Md. Rahimuz Zaman
27771789

Comp 472
Mini-project 2

1. (1 pt) well documented code (Attrib. 1, 5)
2. Report including
   (a) (0.5 pts) SpaCy sentence and token splits for S1 (Attrib. 5)
   (b) (0.5 pts) a graphical representation of the two dependency graphs for S1 (Attrib. 5)
   (c) (5 pts) $T1_{S1}$ and $T2_{S1}$ for the snippet (Attrib. 1, 5)
   (d) (1 pt) 3-means clusters for T1 (for the entire text) (Attrib. 1, 5)
   (e) (1 pt) 2-means clusters for T2 (for the entire text) (Attrib 1, 5)
   (f) (1 pt) 1 page discussion of of results, observations, issues (Attrib. 6, 7)

A-

Sentence split

```
for sent in doc.sents: #for each sentences in doc
  print(sent,"\n")
```

```
U.S. intelligence agencies concluded in January 2017 that Russia mounted a
far-ranging influence campaign aimed at helping Trump beat Clinton.

And the bipartisan Senate Intelligence Committee, after three years of
investigation, affirmed those conclusions, saying intelligence officials had
specific information that Russia preferred Trump and that Russian President
Vladimir Putin had "approved and directed aspects" of the Kremlin's influence
campaign.
```

Word/token split

```
doc = nlp(text)
for sent in doc.sents: #for each sentences in doc
  for token in sent: #for each token/words in the sentences
    print(token,"---",token.tag_,"---",token.ent_type_, "---", token.dep_) #print
the word,tag, their entity type and dependency
```

Result:

```
U.S. --- NNP --- GPE --- compound
```

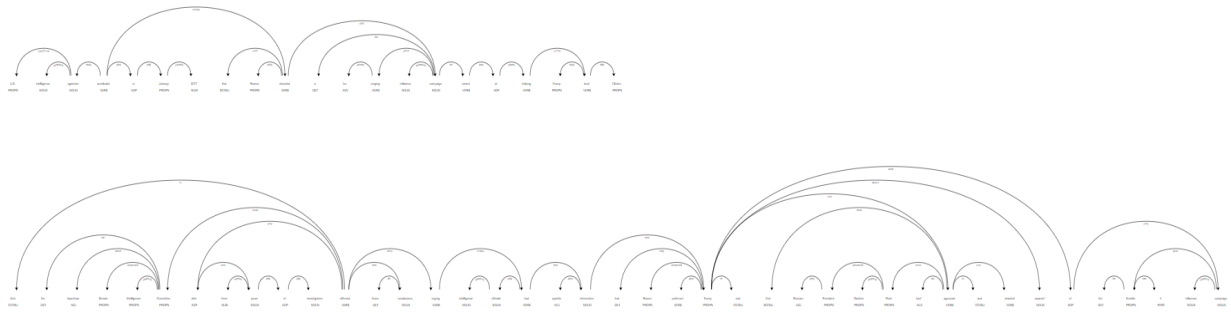
```
intelligence --- NN ---  --- compound
agencies --- NNS ---  --- nsubj
concluded --- VBD ---  --- ROOT
in --- IN ---  --- prep
January --- NNP --- DATE --- pobj
2017 --- CD --- DATE --- nummod
that --- IN ---  --- mark
Russia --- NNP --- GPE --- nsubj
mounted --- VBD ---  --- ccomp
a --- DT ---  --- det
far --- RB ---  --- advmod
- --- HYPH ---  --- punct
ranging --- VBG ---  --- amod
influence --- NN ---  --- compound
campaign --- NN ---  --- dobj
aimed --- VBN ---  --- acl
at --- IN ---  --- prep
helping --- VBG ---  --- pcomp
Trump --- NNP --- PERSON --- nsubj
beat --- VBN ---  --- ccomp
Clinton --- NNP --- PERSON --- dobj
. --- . ---  --- punct
And --- CC ---  --- cc
the --- DT ---  --- det
bipartisan --- JJ ---  --- amod
Senate --- NNP --- ORG --- compound
Intelligence --- NNP --- ORG --- compound
Committee --- NNP --- ORG --- nsubj
, --- , ---  --- punct
after --- IN ---  --- prep
three --- CD --- DATE --- nummod
years --- NNS --- DATE --- pobj
of --- IN ---  --- prep
investigation --- NN ---  --- pobj
, --- , ---  --- punct
affirmed --- VBD ---  --- ROOT
those --- DT ---  --- det
conclusions --- NNS ---  --- dobj
, --- , ---  --- punct
saying --- VBG ---  --- advcl
intelligence --- NN ---  --- compound
officials --- NNS ---  --- nsubj
had --- VBD ---  --- ccomp
```

```
specific --- JJ ---  --- amod
information --- NN ---  --- dobj
that --- WDT ---  --- dobj
Russia --- NNP --- GPE --- compound
preferred --- VBD ---  --- amod
Trump --- NNP --- PERSON --- relcl
and --- CC ---  --- cc
that --- IN ---  --- mark
Russian --- JJ --- NORP --- amod
President --- NNP ---  --- compound
Vladimir --- NNP --- PERSON --- compound
Putin --- NNP --- PERSON --- nsubj
had --- VBD ---  --- aux
“ --- `` ---  --- punct
approved --- VBN ---  --- conj
and --- CC ---  --- cc
directed --- VBN ---  --- conj
aspects --- NNS ---  --- appos
” --- '' ---  --- punct
of --- IN ---  --- prep
the --- DT ---  --- det
Kremlin --- NNP --- ORG --- poss
’s --- POS ---  --- case
influence --- NN ---  --- compound
campaign --- NN ---  --- pobj
. --- . ---  --- punct
```

