

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/1gR9u2Bp3uQ8Rgk5MMUCRf6w/author=14scrollTo=8pwl3oDrFqg

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

Files
Upload Refresh
Unmount Drive
drive
My Drive
AICHAT PPT
AICHAT VIDEO
Colab Notebooks
Copy of test...
ICP7_1st.ipynb
ICP7_2nd.ipynb
big.txt
spelling_corre...
test.ipynb
webscrap.txt
kolmini project vi...
Video for increm...
personal docume...
tweets
video
video increment 4
AI BASED SURVE...
AI BASED SURVE...
AI BASED SURVE...
79.21 GB available

[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/WWP)
confirmed/VBD
//
295/CD
//
that/JN
it/PP
had/VBD
appealed/VBN
the/DT
fine/JN
to/TO
(ORGANIZATION thegeneral/JJ court/JN)
the/DT
(ORGANIZATION European/WWP Union/WWP)
./VBD
296/CD
//
ON/JN
January/WWP
22/CD
//
2019/CD
//
(GPE French/WWP)
data/NNS
regulatorCNLImposed/VBD
a/DT
```

```
ICP7_1st.ipynb - Colaboratory
colab.research.google.com/drive/1gR9u2Bp3uQ8Rgk5MMUCRf6w/author=14scrollTo=8pwl3oDrFqg

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

Files
Upload Refresh
Unmount Drive
drive
My Drive
AICHAT PPT
AICHAT VIDEO
Colab Notebooks
Copy of test...
ICP7_1st.ipynb
ICP7_2nd.ipynb
big.txt
spelling_corre...
test.ipynb
webscrap.txt
kolmini project vi...
Video for increm...
personal docume...
tweets
video
video increment 4
AI BASED SURVE...
AI BASED SURVE...
AI BASED SURVE...
79.21 GB available

[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/1gR9u2Bp3uQ8Rgk5MMUCRf6w/author=14scrollTo=8pwl3oDrFqg

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

Files
Upload Refresh
Unmount Drive
drive
My Drive
AICHAT PPT
AICHAT VIDEO
Colab Notebooks
Copy of test...
ICP7_1st.ipynb
ICP7_2nd.ipynb
big.txt
spelling_corre...
test.ipynb
webscrap.txt
kolmini project vi...
Video for increm...
personal docume...
tweets
video
video increment 4
AI BASED SURVE...
AI BASED SURVE...
AI BASED SURVE...
79.21 GB available

