
SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
*(Deemed to be* **University** *u/s 3 of UGC Act, 1956)*

# DEPARTMENT OF COMPUTER APPLICATIONS

## SRM Institute of Science & Technology

## NCR Campus, Modinagar

### PRACTICAL FILE

### INTRODUCTION TO NATURAL LANGUAGE PROCESSING

### (UDS21303J)

### PROGRAMME: - BCA [DATA SCIENCE]

### YEAR: - 2nd YEAR, 3rd SEMESTER

**Submitted To:**

**Dr. Vishu V.**

**(Associate Professor)**

**Submitted By:**

**Anubhav Yadav**

**RA2231242030004**

**SRM INSTITUTE OF SCIENCE & TECHNOLOGY, NCR CAMPUS, MODINAGAR**

**DEPARTMENT FACULTY OF SCIENCE AND HUMANITIES**

**Register No.:** RA2231242030004

**BONAFIDE CERTIFICATE**

Certified to be the Bonafide record of the work done by **Anubhav Yadav** of BCA-DS, Second year, III Semester for the award of **Bachelor's** degree course in the Department of Computer Applications in **Introduction to Natural Language Processing [UDS21303J]** laboratory during the Academic year-2023-24.

**LAB IN-CHARGE** **HEAD OF DEPARTMENT**

Submitted for the university examination held on _____

**INTERNAL EXAMINER 1** **INTERNAL EXAMINER 2**

# INDEX

| S. No. | Experiment | Date | Signature |
|---|---|---|---|
| 1. | Tokenization | | |
| 2. | Lemmatization | | |
| 3. | Stopwords | | |
| 4. | Stemming | | |
| 5. | Regular Expression Email Detection | | |
| 6. | Regular Expression Numerical Values Detection | | |
| 7. | Frequency of Words & Distinct Words | | |
| 8. | WordPunctTokenizer | | |
| 9. | POS Tags | | |
| 10. | Read dataset, get top 5 records and shape of the dataframe | | |
| 11. | Use info() and describe() function | | |
| 12. | Change to datetime datatype of a dataframe column and check number of unique values | | |
| 13. | Check the number of null values and fill the null values in 'Gender' column | | |
| 14. | Drop the remaining null values and plot the histogram | | |
| 15. | Plot the boxplot and scatterplot (with legend) | | |

# Experiment 1 (Tokenization)

1.Write a Python NLTK program to tokenize words, sentence wise.

**Program**

```
from nltk.tokenize import sent_tokenize,word_tokenize

text = """Amidst the tranquil meadow, a symphony of colors danced as wildflowers swayed in the breeze. Sunlight painted golden ribbons upon the azure canvas of the sky. Birds orchestrated a melodious chatter while a distant stream added its soothing rhythm. Nature's masterpiece unfolded, inviting weary souls to find solace within its embrace."""

print('\n')

print('Original Text:')

print(text)

print('\n')

print('Sent Tokenize:')

print(sent_tokenize(text))

print('\n')

print('Word Tokenize:')

result = [word_tokenize(word) for word in sent_tokenize(text)]

for i in result:

print(i)
```

**Output**

```
Original Text:
Amidst the tranquil meadow, a symphony of colors danced as wildflowers swayed in the
breeze. Sunlight painted golden ribbons upon the azure canvas of the sky. Birds orchestrated
a melodious chatter while a distant stream added its soothing rhythm. Nature's masterpiece
unfolded, inviting weary souls to find solace within its embrace.


Sent Tokenize:
['Amidst the tranquil meadow, a symphony of colors danced as wildflowers swayed in the \nbreeze.', 'S
unlight painted golden ribbons upon the azure canvas of the sky.', 'Birds orchestrated\na melodious c
hatter while a distant stream added its soothing rhythm.', "Nature's masterpiece \nunfolded, inviting
weary souls to find solace within its embrace."]


Word Tokenize:
['Amidst', 'the', 'tranquil', 'meadow', ',', 'a', 'symphony', 'of', 'colors', 'danced', 'as', 'wildfl
owers', 'swayed', 'in', 'the', 'breeze', '.']
['Sunlight', 'painted', 'golden', 'ribbons', 'upon', 'the', 'azure', 'canvas', 'of', 'the', 'sky',
'.']
['Birds', 'orchestrated', 'a', 'melodious', 'chatter', 'while', 'a', 'distant', 'stream', 'added', 'i
ts', 'soothing', 'rhythm', '.']
['Nature', "'s", 'masterpiece', 'unfolded', ',', 'inviting', 'weary', 'souls', 'to', 'find', 'solac
e', 'within', 'its', 'embrace', '.']
```

