Yash Jain MDS202048
Yash Raj MDS202049

DCBD Assignment 1

April 2021

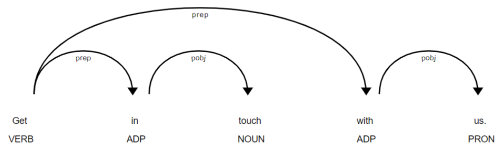
What is the best savings you could achieve on the given input?

On the given input files we were able to achieve a total space gained score of **98.35%**. We were able to achieve a higher score of **98.70%** that worked well for almost 90% of the web pages but we chose to go with the code with lower score and higher efficiency.

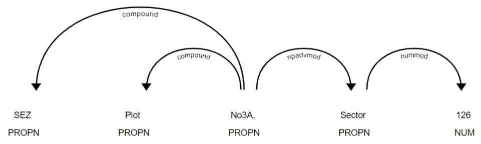
What data processing steps did you perform?

1. **Removing unnecessary HTML tags:**
   We carefully examined the source code of various web pages and created a set of HTML tags that we knew for sure would never contain an address. Some strong assumptions were made like, we hardly come across a website with the main heading (H1) as an address. The **<head>** could also be ruled out. Having decomposed these tags, we were in a zone where we could do some text analysis and rule out other larger chunks of unnecessary data.

2. **Removing strings with Verbs and Adjectives as ROOT of the string:**
   We resorted to using **Spacy**, a Natural Language Processing library, for this task. It was easier to remove strings containing verbs and adjectives using **removepos( )**. Since, we don't see adjectives and verbs in Indian addresses, it made sense to remove such strings which reduced the text considerably.



Here we can see that 'GET' is a verb and is also the ROOT of the string



ROOT of address strings can't be verbs or adjectives

3. **Removing sentences containing the stop words**:
   We created a list of our own stop words that are hardly ever found in addresses but are common in

English sentences like **has**, **or**, **is**. These were case sensitive. We realised this wasn't enough, so we used **NLTK's** list of stop words with length>3 and merged it with our own list. Then we removed those sentences that contained these words.

4. **Optional Step:**
   The idea was to remove words of len>3 which are common in English by setting a threshold frequency using **zipf_frequency( )** and ignoring numbers. This resulted in upto a 5% increase in the space saving score. It gave great results on almost 90% of the web pages preserving all the addresses but in some cases resulted in removal of certain parts of the address. This was based on the assumption that the majority of Indian addresses do not typically have common English words. So, it did work on the majority of addresses.

5. **Removing labels such as time, product, event, work_of_art etc:**
   Finding sentences which have words referring to **'EVENT', 'WORK_OF_ART', 'TIME', 'LAW', 'LANGUAGE', 'PERCENT', 'MONEY', 'QUANTITY', 'ORDINAL'.** Then deleting those sentences. We did not consider **'DATE'** because some of the pin codes can refer to **'DATE'.** (eg. 202011 can refer to NOV-2020)

```
d = nlp("70% of students from Aukamm Elementary School, London took part in English
writing held on 25-01-2021 at 2:00 PM" )
for e in d.ents:
   print(e,":",e.label_)
```

```
70% : PERCENT
Aukamm Elementary School : ORG
London : GPE
English : LANGUAGE
25-01-2021 : DATE
2:00 PM : TIME
```

6. **Removing sentences with unwanted names:**
   Indian Addresses may contain names (like Rajiv Gandhi Road). We can lose some part of addresses if we remove all the names. So, we removed only those sentences containing names with a title (**Mr., Ms., Mrs.,** etc.).

If provided more time, what more could you have done to improve your savings score?

Approach 1

We could have filtered out more content by identifying more HTML tags that weren't important to us. We could have made it more adaptive w.r.t. different websites based on the structure of a web page and group websites following a certain kind of template design.

Approach 2
Suppose we had some data containing only addresses. We can train a model which can detect 'addresses' from any given text.
Brief:
Using SPACY, we can find the structure of every sentence in the train set (addresses). Using these structures can train a model to detect all similar structures in the inputs.

How easy/difficult was this task? What challenges did you come across?

The Difficult Part

The most obvious roadblock in our initial approach was the number of formats in which addresses could be written. For any assumption, we could find a possible address that would fail our assumption. It was hard to make strong assumptions at the beginning..

We also struggled a bit to generalise our approach as much as possible. Websites come in all shapes and sizes and what worked for a particular website ended up giving opposite results for the other websites. We debated over choosing the code which gives better score but doesn't generalise well or choose the code that gives slightly lower score but generalises well across different websites.

The Easy Part

We realised pretty early that rather than spending time on identifying addresses, we should focus more on identifying the non-addresses. Knowing this made the task easier for us. Getting to a 95+ score was quite easy but we spent a significant amount of time getting to a 97.5+ score.

The freedom to use libraries also made it much simpler and helped us in achieving better results. Also, being a fun and interesting assignment, it felt less like a burden which in turn made it easy to work on and we sure learned a lot from it.

Note: Please open README.txt and install all the libraries before running the code.