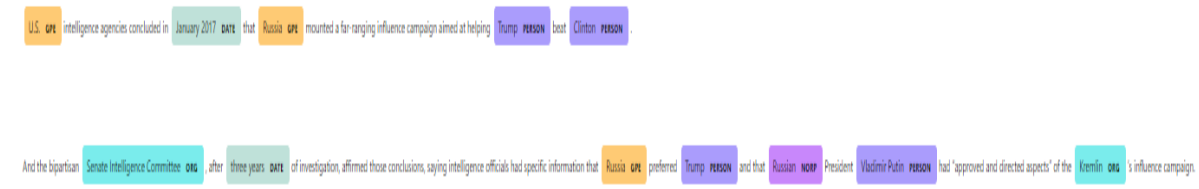
B-

```
# #for visualizing sentence dependency breakdown
sentence_spans = list(doc.sents)
displacy.serve(sentence_spans, style="dep")#for dependencies
displacy.serve(sentence_spans, style="ent")#for entities
```

Dependency graph



Entity graph



view the attached files for the full resolution

C-

```
#----------------Sentiment analysis/kmeans----------------------
af= Afinn()
ne=0
t=[]
tt=[]
def checkScore(score):
    if(score>0):
        return "positive"
    elif score==0:
        return "neutral"
    elif score<0:
        return "negative"

for sent in doc.sents:
    for token in sent:
        tokenArray=[]
```

```
        ttArray=[]
        sentence_score=af.score(sent.text)
        token_score=af.score(token.text)
        if(token.ent_type_):
            ne=1
        else:
            ne=0
        #print(f"token# {i}: {token.text} --> NE?: {ne} --> entity
type:{token.ent_type_} --> governor: {token.head} --> dependency: {token.dep_}
--> SentimentValueOfToken: {token_score} --> SentimentValueOfSentence:
{sentence_score}")

        #for T1--------------------
        tokenArray.append(token.text)
        tokenArray.append(ne)
        tokenArray.append(token.ent_type_)
        tokenArray.append(token.head)
        tokenArray.append(checkScore(token_score))
        tokenArray.append(checkScore(sentence_score))
        #print(tokenArray)
        t.append(tokenArray)
        #forT2------------------------
        if(ne==1):
          ttArray.append(token.ent_type_)
          ttArray.append(token.head)
          ttArray.append(checkScore(token_score))
          ttArray.append(checkScore(sentence_score))
          tt.append(ttArray)


t1=np.asarray(t)
#print(t1)
t2=np.asarray(tt)
#print(t2)
```