# Experiment 2 (Lemmatization)

2. Write a Python NLTK program to lemmatize words.

**Program**

```
from nltk.stem import WordNetLemmatizer
lm = WordNetLemmatizer()
words = ['read','reads','reading']
for i in words:
    print(i,':',lm.lemmatize(i))
```

**Output**

```
read : read
reads : read
reading : reading
```

# Experiment 3 (Stopwords)

3. Write a Python NLTK program to remove stopwords from text.

**Program**

```
from nltk.corpus import stopwords

text = """The bustling city streets bustled with people hurrying to their destinations. Amidst the chaos, a sense of purpose lingered in the air. The constant motion painted a vivid picture of urban life, where time never seemed to pause. Yet, amidst it all, moments of quiet introspection emerged."""

stopword = stopwords.words('english')

word = word_tokenize(text)

filter_words = [ ]

for i in word:

    if i not in stopword:

        filter_words.append(i)

print(filter_words)
```

**Output**

```
['The', 'bustling', 'city', 'streets', 'bustled', 'people', 'hurrying', 'destinations', '.', 'Amids
t', 'chaos', ',', 'sense', 'purpose', 'lingered', 'air', '.', 'The', 'constant', 'motion', 'painted',
'vivid', 'picture', 'urban', 'life', ',', 'time', 'never', 'seemed', 'pause', '.', 'Yet', ',', 'amids
t', ',', 'moments', 'quiet', 'introspection', 'emerged', '.']
```

# Experiment 4 (Stemming)

4. Write a Python NLTK program to stemming words.

**Program**

```
from nltk.stem import PosterStemmer
ps = PosterStemmer()
words = [‘program’ , ’programs’ , ’programming’ , ‘programmers’ , ‘programmer’]
for i in words:
        print(i , ‘:’ , ps.stem(i))
```

**Output**

```
program : program
programs : program
programming : program
programmers : programm
programmer : programm
```

# Experiment 5

## (Regular Expressions Email Detection)

5. Extracting email addresses using regular expressions in Python

**Program**

```
import re

text = """Amidst the city's hustle, Jane found solace in a hidden cafe. She reunited with friends: Mark at mark34@gmail.com, Emily via emily.smith@email.com, Jason through jason_doe@hotmail.com, Lisa using lisa.jones@example.org, and Alex at alex.brown@email.com Over lattes, they laughed, shared stories, and cemented their bond."""

lst  = re.findall('\S+@S\+',text)

print(lst)
```

**Output**

```
In [29]: print(lst)

['mark34@gmail.com,', 'emily.smith@email.com,', 'jason_doe@hotmail.com,', 'lisa.jones@example.org,',
'alex.brown@email.com']
```

# Experiment 6

## (Regular Expression Numerical Values Detection)

6. Extracting numerical values using regular expressions in Python

**Program**

```
import re

text = """In a field of wildflowers, there stood a house numbered 42. A group of 8 adventurers gathered, carrying 25 backpacks filled with supplies. They embarked on a 10-mile hike, reaching a summit 3000 meters high. As the sun set, they captured 50 photos, preserving memories of their remarkable journey."""

num_lst = re.findall('[0-9]',text)

print(num_lst)
```

**Output**

```
In [32]: print(num_lst)

['42', '8', '25', '10', '3000', '50']
```

# Experiment 7

## (Frequency of Words & Distinct Words)

7. WAP to Create a monolingual corpus Choose a corpus of 200,000 words. Segment it into words and compute the frequency of each word. How many distinct words are there? count frequencies of bigrams (two consecutive words) and trigrams (three consecutive words).

