

ICP 7

1. Extract the following web URL text using BeautifulSoup

<https://en.wikipedia.org/wiki/Google>

2. Save it in input.txt3. Apply the following on the text and show output:

- a. Tokenization
- b. POS
- c. Stemming
- d. Lemmatization
- e. Trigram
- f. Named Entity Recognition

```
ICP7_1st.ipynb - Colaboratory
colab.research.google.com/drive/1gR9u2Bp3uQ8Rlqgk5MMUCRf6w/author=14scro15o-8pwe83oDrFqg

ICP7_1st.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
RAM 1
Disk 1
Editing

[28] #NAMED ENTITY RECOGNITION
from nltk import wordpunct_tokenize, pos_tag, ne_chunk
nltk.download('maxent_ne_chunker')
nltk.download('words')
print(ne_chunk(pos_tag(wordpunct_tokenize(w))))

w = w
//
2018/CD
//
(PERSON Google/HP)
confirmed/VBD
[]
295/CD
//
that/JN
it/PP
had/VBD
appealed/VBN
the/DT
fine/JN
to/TO
(ORGANIZATION thegeneral/J) courtof/HP
the/DT
(ORGANIZATION European/HP Union/HP)
./VBD
296/CD
//
ON/JN
January/HP
22/CD
//
2019/CD
//
(GPE French/HP)
data/NNS
regulatorCNLImposed/VBD
a/DT
```

```
ICP7_1st.ipynb - Colaboratory
colab.research.google.com/drive/1gR9u2Bp3uQ8Rlqgk5MMUCRf6w/author=14scro15o-8pwe83oDrFqg

ICP7_1st.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
RAM 1
Disk 1
Editing

[27] #Lemmatization
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
lem = WordNetLemmatizer()

for r in word_token:
    print(lem.lemmatize(r))

when
she
became
one
of
the
most
searched
item
on
the
search
engine
.
[
300
]
The
term
DeGoogle
..
ha
grown
in
use
asprivacyactivists
```

```
ICP7_1st.ipynb - Colaboratory
colab.research.google.com/drive/1gR9u2Bp3uQ8Rlqgk5MMUCRf6w/author=14scro15o-8pwe83oDrFqg

ICP7_1st.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
RAM 1
Disk 1
Editing

[29] #trigram
from nltk.util import ngrams
trigrams=ngrams(word_token,3)
for t in trigrams:
    print(t)

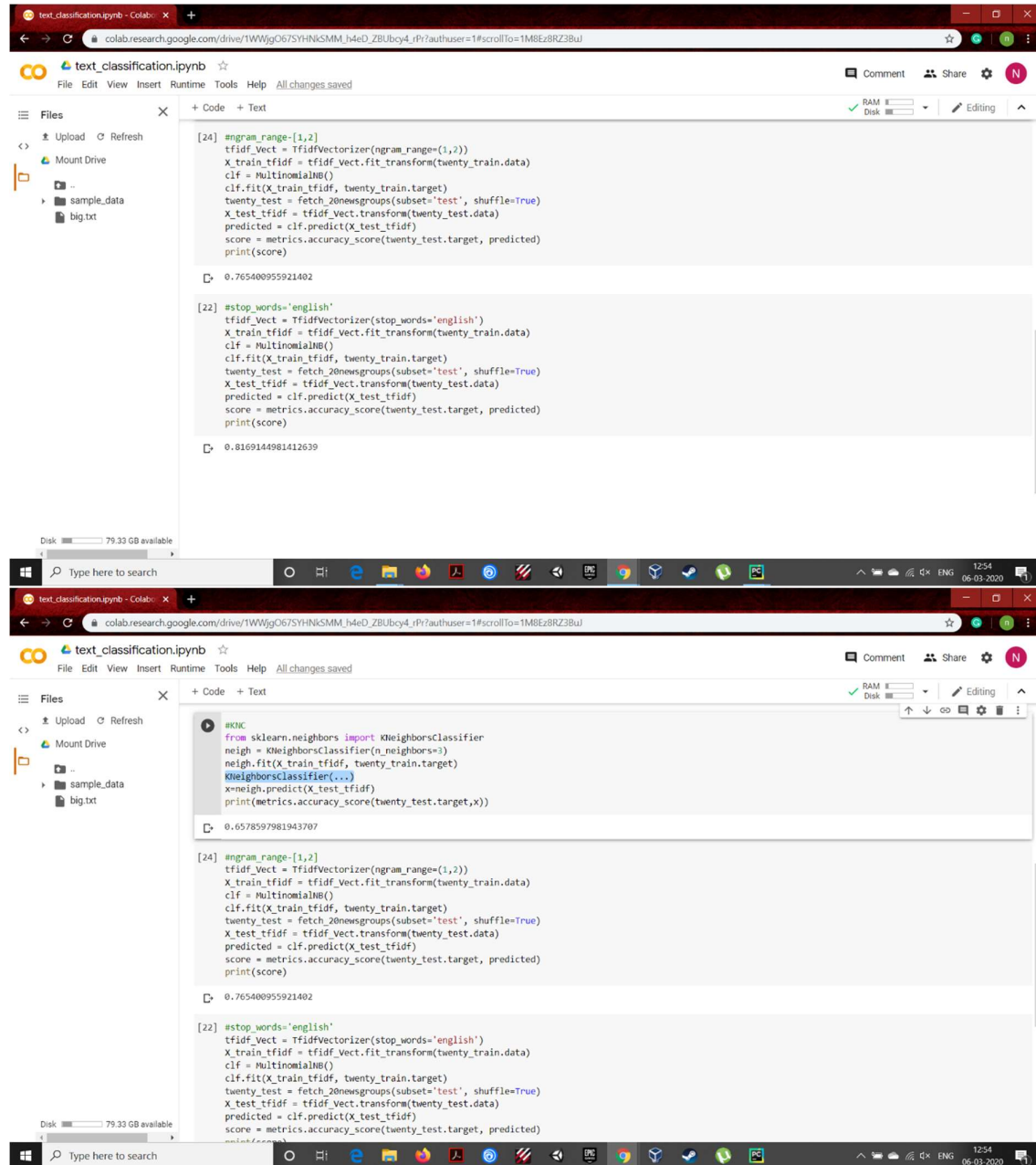
('of', 'trade', '.')
('trade', '.', 'anti-competitive')
('.', 'anti-competitive', 'practices')
('anti-competitive', 'practices', '.')
('practices', '.', 'adaptable')
('.', 'adaptable', 'infringement.google')
('adaptable', 'infringement.google', 'formerly')
('infringement.google', 'formerly', 'adhered')
('formerly', 'adhered', 'to')
('to', 'theInternet', 'copyright')
('theInternet', 'copyright', 'policies')
('copyright', 'policies', 'of')
('policies', 'of', 'China')
('of', 'China', '.')
('China', '.', '276')
('.', '276', '1')
('276', '1', 'enforced')
('1', 'enforced', 'by')
('enforced', 'by', 'means')
('by', 'means', 'of')
('means', 'of', 'filters')
('of', 'filters', 'colloquially')
('filters', 'colloquially', 'known')
('colloquially', 'known', 'as')
('known', 'as', '...')
('as', '...', 'threat')
('...', 'threat', 'firewall')
('threat', 'firewall', 'of')
('firewall', 'of', 'china')
```