T1

[Token entity? entityType Governor SentimentValueOfToken SentimentValueOfSentence]

```
[['U.S.' 1 'GPE' agencies 'neutral' 'positive']
 ['intelligence' 0 '' agencies 'neutral' 'positive']
 ['agencies' 0 '' concluded 'neutral' 'positive']
 ['concluded' 0 '' concluded 'neutral' 'positive']
 ['in' 0 '' concluded 'neutral' 'positive']
 ['January' 1 'DATE' in 'neutral' 'positive']
 ['2017' 1 'DATE' January 'neutral' 'positive']
 ['that' 0 '' mounted 'neutral' 'positive']
 ['Russia' 1 'GPE' mounted 'neutral' 'positive']
 ['mounted' 0 '' concluded 'neutral' 'positive']
 ['a' 0 '' campaign 'neutral' 'positive']
 ['far' 0 '' ranging 'neutral' 'positive']
 ['-' 0 '' ranging 'neutral' 'positive']
 ['ranging' 0 '' campaign 'neutral' 'positive']
 ['influence' 0 '' campaign 'neutral' 'positive']
 ['campaign' 0 '' mounted 'neutral' 'positive']
 ['aimed' 0 '' campaign 'neutral' 'positive']
 ['at' 0 '' aimed 'neutral' 'positive']
 ['helping' 0 '' at 'positive' 'positive']
 ['Trump' 1 'PERSON' beat 'neutral' 'positive']
 ['beat' 0 '' helping 'neutral' 'positive']
 ['Clinton' 1 'PERSON' beat 'neutral' 'positive']
 ['.' 0 '' concluded 'neutral' 'positive']
 ['And' 0 '' affirmed 'neutral' 'positive']
 ['the' 0 '' Committee 'neutral' 'positive']
 ['bipartisan' 0 '' Committee 'neutral' 'positive']
 ['Senate' 1 'ORG' Committee 'neutral' 'positive']
 ['Intelligence' 1 'ORG' Committee 'neutral' 'positive']
 ['Committee' 1 'ORG' affirmed 'neutral' 'positive']
 [',' 0 '' affirmed 'neutral' 'positive']
 ['after' 0 '' affirmed 'neutral' 'positive']
 ['three' 1 'DATE' years 'neutral' 'positive']
 ['years' 1 'DATE' after 'neutral' 'positive']
 ['of' 0 '' years 'neutral' 'positive']
 ['investigation' 0 '' of 'neutral' 'positive']
 [',' 0 '' affirmed 'neutral' 'positive']
 ['affirmed' 0 '' affirmed 'neutral' 'positive']
 ['those' 0 '' conclusions 'neutral' 'positive']
 ['conclusions' 0 '' affirmed 'neutral' 'positive']
 [',' 0 '' affirmed 'neutral' 'positive']
```

```
 ['saying' 0 '' affirmed 'neutral' 'positive']
 ['intelligence' 0 '' officials 'neutral' 'positive']
 ['officials' 0 '' had 'neutral' 'positive']
 ['had' 0 '' saying 'neutral' 'positive']
 ['specific' 0 '' information 'neutral' 'positive']
 ['information' 0 '' had 'neutral' 'positive']
 ['that' 0 '' Trump 'neutral' 'positive']
 ['Russia' 1 'GPE' Trump 'neutral' 'positive']
 ['preferred' 0 '' Trump 'neutral' 'positive']
 ['Trump' 1 'PERSON' information 'neutral' 'positive']
 ['and' 0 '' Trump 'neutral' 'positive']
 ['that' 0 '' approved 'neutral' 'positive']
 ['Russian' 1 'NORP' President 'neutral' 'positive']
 ['President' 0 '' Putin 'neutral' 'positive']
 ['Vladimir' 1 'PERSON' Putin 'neutral' 'positive']
 ['Putin' 1 'PERSON' approved 'neutral' 'positive']
 ['had' 0 '' approved 'neutral' 'positive']
 ['"' 0 '' approved 'neutral' 'positive']
 ['approved' 0 '' Trump 'positive' 'positive']
 ['and' 0 '' approved 'neutral' 'positive']
 ['directed' 0 '' approved 'neutral' 'positive']
 ['aspects' 0 '' Trump 'neutral' 'positive']
 ['"' 0 '' Trump 'neutral' 'positive']
 ['of' 0 '' Trump 'neutral' 'positive']
 ['the' 0 '' Kremlin 'neutral' 'positive']
 ['Kremlin' 1 'ORG' campaign 'neutral' 'positive']
 ['s' 0 '' Kremlin 'neutral' 'positive']
 ['influence' 0 '' campaign 'neutral' 'positive']
 ['campaign' 0 '' of 'neutral' 'positive']
 ['.' 0 '' affirmed 'neutral' 'positive']]
```

