Page No.
Date

Aim- To implement Named Entity Recognizer for text input.

Theory-
Named entity Recognizer also called as NER is a subtask of information extraction that seeks to locate & clarify named entity mentioned in unstructed text into pre-defined categories such as person, names, organisations, location, medical, codes, time expression, quantities monetary values, percentages etc

For eg (US, 'GPE') (15000, 'CARDINAL')
(China, 'GPE'), (First, 'ORDINAL)
(Google, 'Org'), (Zuckerberg, 'PERSON')

There are in all 15 Named Entity Recognition tags
NER systems have created that use linguistic grammer based techniques as well as statistical models such as machine learning. Hand crafted grammar based system typically obtain better precision, but at the cost of lower recall & months of work by experienced computational linguistic
The sentence should be annotated into block of text thus we can highlight the names of the entities
eg Jim bought 300 shares of Acme Corp in 2006

Teacher's Sign.:

Page No.
Date

Here, (Jim, Person), 300 cordinal, (Acme corp, organisation), (2006, Time).

Applications of NER :
1) Classifying content for providers
News houses generate large amount of online content on a daily basis & managing them correctly is very important to ~~daily basis &~~ get most out of articles. NER scans entire articles & reveal which are the major people, org and places discussed in them.
2) Powering content Recommendation:
NER is used to recommend similar articles or shows or movies or products to users. This will automate the recommendation process
3) Customer Support:
NER is used to extract entities from the complaints and thus can be helpful in organizing the complaints and assign them to a relevant department within the organization that is capable of handling this

Conclusion - Thus we have successfully implemented NER to extract names of entities from the given texts.

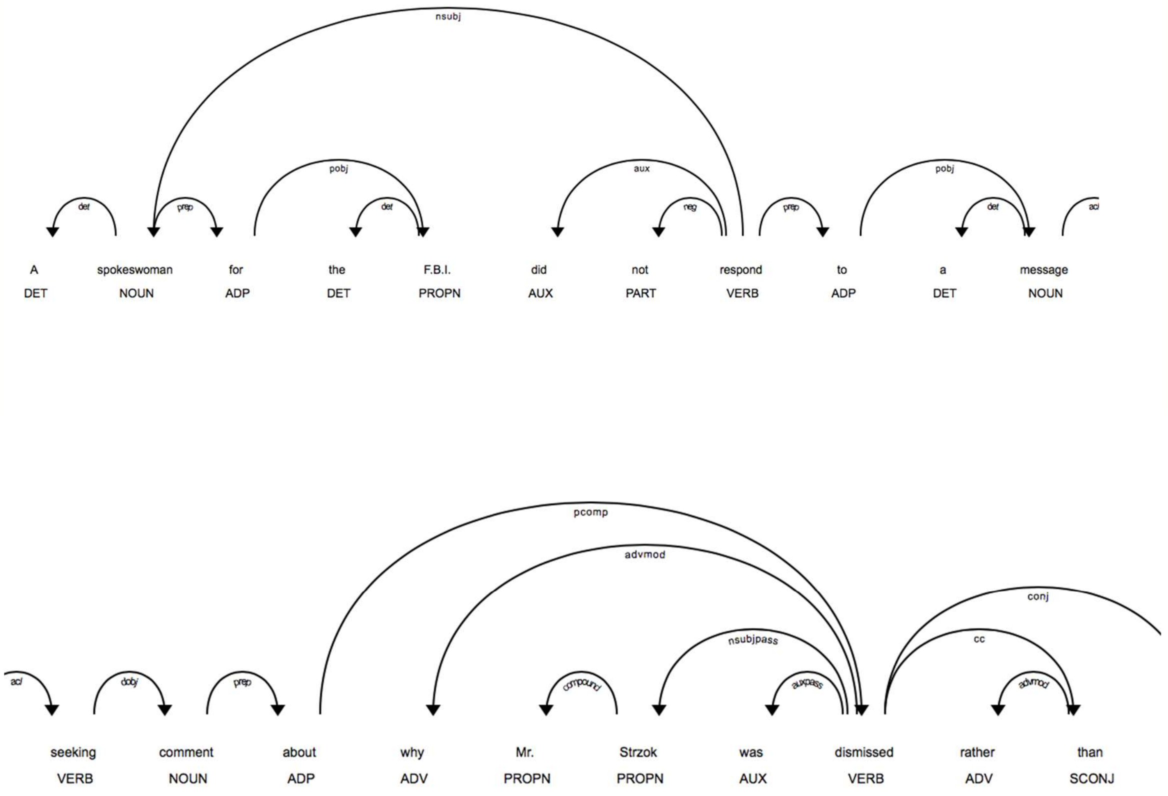
Teacher's Sign.:

