Reading Elective Interim Report

-K.Sai Krishna

The main aim of the project is to anonymize sensitive data like names of the person, credit card number etc… So, to do that first we need to do NER(Named Entity Recognition) and then try to use some algorithm to replace our entity of interest with some other entity so that the output text still makes sense.

Coreference resolution

Coreference resolution is the task of clustering the expressions in a sentence which refer to the same real-world entities.

For example, consider the sentence ***"I voted for modi because he was aligned with my values", Harsha said.*** In this sentence 'I' and 'Harsha' refer to the same real-world entity, similarly 'modi' and 'he' refer to the same real-world entity. So, when we do coreference resolution, the clusters generated will be ['I', 'Harsha'], ['modi', 'he'] and now using these clusters we replace the pronouns in the sentence with corresponding nouns which we got from clusters.

In the below tables, I showed two examples of coreference resolution generated using the AllenNLP library in python.

Outputs using Allen NLP's Coreference Resolution model.

Sentence	`"David went to the concert. He said it was an amazing experience."`
Clusters	`[['David', 'He']]`
Modified sentence	`”David went to the concert. David said it was an amazing experience.”`

Sentence	`"Barack Obama nominated Hillary Rodham Clinton as his secretary of state on Monday. He chose her because she had foreign affairs experience as a former First Lady."`
Clusters	`[['Barack Obama', 'his', 'He'], ['Hillary Rodham Clinton', 'her', 'she']]`

Modified sentence	”Barack Obama nominated Hillary Rodham Clinton as Barack Obama's secretary of state on Monday. Barack Obama chose Hillary Rodham Clinton because Hillary Rodham Clinton had foreign affairs experience as a former First Lady.”

Code details:

- Used Allen NLP’s coreference resolution model.
- Consider the sentence ***"Barack Obama nominated Hillary Rodham Clinton as his secretary of state on Monday. He chose her because she had foreign affairs experience as a former First Lady."***
- When I used the model for the above sentence, it emits the clusters using the indices of words in it as shown below.
    - `[[[0, 1], [7, 7], [14, 14]], [[3, 5], [16, 16], [18, 18]]]`
- So, to get better visualization of words in the clusters, I wrote a small code for it and after that we can visualize the exact clusters as shown below.
    - `[['Barack Obama', 'his', 'He'], ['Hillary Rodham Clinton', 'her', 'she']]`

<u>Code snippet of above mentioned logic</u>

```
clus_all = [] # To store all clusters
cluster = [] # To store all texts that belong to the same real-world entity.
temp = ""
for i in range(0, len(clusters)):
 one_cl = clusters[i]
 for count in range(0, len(one_cl)):
   obj = one_cl[count]
   for s in range((obj[0]), (obj[1]+1)):
     temp = temp + " " + doc[s]
   cluster.append(temp[1:])
   temp=""

 clus_all.append(cluster)
 cluster = []
print (clus_all) #And finally, this shows all coreferences
```

Named Entity Recognition(NER)

NER is a standard NLP problem which deals with information extraction. The objective of NER is to locate and classify different entities in the input text into predefined categories like name, organization and location. To generate named entities for an input text, I used Allen NLP's pre-trained NER model.

1) Allen NLP's pre-trained model for NER uses the CoNLL-2003 NER dataset.

2) There are four types of named entities in the CoNLL-2003 NER dataset. They are;
    - Persons
    - Locations
    - Organizations
    - Names of miscellaneous entities

3) Different types of tagging schemes present are;
    - IOB
    - IOB2
    - Start/End
    - BIOES/BILOU

4) In this link it is mentioned that the CoNLL-2003 NER dataset uses the IOB2 tagging scheme which is as follows;(or chunk)

    - **O -** 0
    - **B-PER -** 1
    - **I-PER -** 2
    - **B-ORG -** 3
    - **I-ORG -** 4
    - **B-LOC -** 5
    - **I-LOC -** 6
    - **B-MISC -**7
    - **I-MISC -** 8

5) But when I ran the model, I also got tags with prefixes 'U-', 'L-' where U stands for *Unit* and L stands for *Last.* So, I think the tagging scheme should be ***BILOU***

6) Note that whenever I use the word *token* it means that it is a single word generated by tokenizer.

7) Each letter in tagging scheme are used as prefixes for each token and are abbreviated as follows;
    - B - Beginning token of the entity(or chunk).
    - I - Inside token of the entity(or chunk).
    - L- Last token of the entity(or chunk).
    - O - Outside of all the entities(or chunks).
    - U - Unit entity(i.e; the entity contains only one token).

In the below tables, I showed two examples of NER generated using the AllenNLP library in python. Note that I gave the drive link in example-2 which is run for a text file input instead of hardcoding the sentence in the code.

Outputs using Allen NLP's pre-trained NER model.