T2

[entityType Governor sentimentValueOfToken sentimentValueOfSentence]

```
[['GPE' agencies 'neutral' 'positive']
 ['DATE' in 'neutral' 'positive']
 ['DATE' January 'neutral' 'positive']
 ['GPE' mounted 'neutral' 'positive']
 ['PERSON' beat 'neutral' 'positive']
 ['PERSON' beat 'neutral' 'positive']
 ['ORG' Committee 'neutral' 'positive']
 ['ORG' Committee 'neutral' 'positive']
 ['ORG' affirmed 'neutral' 'positive']
```

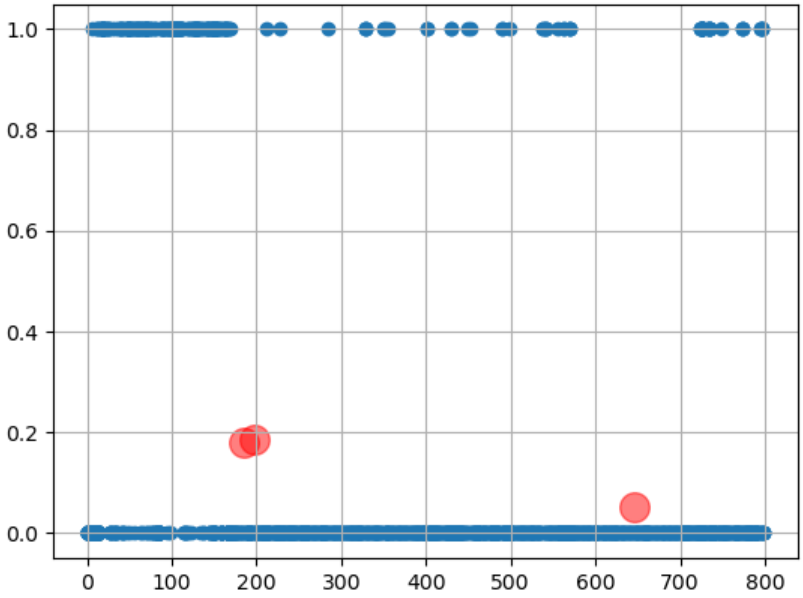
```
 ['DATE' years 'neutral' 'positive']
 ['DATE' after 'neutral' 'positive']
 ['GPE' Trump 'neutral' 'positive']
 ['PERSON' information 'neutral' 'positive']
 ['NORP' President 'neutral' 'positive']
 ['PERSON' Putin 'neutral' 'positive']
 ['PERSON' approved 'neutral' 'positive']
 ['ORG' campaign 'neutral' 'positive']]
```

