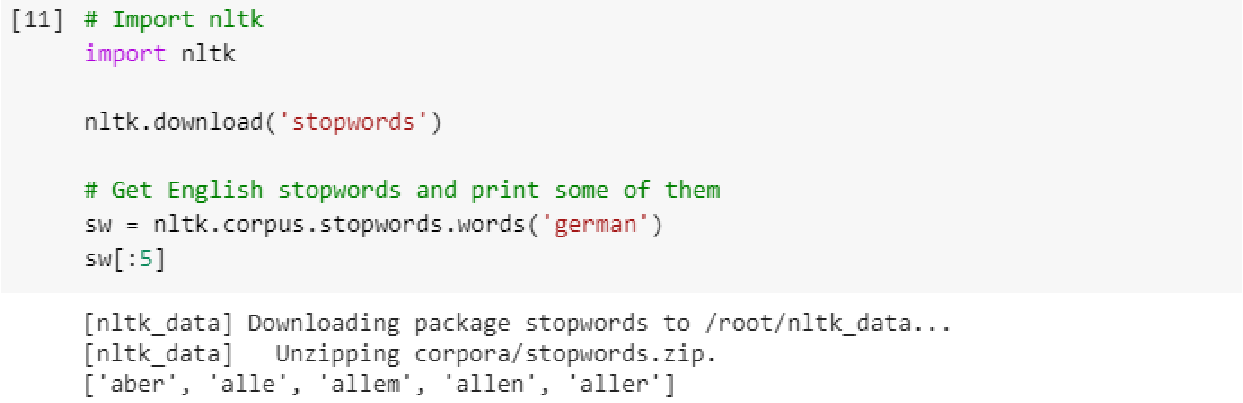
```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
['aber', 'alle', 'allem', 'allen', 'aller']
```

Creating a Corpus of Data

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

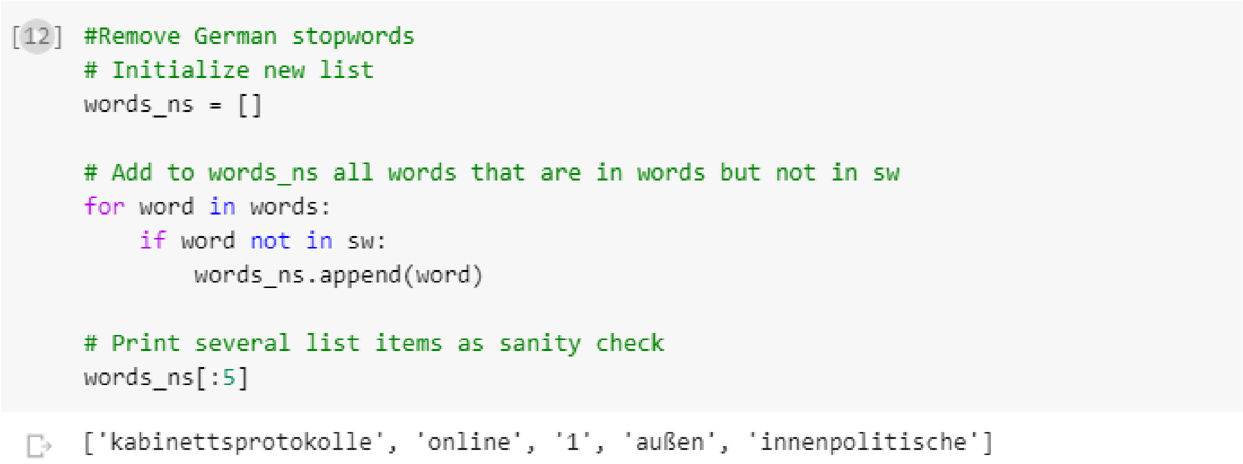
Which will also help us project the Frequency distribution of words, to facilitate the validation of the data and we must find if its Balanced.

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']
```

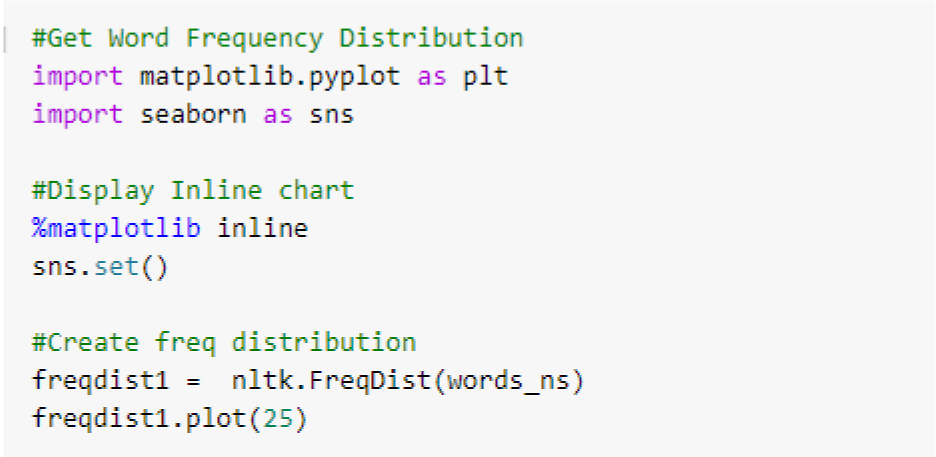