Sentence	`"The ISIS has claimed responsibility for a suicide bomb blast in the Tunisian capital earlier this week. A militant wearing an explosive belt blew himself."`
NER(IOB2 tagging)	`[('The', 'O'), ('ISIS', 'U-ORG'), ('has', 'O'), ('claimed', 'O'), ('responsibility', 'O'), ('for', 'O'), ('a', 'O'), ('suicide', 'O'), ('bomb', 'O'), ('blast', 'O'), ('in', 'O'), ('the', 'O'), ('Tunisian', 'U-MISC'), ('capital', 'O'), ('earlier', 'O'), ('this', 'O'), ('week', 'O'), ('.', 'O'), ('A', 'O'), ('militant', 'O'), ('wearing', 'O'), ('an', 'O'), ('explosive', 'O'), ('belt', 'O'), ('blew', 'O'), ('himself', 'O'), ('.', 'O')]`
Combining clusters	`[('The', 'O'), ('ISIS', 'ORG'), ('has claimed responsibility for a suicide bomb blast in the', 'O'), ('Tunisian', 'MISC'),('capital earlier this week . A militant wearing an explosive belt blew himself .', 'O')]`

Sentence, NER tags	https://drive.google.com/drive/folders/1z6BgtloOlg24XPmFB3eV8CoH2XtVvnwd
Example taken from this website	https://www.holisticseo.digital/theoretical-seo/named-entity-recognition/

Code details:

- Used Allen NLP's NER model.
- It doesn't identify the whole entity and give it the corresponding tag like PER, LOC, ORG etc… It uses prefixes ['B-', 'I-', 'L-', 'U-'] before the main ner tags and gives them to each word we got with the tokenizer.
- So, to get rid of all those prefixes and give the ner tags to the whole entity, I wrote a piece of code. For example,
    - **Sentence:** *"George Washington went to Washington"*
    - **Before applying that code:** `[('George', 'B-PER'), ('Washington', 'L-PER'), ('went', 'O'), ('to', 'O'), ('Washington', 'U-LOC')]`

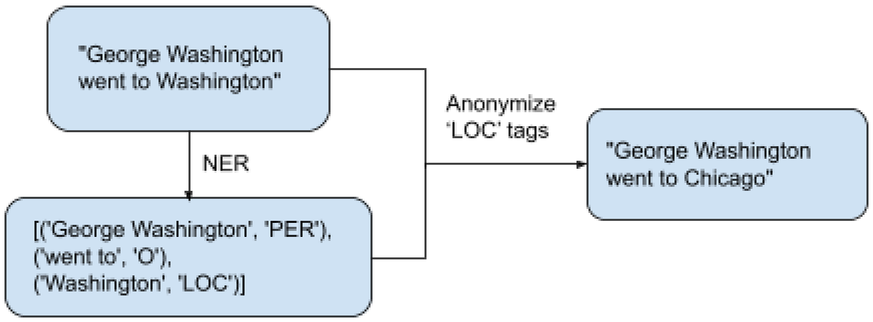
- **After applying that code:** `[('George Washington', 'PER'), ('went to', 'O'), ('Washington', 'LOC')]`

<u>Code snippet of above mentioned logic</u>

```
final_lst = [] # This is list of tuples where each tuple contains entity and its NER tag
i = 0
n = len(lst) # This list contains tuples where each tuple contains a token and its tag according to the tagging scheme.
s = ""
prev_tag = ""    # Stores tag(like PER, LOC, ORG, MISC, O) of the previous token in the sentence.
while i < n:
  ps = lst[i][0] # Stores the present token.
  pt = lst[i][1] # Stores the tag of present token.

  if pt[0] == 'B':
    if s != "":
      tpl = (s, prev_tag)
      final_lst.append(tpl)
    s = ps
    prev_tag = pt[2:]

  elif pt[0] == 'I':
    s = s + " " + ps
    prev_tag = pt[2:]

  elif pt[0] == 'L':
    s = s + " " + ps
    tpl = (s, pt[2:])
    final_lst.append(tpl)
    s=""

  elif pt[0] == 'U':
    if s!="":
      tpl = (s, prev_tag)
      final_lst.append(tpl)
    tpl1 = (ps, pt[2:])
    final_lst.append(tpl1)
    s=""
    prev_tag=""

  elif pt == "O":
    if(prev_tag == "O"):
      if ps == '.' or ps == ',':
        s = s+ps
      else:
        s = s + " " + ps
    else:
      if s!="":
        tpl = (s, prev_tag)
        final_lst.append(tpl)
      s = ps
    prev_tag = pt

  i+=1

if s!="":
  tpl = (s, prev_tag)
  final_lst.append(tpl)
```

Future work:

Now I will use the data obtained from AllenNLP's NER and the Word2Vec model to try and anonymize a particular named entity tag(say LOC). I thought of changing the input sentence in such a way that each entity can be considered as one token when done tokenization so that embedding for that word will be generated instead of individual words in the entity. Then I will pass it to the Word2Vec model and find the similar words of the interested words(i.e., LOC entities) and replace the interested word with the most similar word. To avoid clashes, I will maintain a look-up table.


"George Washington went to Washington"
NER
[('George Washington', 'PER'), ('went to', 'O'), ('Washington', 'LOC')]
Anonymize 'LOC' tags
"George Washington went to Chicago"

References:

About	Link
Pre-trained dataset used by Allen NLP for NER	https://huggingface.co/datasets/conll2003#dataset-structure
Allen NLP Demo	https://demo.allennlp.org/named-entity-recognition/named-entity-recognition
Shaip(One good format which can be used for representing the NER output)	https://www.shaip.com/solutions/named-entity-recognition-ner/?gclid=Cj0KCQjw7KqZBhCBARIsAI-fTKJ7sa4Ea34Pg3AOxvyz7vtFOljViYO1EOOokeWjcmwvh15SPnndGXEaAk2DEALw_wcB
Different types of Tagging Schemes	https://en.wikipedia.org/wiki/Inside%E2%80%93outside%E2%80%93beginning_(tagging)
Collab Notebook Link	**https://colab.research.google.com/drive/17m5sjl-SwL83Eekee1itftSglFAU3WWY#scrollTo=_CXDZTWOjx2L**