D-

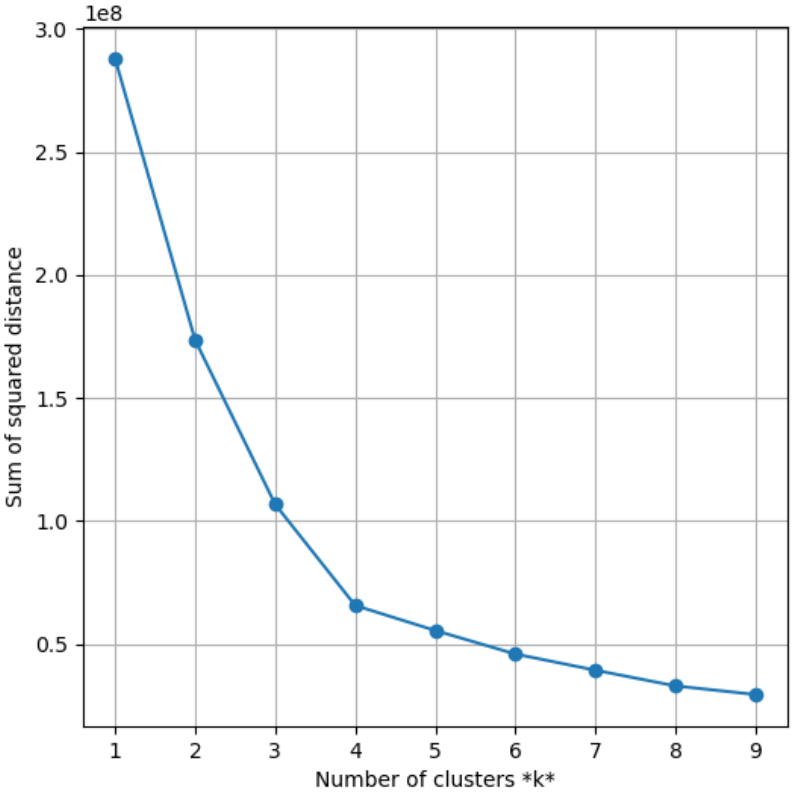
```
le = preprocessing.LabelEncoder()

X=t1[: , 0:6]
X[:, 0] = le.fit_transform(X[:, 0])#transforms the column 0 and so on
X[:, 2] = le.fit_transform(X[:, 2])
X[:, 3] = le.fit_transform(X[:, 3])
X[:, 4] = le.fit_transform(X[:, 4])
X[:, 5] = le.fit_transform(X[:, 5])

Y=t2[: , 0:4]
Y[:, 0] = le.fit_transform(Y[:, 0])
Y[:, 1] = le.fit_transform(Y[:, 1])
Y[:, 2] = le.fit_transform(Y[:, 2])
Y[:, 3] = le.fit_transform(Y[:, 3])

#for T1
kmeansT1 = KMeans(init='random', n_clusters=3, n_init=10)
kmeansT1.fit(X)
t1Labels=kmeansT1.labels_
t1center=kmeansT1.cluster_centers_
#print(kmeansT1.labels_)
# print(kmeansT1.cluster_centers_)

#elbow method analysis
sse=[]
list_k=list(range(1,10))
for k in list_k:
    km = KMeans(n_clusters=k)
```

```
    km.fit(X) #choose X or Y
    sse.append(km.inertia_)
plt.figure(figsize=(6, 6))
plt.plot(list_k, sse, '-o')
plt.xlabel(r'Number of clusters *k*')
plt.ylabel('Sum of squared distance');


#for T2
kmeansT2 = KMeans(init='random', n_clusters=2, n_init=10)
kmeansT2.fit(Y)
t2Labels=kmeansT2.labels_
t2center=kmeansT2.cluster_centers_
#print(kmeansT2.labels_)
# print(kmeansT2.cluster_centers_)

#for T1 graph
plt.scatter(x=X[:,0], y=X[:,1])
plt.scatter(x=t1center[:, 0],y=t1center[:, 1], c='red',s=200,
alpha=0.5,label='centroid')

#for T2 graph
plt.scatter(Y[:,0], Y[:,1],s=50)
plt.scatter(t2center[:, 0], t2center[:, 1], c='red',s=200,
alpha=0.5,label='centroid')

plt.grid()
plt.show()
```