4. Change the classifier in the given code to:

a. KNeighborsClassifier and see how accuracy changes

b. change the tfidfvectorizer to use bigram and see how the accuracy changes TfidfVectorizer(ngram_range=(1,2))

c. Put argument stop_words='english' and see how accuracy changes



The image shows two screenshots of a Google Colab notebook titled 'text_classification.ipynb'. The first screenshot shows the initial code and results. The second screenshot shows the code modified to use KNeighborsClassifier, TfidfVectorizer with ngram_range=(1,2), and stop_words='english'.

Initial Code and Results:

```
[24] ngram_range=[1,2]
tfidf_vect = TfidfVectorizer(ngram_range=(1,2))
X_train_tfidf = tfidf_vect.fit_transform(twenty_train.data)
clf = MultinomialNB()
clf.fit(X_train_tfidf, twenty_train.target)
twenty_test = fetch_20newsgroups(subset='test', shuffle=True)
X_test_tfidf = tfidf_vect.transform(twenty_test.data)
predicted = clf.predict(X_test_tfidf)
score = metrics.accuracy_score(twenty_test.target, predicted)
print(score)
```

0.765400955921402

```
[22] #stop_words='english'
tfidf_vect = TfidfVectorizer(stop_words='english')
X_train_tfidf = tfidf_vect.fit_transform(twenty_train.data)
clf = MultinomialNB()
clf.fit(X_train_tfidf, twenty_train.target)
twenty_test = fetch_20newsgroups(subset='test', shuffle=True)
X_test_tfidf = tfidf_vect.transform(twenty_test.data)
predicted = clf.predict(X_test_tfidf)
score = metrics.accuracy_score(twenty_test.target, predicted)
print(score)
```

0.8169144981412639

Modified Code and Results:

```
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(X_train_tfidf, twenty_train.target)
KNeighborsClassifier(...)
x=neigh.predict(X_test_tfidf)
print(metrics.accuracy_score(twenty_test.target,x))
```

0.6578597981943707

```
[24] ngram_range=[1,2]
tfidf_vect = TfidfVectorizer(ngram_range=(1,2))
X_train_tfidf = tfidf_vect.fit_transform(twenty_train.data)
clf = MultinomialNB()
clf.fit(X_train_tfidf, twenty_train.target)
twenty_test = fetch_20newsgroups(subset='test', shuffle=True)
X_test_tfidf = tfidf_vect.transform(twenty_test.data)
predicted = clf.predict(X_test_tfidf)
score = metrics.accuracy_score(twenty_test.target, predicted)
print(score)
```

0.765400955921402

```
[22] #stop_words='english'
tfidf_vect = TfidfVectorizer(stop_words='english')
X_train_tfidf = tfidf_vect.fit_transform(twenty_train.data)
clf = MultinomialNB()
clf.fit(X_train_tfidf, twenty_train.target)
twenty_test = fetch_20newsgroups(subset='test', shuffle=True)
X_test_tfidf = tfidf_vect.transform(twenty_test.data)
predicted = clf.predict(X_test_tfidf)
score = metrics.accuracy_score(twenty_test.target, predicted)
print(score)
```

BY

DUKKIPATI SRI SAI NITHIN CHOWDARY

CLASS ID : 4