COREFERENCING

Anaphora and Cataphora Co-Referencing

Suprita Ganesh 1490723

BASICS

CO-REFERENCING

When more parts of the text refer to the same entity, the multiple occurrences are said to Co-reference each other

Anaphora

When pronouns/noun-phrases follow their antecedent (nouns/noun-phrases)

E.g. **Mathew** is a brilliant student. **He** always comes first in class -----(*Here ''He' trails 'Mathew'*)

Cataphora

When pronouns/noun-phrases lead their antecedent (nouns/noun-phrases)

E.g. If **she** does not study, **Gloria** will fail her tests ------(*Here 'she' leads 'Gloria'*)

Algorithm

Algorithm:

1.POS-TAGGING

2. Forming Chunks OF PRONOUNS , NOUN PHRASES, NOUNS using Names entity recognition and Regex

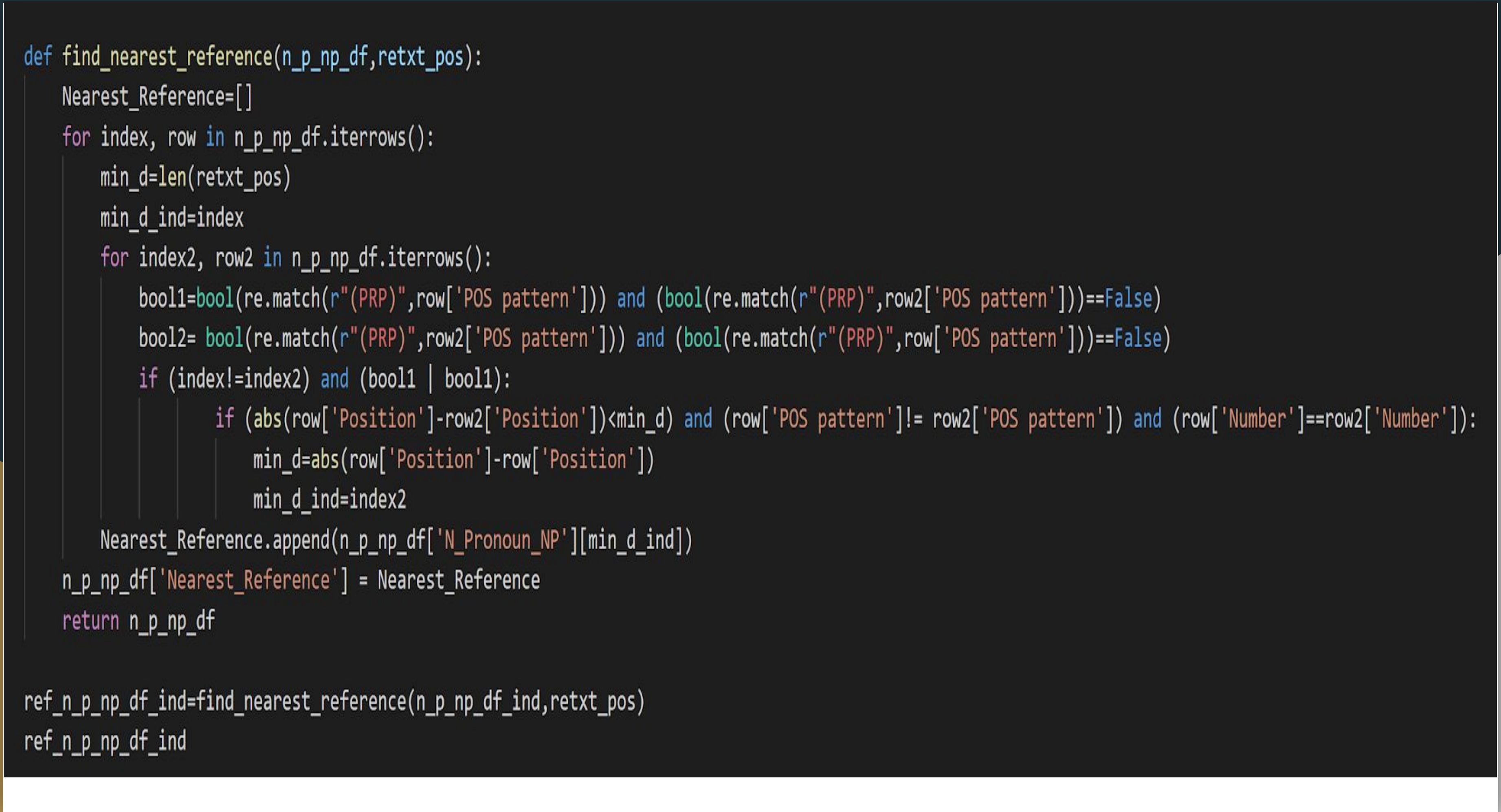
3.. Attaching Appropriate NOUN, NOUN-PHRASE TO PRONOUN first by Grammer of Singularity/Plurality and word distance

