CS 445 Natural Language Processing

Project 1: Lexicon and Rule-based Named Entity Recognition

Due Date: November 15, 23:55

Yağmur Duman 25133

In this assignment, we were expected to develop a lexicon and rule-based named entity recognition system (NER) for Turkish that identifies the followings over a given Turkish text:

- Person
- Location
- Organization
- Date and Time

To achieve our goal we were supposed to create Lexicons for Person, Location and Organization data as well as writing regular expression rules. We were only expected to write regular expressions to identify Date and Time.

Lexicons:

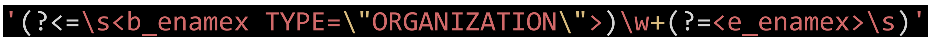
I resorted to different ways while creating my Lexicons, at first I tried doing web-scraping but I decided that it would be easier to collect data from sources on the Internet. I also scraped the NER_example_data which was given to us and I included to code file I used for that in my submission.

Although, I faced many difficulties while trying to collect data as it's hard to access exactly what you want and if I was just starting this project now, I would use web-scraping. I collected most of my location data for example, in English and I tried to translate them using Google Translate API, however after seeing some ridiculous outcomes I gave that up and tried to collect as much data in Turkish as I can. I also lost some Turkish characters in translation hence why in my **ner.py** before creating lists I'm eliminating words that have problems.

Here are the main sources I used:

- https://gist.github.com/ismailbaskin/1325813
- https://simplemaps.com/data/world-cities
- https://mii.com.tr/blog/turkiye-il-ve-ilceler-listesi-indir/
- Fortune 500 Türkiye Listesi
- Fortune 1000 Companies List

While scraping the NER_example_data I also used regular expressions to get specifically the words between the name tags, although, I could only go up to 6 words and that was the maximum amount of words inside the tags. Below is the example regex I used but it works for only one word inputs, I added together 6 like this.



```
'(?<=\s<b_enamex TYPE=\"ORGANIZATION\">)\w+(?=<e_enamex>\s)'
```

Rules:

As for the rules, I wrote many regular expressions, some are similar to each other while some are different. However, I based my regular expressions mostly on assumptions such as no person has more than 4 names or organization names must not exceed 6 words. To give examples from each regular expression:

```
'[A-ZÇĞİÖŞÜ][a-zçğıöşü]*\s[A-ZÇĞİÖŞÜa-zçğıöşü]*\s[A-ZÇĞİÖŞÜ][a-zçğıöşü]*|[A-
ZÇĞİÖŞÜ][a-zçğıöşü]*\s[A-ZÇĞİÖŞÜ][a-zçğıöşü]*|[A-ZÇĞİÖŞÜ][a-zçğıöşü]*'
```

- ➢ Above regular expression takes words of length 1, 2 and 3, this regex was written for organizations and the assumption that all organizations need to start and end with an uppercase letter was made, however words in between can vary so that organizations such as ‘London School of Economics’ can be captured. This assumption was also made for Person and Location names, however with them I also assumed that all words consisting of those names must start with an uppercase letter. I used regexes similar to this for checking Organization, Location and Person names and looking up if they exist in my Lexicons, if they did I would add them to a list to keep track of which entities were found. I’m using lots of ‘or’ operations for this.

```
'[A-ZÇĞİÖŞÜ][a-zçğıöşü]* Bey*'
```

- ➢ I used this format in most of my regular expressions that wanted to capture words that occurred before a certain word, although there were some changes in all of them of course such as what follows that certain word and how is the format of the words before that certain word formed.

```
'(?<=Bakan )[A-ZÇĞİÖŞÜ][a-zçğıöşü]*\s[A-ZÇĞİÖŞÜ][a-zçğıöşü]*'
```

- ➢ I used this format in most of my regular expressions that wanted to capture words that occurred after a certain word, although there were some changes in all of them such as how is the format of the words after that certain word formed. I’m using positive lookbehind for this.

As for writing rules for TIME I created two lists: months and days list to check if a certain word is month or day.

```
\d{4}-\d{2}-\d{2}|\d{2}-\d{2}-\d{4}
\d{4}\d{2}\d{2}|\d{2}\d{2}\d{4}|\d{4}\/\d{2}\/\d{2}|\d{2}\/\d{2}\/\d{4}|\d{2}(
\\)\d{2}(\\)\d{4}|\d{4}(\\)\d{2}(\\)\d{2}|\d{2}.\d{2}.\d{4}|\d{4}.\d{2}.\d{2}
```

- ➢ For capturing dates like 02.10.1999 I used the above format and I used a lot of ‘or’ operations and digit operators for this. I believe I’m capturing almost every date style that consists of (- , . , \ , /).

```
\d{2}\s[A-ZÇĞIÖŞÜa-zçğıöşü]*\s\d{4}|\d{2}\s[A-ZÇĞIÖŞÜa-zçğıöşü]*
```

- ➢ For dates like: 12 Mayıs 1998, 12 mayıs 1998 ve 12 mayıs, 12 Mayıs

```
'\d{4}\'[A-ZÇĞIÖŞÜa-zçğıöşü]*'
```

- ➢ For dates like: 1999’da

In the below regular expression positive look ahead is used as a change to capture words that come before a certain word:

```
'\d{4}(?=\s+yıl\w+)'
```

- For dates like, 1999 yılında

```
'\d{4}(?=\s+sene\w+)'
```

- For dates like, 1999 senesinde

```
[A-ZÇĞIÖŞÜa-zçğıöşü]*(?=\s+ay[a-zçğıöşü])
```

- For dates like, Mart ayında

```
[A-ZÇĞIÖŞÜa-zçğıöşü]*(?=\s+gün\w+)
```

- For dates like, pazartesi günü

```
(?<=saat )\d{2}\:\d{2}|(?<=saat )\d{2}\.\d{2}|(?<=saat )\d{2}|(?<=saat )\d{1}
```

- For dates like, saat 12, saat 12:00, saat 12.00, here I’m using positive look behind

Some notes and Outcomes:

My code doesn’t work for every single case and as I was doing this project, I’ve noticed some weird outcomes that challenged me a lot so I would like to talk a little bit about them. In cases such as where a person’s name is also a street’s name with just using regular expressions the difference can’t be understood by the program and the street name will be outputted as both PERSON and LOCATION. I couldn’t handle this problem just by using regular expressions so I wrote some if statements to prevent cases like this (If ‘Sokak’ exists in the word group don’t take it as a Person) however I still don’t think this is the most efficient and reasonable way to handle this. Another case was when an organization name was also a person name such as ‘Renaud’, here the only way to understand the difference between ORGANIZATION and PERSON was to look at the context of the sentence which I believe is impossible using regular expressions and even if statements couldn’t fix this issue. Also while writing statements such as if ‘ilçe’ is seen then return the word group before it, failed in cases where the sentence was for example like: ‘Uğradığı ilçe..’ this could be checked from the lexicons but then the ‘ilçe’ statement wouldn’t be useful. However, for dates and times regular expressions will probably work very well in most cases. These were just a few of my notes I gathered while I was doing the project, thanks for reading.