**Program**

```
import nltk

from nltk import FreqDist

from nltk.util import bigrams,trigrams

from nltk.corpus import reuters


nltk.download('reuters')

corpus_words = reuters.words()

corpus_length = len(corpus_words)

word_freq = FreqDist(corpus_words)

bigram_freq = FreqDist(list(bigrams(corpus_words)))

trigram_freq = FreqDist(list(trigrams(corpus_words)))

distinct_words = len(word_freq)

print(f"Distinct Words:{distinct_words}")


print("\nTop 10 most common words:")

print(word_freq.most_common(10))


print("\nTop 10 most common bigrams:")

print(bigram_freq.most_common(10))


print("\nTop 10 most common trigrams:")

print(trigram_freq.most_common(10))
```

## Output

```
Distinct Words:41600

Top 10 most common words:
[('.', 94687), (',', 72360), ('the', 58251), ('of', 35979), ('to', 34035), ('in', 26478), ('said', 25
224), ('and', 25043), ('a', 23492), ('mln', 18037)]

Top 10 most common bigrams:
[((',', '000'), 10266), (("'", 's'), 9220), (('lt', ';'), 8693), (('&', 'lt'), 8688), (('.', 'The'),
8530), (('said', '.'), 7888), (('of', 'the'), 6803), (('in', 'the'), 6487), (('U', '.'), 6350),
(('.', 'S'), 5833)]

Top 10 most common trigrams:
[(('&', 'lt', ';'), 8687), (('U', '.', 'S'), 5693), (('.', 'S', '.'), 5360), ((',', '000', 'vs'), 257
7), (('the', 'U', '.'), 1959), ((',', '000', 'dlrs'), 1524), (('said', '.', 'The'), 1516), (('.',
'5', 'mln'), 1337), (('he', 'said', '.'), 1229), ((',', '000', 'Revs'), 1198)]
```

# Experiment 8 (WordPunctTokenizer)

8. Write a program with your knowledge of the English language, split 10 sentences of your choice into words and punctuation: Find out the words words that do not usually appear in a standard lexicon? The separators are whitespaces quote (’), full- stop/period (.), parenthesis, are kept as tokens, tokenize the earlier sentence.

**Program**

```
import nltk

from nltk.tokenize import WordPunctTokenizer

sentences = [

    "The sun sets behind the mountains, painting the sky in hues of orange and pink.",

    "Dogs playfully chase each other, their tails wagging with joy!",

    "After a long day at work, I love to relax with a good book or a movie.",

    "'Where are you going?' she asked, her voice filled with curiosity.",

    "The smell of freshly baked cookies wafts through the air, making my mouth water.",

    "The rain patters on the roof, creating a soothing rhythm that lulls me to sleep.",

    "Excitement bubbles up within me—I can't wait for the upcoming vacation!",

    "Traffic in the city can be quite chaotic; it's like a symphony of honking horns.",

    "With a deep breath, he stepped onto the stage, ready to face the audience.",

    "The old house on the hill seems haunted; its windows are shattered, and the garden is overgrown."

]

tokenizer = WordPunctTokenizer()

tokenized_sent = [tokenizer.tokenize(sentence) for sentence in sentences]

for i,tokens in enumerate(tokenized_sent):

    print(f"Sentence {i+1}:",tokens,'\n')
```