REGEX EXPRESSIONS:

ANAPHORA CHUNKS= "anaphora: {<DT>?<PRP.>?<JJ.?>*<NNP?S?>+<.*>*<PRP>+}"

CATAPHORA CHUNKS= "cataphora: {<PRP.?>+<.*>*<DT>?<PRP.>?<NNP?S?>+}"

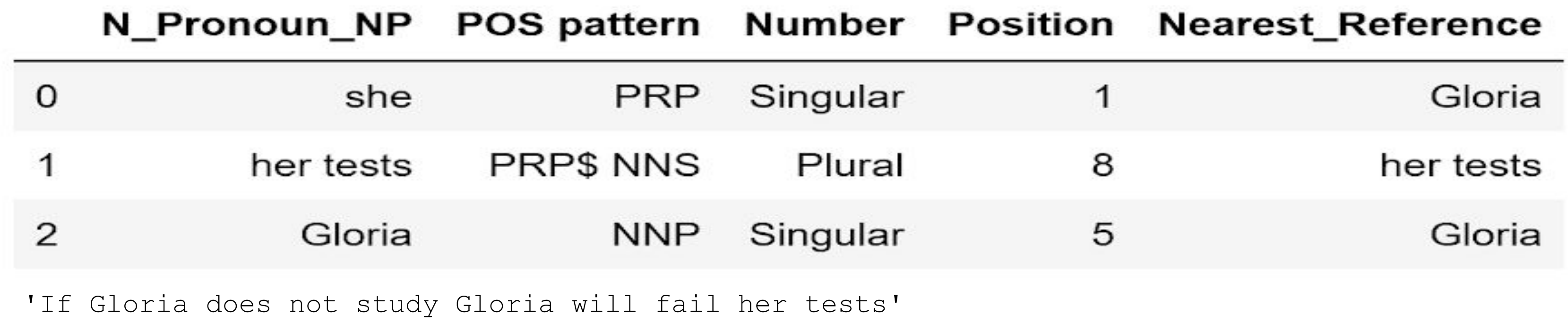
PRONOUN CHUNKING {<PRP>+} NOUN PHRASE= "NP: {<PRP.>?<DT>?<JJ.?>*<NNS?>+}"



```
def find_nearest_reference(n_p_np_df,retxt_pos):
    Nearest_Reference=[]
    for index, row in n_p_np_df.iterrows():
        min_d=len(retxt_pos)
        min_d_ind=index
        for index2, row2 in n_p_np_df.iterrows():
            bool1=bool(re.match(r"(PRP)",row['POS pattern'])) and (bool(re.match(r"(PRP)",row2['POS pattern']))==False)
            bool2= bool(re.match(r"(PRP)",row2['POS pattern'])) and (bool(re.match(r"(PRP)",row['POS pattern']))==False)
            if (index!=index2) and (bool1 | bool1):
                    if (abs(row['Position']-row2['Position'])<min_d) and (row['POS pattern']!= row2['POS pattern']) and (row['Number']==row2['Number']):
                        min_d=abs(row['Position']-row['Position'])
                        min_d_ind=index2
        Nearest_Reference.append(n_p_np_df['N_Pronoun_NP'][min_d_ind])
    n_p_np_df['Nearest_Reference'] = Nearest_Reference
    return n_p_np_df


ref_n_p_np_df_ind=find_nearest_reference(n_p_np_df_ind,retxt_pos)
ref_n_p_np_df_ind
```