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

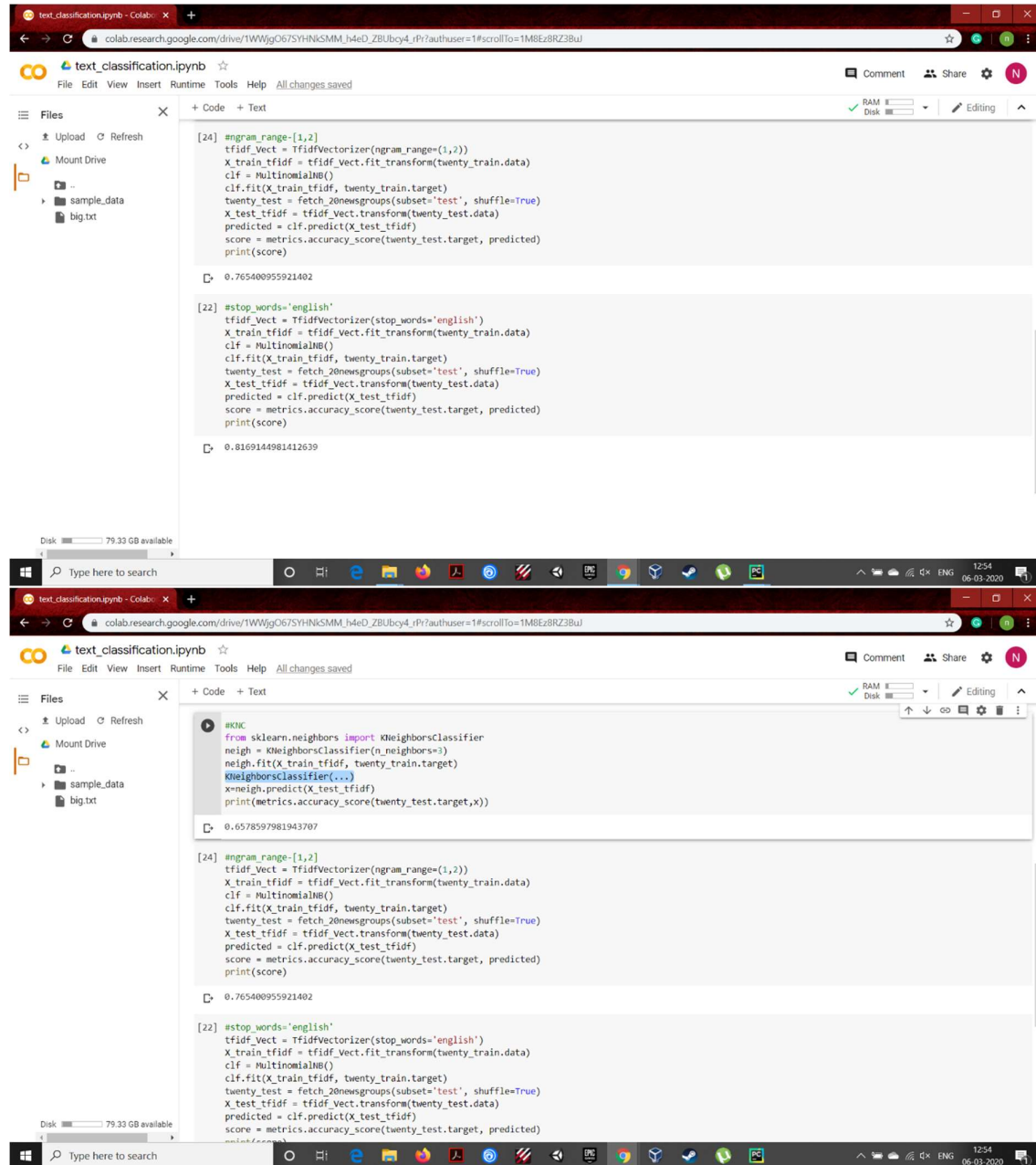
('of', 'trade', '.')
('trade', '.', 'anti-competitive')
('.', 'anti-competitive', 'practices')
('anti-competitive', 'practices', '.')
('practices', '.', 'adaptable')
('.', 'adaptable', 'infringement.google')
('adaptable', 'infringement.google', 'formerly')
('infringement.google', 'formerly', 'adhered')
('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 displays two screenshots of a Google Colab notebook titled 'text_classification.ipynb'. The notebook is open in a web browser, showing the file explorer on the left and the code editor on the right. The code is written in Python and uses the sklearn library for machine learning.

The first screenshot shows the initial code (lines 24-35) which uses a `TfidfVectorizer` with `ngram_range=(1,2)` and a `LogisticRegression` classifier. The accuracy score is printed as 0.765400955921402.

The second screenshot shows the code after modifications (lines 22-35). The `TfidfVectorizer` is updated with `stop_words='english'`. The classifier is changed to `KNeighborsClassifier` with `n_neighbors=3`. The accuracy score is printed as 0.8169144981412639.

```
[24] ngram_range=(1,2)
tfidf_vect = TfidfVectorizer(ngram_range=(1,2))
X_train_tfidf = tfidf_vect.fit_transform(twenty_train.data)
clf = LogisticRegression()
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 = LogisticRegression()
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
```

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

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 = LogisticRegression()
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 = LogisticRegression()
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
```