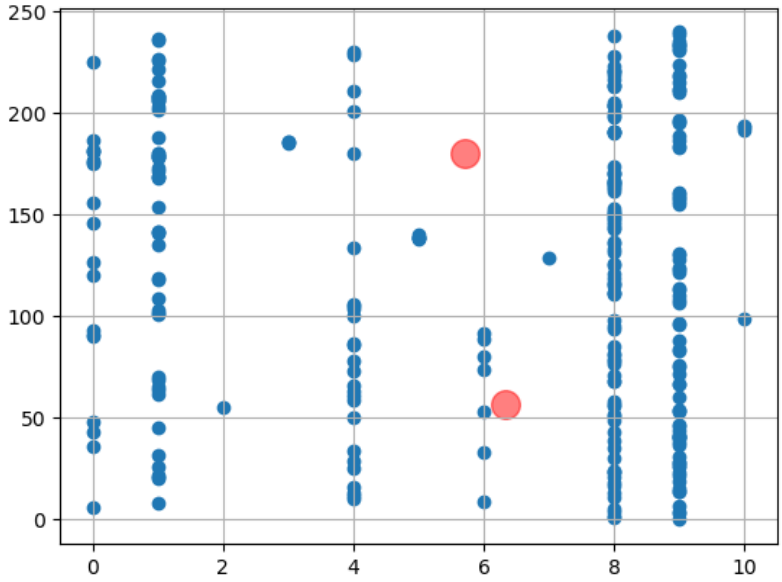
3-means cluster for T1



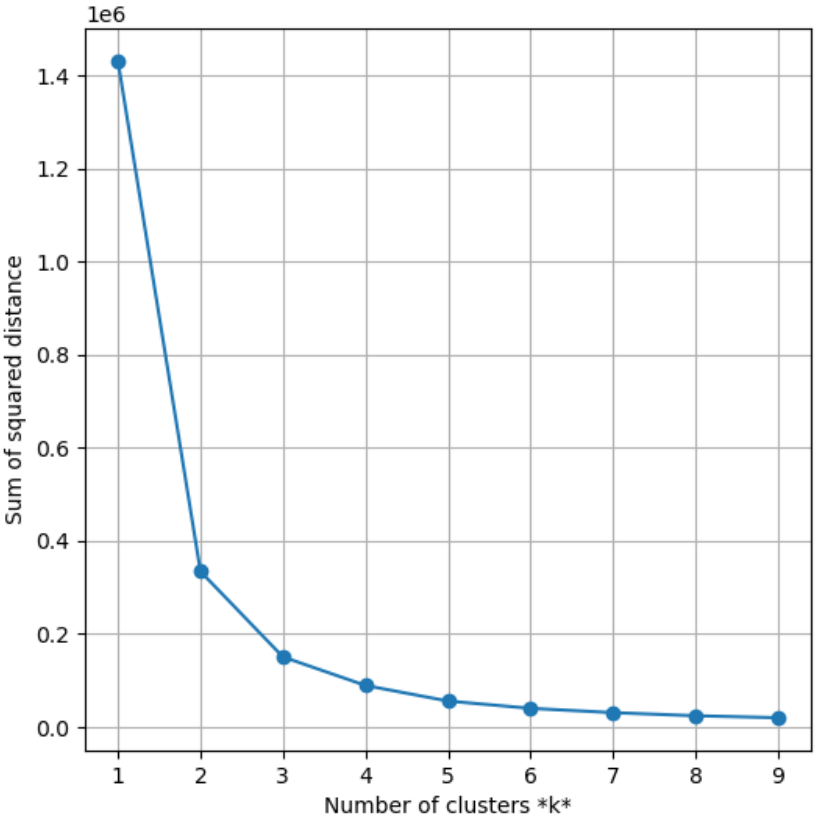
Elbow method analysis T1:



2-means cluster for T2



Elbow Method analysis T2



F-

Discussion:

The reason why the clusters look parallel is because of having converted the sentiment values of tokens and sentences, which were either 0 or 1. Most of it was due to having a large number of tokens that were neutral. As for the K-means, I believe it should have been better to increase the number of clusters for T1 to 4 at least, since the elbow graph shows that the curve slowly starts to flatten around k=4.

As for T2, since we cut down some data from T1, the result looks drastically different. The elbow curve shows that at k=2, we see the curve starting to flatten gradually, which is aligned with our chosen k-means value. Also, most of the data appears aligned vertically and in parallel, more dense towards the right side of the graph.

I believe the result can be improved by cleaning up a lot of punctuations that have been taken into account while running the processing bit of the experiment.