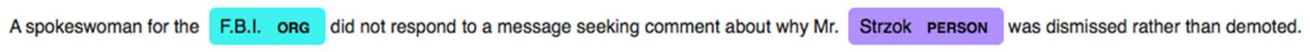
Code :

```
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag
from nltk.chunk import ne_chunk
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
sentence = 'Air India pilots write to CMD of Air India: By effecting a cut only on allowances, directors
and management executives have deviously exempted themselves from any meaningful austerity cut as
their allowances are extremely small. The pay cut on allowance is unequal and not acceptable to us'
ne_tree = ne_chunk(pos_tag(word_tokenize(sentence)))
def preprocess(sent):
  sent = nltk.word_tokenize(sent)
  sent = nltk.pos_tag(sent)
  return sent
ex = 'Air India pilots write to CMD of Air India: By effecting a cut only on allowances, directors and
management executives have deviously exempted themselves from any meaningful austerity cut as their
allowances are extremely small. The pay cut on allowance is unequal and not acceptable to us'
sent = preprocess(ex)
pattern = 'NP: {<DT>?<JJ>*<NN>}'
cp = nltk.RegexpParser(pattern)
cs = cp.parse(sent)
iob_tagged = tree2conlltags(cs)
pprint(iob_tagged)
doc = nlp('The United States is doing it bottom-up. Each state is beginning a lockdown process and it is
adding up to the United States. Take Sweden for example. Sweden has made a compact with its citizens
and asking citizens to actually honor the system and do a lockdown themself. India is a country that has
done lockdown across the country; 1.3 billion, unprecedented and unheard of. ')
pprint([(X.text, X.label_) for X in doc.ents])
pprint([(X, X.ent_iob_, X.ent_type_) for X in doc])
def url_to_string(url):
  res = requests.get(url)
  html = res.text
  soup = BeautifulSoup(html, 'html5lib')
  for script in soup(["script", "style", 'aside']):
     script.extract()
  return " ".join(re.split(r'[\n\t]+', soup.get_text()))
ny_bb = url_to_string('https://www.nytimes.com/2018/08/13/us/politics/peter-strzok-fired-
fbi.html?hp&action=click&pgtype=Homepage&clickSource=story-heading&module=first-column-
region&region=top-news&WT.nav=top-news')
article = nlp(ny_bb)
len(article.ents)
labels = [x.label_ for x in article.ents]
Counter(labels)
items = [x.text for x in article.ents]
Counter(items).most_common(3)
```

```
sentences = [x for x in article.sents]
print(sentences[20])
displacy.render(nlp(str(sentences[20])), jupyter=True, style='ent')
[(x.orth_,x.pos_, x.lemma_) for x in [y
                                      for y
                                      in nlp(str(sentences[20]))
                                      if not y.is_stop and y.pos_ != 'PUNCT']]
```

OUTPUTS :





A spokeswoman for the F.B.I. ORG did not respond to a message seeking comment about why Mr. Strzok PERSON was dismissed rather than demoted.

```
Counter({'CARDINAL': 38,
         'DATE': 24,
         'FAC': 7,
         'GPE': 36,
         'MONEY': 1,
         'NORP': 6,
         'ORG': 78,
         'PERSON': 12,
         'PRODUCT': 6,
         'TIME': 4,
         'WORK_OF_ART': 1})
```

```
[('The United States', 'GPE'),
 ('the United States', 'GPE'),
 ('Sweden', 'GPE'),
 ('Sweden', 'GPE'),
 ('India', 'GPE'),
 ('1.3 billion', 'CARDINAL')]
```