```
[12] #Remove German stopwords
     # Initialize new list
     words_ns = []

     # 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]
```

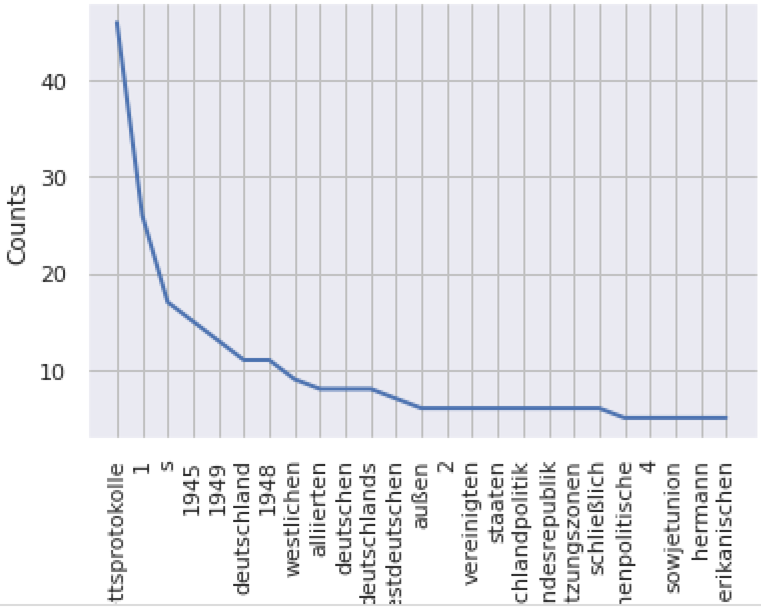
```
['kabinettsprotokolle', 'online', '1', 'außen', 'innenpolitische']
```

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)
```

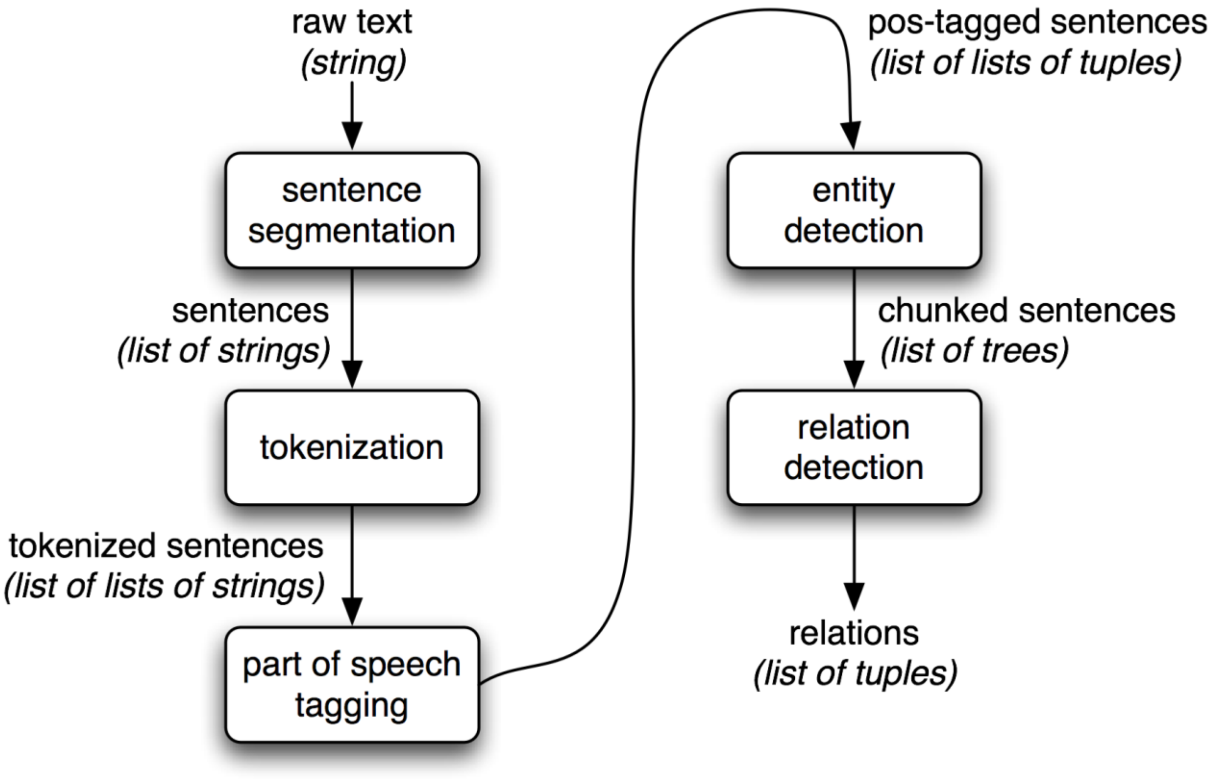


Gender Recognition

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.

Literature References

[1] https://en.wikipedia.org/wiki/National_archives
[2] https://en.wikipedia.org/wiki/Records_management
[3]