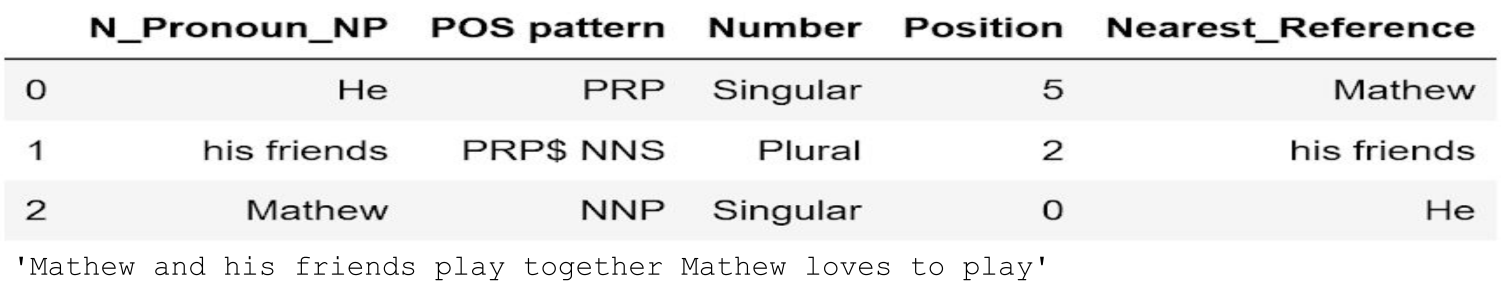
C:If she does not study, Gloria will fail her tests



	N_Pronoun_NP	POS pattern	Number	Position	Nearest_Reference
0	she	PRP	Singular	1	Gloria
1	her tests	PRP$ NNS	Plural	8	her tests
2	Gloria	NNP	Singular	5	Gloria

```
'If Gloria does not study Gloria will fail her tests'
```

A: Mathew and his friends play together. He loves to play.



	N_Pronoun_NP	POS pattern	Number	Position	Nearest_Reference
0	He	PRP	Singular	5	Mathew
1	his friends	PRP$ NNS	Plural	2	his friends
2	Mathew	NNP	Singular	0	He

```
'Mathew and his friends play together Mathew loves to play'
```

CHALLENGES

1. Only refers on basis of singular/ plural and closeness to the noun/pronoun/ noun-phrase
2. Reference based on Gender , between nouns it is easy to reference to non associated noun
3. When to assign noun-phrase to proper noun, when to let it be
    a. **The hardworking boy** and his friend study together and have **fun**. **He** is class topper
    b. **The tall girl** is very friendly. Her name is **Wanda**
4. In situations where there are multiple nouns but the right proper noun leads/trails the pronoun by a lot of indices it is easy to lose the reference
5. More complex regex and grammar rules have to be exercised for better resolution
6. Advanced Algorithms must be applied to do coreferencing, a better option is explore applying Neural Networks on labelled text with appropriate references

ELASTIC SEARCH

1. Elasticsearch, an open-source analytics and search engine
2. Elasticsearch is a distributed, RESTful search and analytics engine that centrally stores your data so you can search, index, and analyze data of all shapes and sizes.
3. Logstash provides a convenient way to use the bulk API to upload data in Elasticsearch
4. It can be visualised on Kibana
5. It allows Facetting
6. It allows you to store, search, and analyze big volumes of data quickly and in near real time