```
from allennlp.data.tokenizers.word_tokenizer import WordTokenizer
from allennlp.predictors.predictor import Predictor

coref_predictor = Predictor.from_path("C:\Users\Downloads\
                                       coref-model-2018.05.tar.gz")

def coref_resol_allennlp(article):
    wt = WordTokenizer()
    art_words = wt.tokenize(article)
    coref_article = dict()
    res = coref_predictor.predict(article)
    for cluster in res['clusters']:
        key_start = cluster[0][0]
        key_end = cluster[0][1]+1
        coref_key = ' '.join([str(w) for w in
                              art_words[key_start:
                                        key_end]])
        coref_article[coref_key] = []
        for words in cluster:
            coref_article[coref_key].append(words[0])

    return coref_article.
```

In main:

```
coref_art = coref_resol_allennlp(article)
```

↳ no need to check coref_art separately the return value is coref_al

+

Relevance model embedded with Allen NLP coref model

+ Syntax Index of coref resol.

allennlpner.py

```
from allennlp.predictors.predictor import Predictor
predictor = Predictor.from_path(r"C:\Users\<redacted>\Downloads\ner-model-2018.12.18-bx.tar.gz")
```

```
import pandas as pd
df = pd.read_csv(r"____\\reviews.csv", encoding="latin1")
mylist = list(df['Body'])
final = list()
for text in mylist:
    result = predictor.predict(text)
    for i in range(len(result['tags'])):
        if(result['tags'][i] != 'O'):
            final.append((result['words'][i],
                          result['tags'][i]))
df = pd.DataFrame(final)
```

```
actual = list()
i = 0
for i in range(len(final)):
    print(i)
    if("U-" in final[i][1]):
        actual.append(final[i])
        i += 1
    elif("B-" in final[i][1]):
        temp = final[i][0]
        i += 1
        while("L-" not in final[i][1]):
            temp += " " + final[i][0]
            if(i < 0):
                i += 1
        temp += " " + final[i][0]
        i += 1
        actual.append((temp, final[i][1][2:]))
```

```
else:
    i += 1
```

attempt.py

EDA

```
import sys
import numpy as np

directory = r'C:\Users\Intellipaat\'

f = open(directory + 'result.txt', 'a')

sys.stdout = f

analysis = ['dnaDF.describe()', 'dnaDF.shape',
            'dnaDF.columns', 'dnaDF.dtypes',
            'dnaDF.columns[dnaDF.dtypes!=np.object]',
            'dnaDF.info()', 'dnaDF.isnull().values.any()',
            'dnaDF.isnull().sum()',
            'dnaDF['body'][:100]',
            'dnaDF.columns[dnaDF.isna().any()]',
            'dnaDF.isnull.sum()']

for i in range(len(analysis)):
    print(analysis[i])
    print(eval(analysis[i]))
```

```
df.shape[0] - sum(df['label'].value_counts())
```

```
import tarfile
tf = tarfile.open("___.tar.gz")
tf.extractall("____")
```

Stanford NLP

```
import pandas as pd
from nltk.tag import StanfordNERTagger
from nltk.tokenize import word_tokenize
st = StanfordNERTagger(
'C:\\___\\stanford-ner-2018-10-16\\___\\classifiers
\\english.all.3class.distsim.crf.
ser.gz'
'C:\\___\\___\\stanford-ner.jar',
encoding='utf-8')
df=pd.read_csv("___", encoding='latin1')
mylist=list(df['Body'])
```

```
mytext = myfile.read()
mytext = myfile()
tokenized_text = word_tokenize(mytext)
classified_text = st.tag(tokenized_text)
output = list()
for item in classified_text:
    if item[1] in tags.keys():
        tags[item[1]].append(item[0])
        output.append(item)
print(tags)
output_data = pd.DataFrame(output)
    )
to_csv
```

```
count_vectorizer vhy AllenNLP.
import pickle
from allennlp.predictors.predictor import
Predictor
```

```
from allennlp.data.tokenizers.sentence_splitter import
    SpacySentenceSplitter

import pandas as pd

predictor = Predictor.from_path("___/bidaf-model-
                                2018.12.18-tz.gz")

def get_entities(result):
    Same code as allennlp.py

    return actual.

load data in (mylist)
              ↳ list of articles

st = SpacySentenceSplitter()
sentences_list = st.batch_split_sentences(
                    mylist)

k=0
all_list = list()
for article in sentences_list:
    print('j:', j)
    k+=1
    if k==2:
        print(len(article))
```

```
for sentence in article:
    if (sentence != None):
        if (len(sentence)>2):
            try:
                pdvec = predictor.predict(sentence)
            except:
                print('except')
            print(j)
            j+=1
    test_ent = get_entities(pdvec)
    if (len(test_ent)>1):
        alt_ent.extend(list(itertools.combinations
                            (sorted(test_ent),2)))
```

And this alt_ent to csv.

then Count relations by code is used

```
with open('alt.csv', mode='w') as infile:
    reader = csv.reader(infile)
    for rows in reader:
        if (len(rows) != 0):
            rows = rows[1:]
            mylist.append(tuple(sorted(rows)))
```

mylist = list(set(mylist[1:]))
→ remove 1st row

sum = 100

for i in range(len(mylist)):

sum1 += (mylist.count(mylist[i]))

mylist[i] = mylist[i] + (mylist.count(mylist[i]),)

↓

Convert tuples

get row - count

mylist.insert(0, (mylist[0] + ("count:",)))

→ column name added

write to CSV

In location.py (updated)

thing to note

→ Arrange the articles by recent time

For the timestamp is convert to datetime

by

import datetime

dnoDF['date'] = pd.to_datetime(dnoDF['publication_date'], unit = ms).

SPARQL Query .

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
SELECT ?s ?property ?hasValue WHERE
{
    ?s ?property ?hasValue
}
SELECT DISTINCT ?s
WHERE
{
    ?s a dbo:Company
    ?s dbp:traded AS
        <http://dbpedia.org/resource/SEP-500>
}LIMIT 50
OFFSET
```

```
import json
import re
results = list()
text = ""
for i in range(1,6):
    with open(r"C:\Users\Installed\Project Files\kb" + str(i) + ".json",
              mode='r') as myfile:
        result.append(json.load(myfile))

KnowledgeBank = dict()
for r in result:
    for res in r['results']['bindings']:
        if( res['s']['type']=="uri" ):
            company = re.sub("http://dbpedia.org/resource/","",
                             res['s']['value'])
            company = re.sub('[0-9_#]',"",company)
            res['s']['value'] = company
        else:
            company = res["s"]["value"].
        KnowledgeBank[company] = dict().
```

```
for r in result:
    for res in r['results']['bindings']:
        if(res['property']['type']=="uri"):
            prop= re.sub(_____)
            prop= re.sub(_____)
        else:
            res['property']['value']=prop
            prop= res['property']['value']
        KnowledgeBase[res['s']['value']][prop]
                                    =set()
```

Same as for 's', replace 's' with 'property'

```
for y in result:
    for res in r['results']['bindings']:
        if(res['hasValue']['type']=="uri"):
            value = res['hasValue']['value']
        else:
            value = res['hasValue']['value']
        KnowledgeBase[res['s']['value']]
                    ['res['property']['value']].add(value)
```

Same as for prev if else

data = draft[draft['date']
> datetime.date(time(2018,11,1))]

Console command of Pandas

pd.set_option('display.max_columns', 100)
pd.set_option('display.max_rows', 100)
pd.set_option('display.width', 1000)