## Output

In [7]:
```
for i,tokens in enumerate(tokenized_sent):
    print(f"Sentence {i+1}:",tokens,'\n')
```

```
Sentence 1: ['The', 'sun', 'sets', 'behind', 'the', 'mountains', ',', 'painting', 'the', 'sky', 'in', 'hues', 'of', 'orange',
'and', 'pink', '.']

Sentence 2: ['Dogs', 'playfully', 'chase', 'each', 'other', ',', 'their', 'tails', 'wagging', 'with', 'joy', '!']

Sentence 3: ['After', 'a', 'long', 'day', 'at', 'work', ',', 'I', 'love', 'to', 'relax', 'with', 'a', 'good', 'book', 'or',
'a', 'movie', '.']

Sentence 4: ["'", 'Where', 'are', 'you', 'going', "?'", 'she', 'asked', ',', 'her', 'voice', 'filled', 'with', 'curiosity',
'.']

Sentence 5: ['The', 'smell', 'of', 'freshly', 'baked', 'cookies', 'wafts', 'through', 'the', 'air', ',', 'making', 'my', 'mout
h', 'water', '.']

Sentence 6: ['The', 'rain', 'patters', 'on', 'the', 'roof', ',', 'creating', 'a', 'soothing', 'rhythm', 'that', 'lulls', 'me',
'to', 'sleep', '.']

Sentence 7: ['Excitement', 'bubbles', 'up', 'within', 'me', '—', 'I', 'can', "'", 't', 'wait', 'for', 'the', 'upcoming', 'vacat
ion', '!']

Sentence 8: ['Traffic', 'in', 'the', 'city', 'can', 'be', 'quite', 'chaotic', ';', 'it', "'", 's', 'like', 'a', 'symphony', 'o
f', 'honking', 'horns', '.']

Sentence 9: ['With', 'a', 'deep', 'breath', ',', 'he', 'stepped', 'onto', 'the', 'stage', ',', 'ready', 'to', 'face', 'the', 'a
udience', '.']

Sentence 10: ['The', 'old', 'house', 'on', 'the', 'hill', 'seems', 'haunted', ';', 'its', 'windows', 'are', 'shattered', ',',
'and', 'the', 'garden', 'is', 'overgrown', '.']
```

# EXPERIMENT 9 (POS Tags)

9. Write a pos tags program where we can see each word has its own lexical term tags.

**Program**

```
import nltk
from nltk import word_tokenize
sentence="Hello World! This is NLP Program. And I am writing POS tags program."
for i in sentence.split():
    print(nltk.pos_tag(word_tokenize(i)))
```

**Output**

```
[('Hello', 'NN')]
[('World', 'NN'), ('!', '.')]
[('This', 'DT')]
[('is', 'VBZ')]
[('NLP', 'NN')]
[('Program', 'NN'), ('.', '.')]
[('And', 'CC')]
[('I', 'PRP')]
[('am', 'VBP')]
[('writing', 'VBG')]
[('POS', 'NN')]
[('tags', 'NNS')]
[('program', 'NN'), ('.', '.')]
```

# EXPERIMENT 10 (EMPLOYEE DATASET)

10. Import necessary libraries and check top 5 records and shape of the dataframe.

**Program**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline

df = pd.read_csv('employees.csv')
df.head()
```

**Output**

Out[4]:

| | First Name | Gender | Start Date | Last Login Time | Salary | Bonus % | Senior Management | Team |
|---|---|---|---|---|---|---|---|---|
| 0 | Douglas | Male | 8/6/1993 | 12:42 PM | 97308 | 6.945 | True | Marketing |
| 1 | Thomas | Male | 3/31/1996 | 6:53 AM | 61933 | 4.170 | True | NaN |
| 2 | Maria | Female | 4/23/1993 | 11:17 AM | 130590 | 11.858 | False | Finance |
| 3 | Jerry | Male | 3/4/2005 | 1:00 PM | 138705 | 9.340 | True | Finance |
| 4 | Larry | Male | 1/24/1998 | 4:47 PM | 101004 | 1.389 | True | Client Services |

**Program**

```
df.shape
```

**Output**

```
In [ ]: df.shape
Out[6]: (1000, 8)
```

# EXPERIMENT 11 (EMPLOYEE DATASET)

11. Check the information and describe function on defined dataframe.

**Program**

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   First Name        933 non-null    object
 1   Gender            855 non-null    object
 2   Start Date        1000 non-null   object
 3   Last Login Time   1000 non-null   object
 4   Salary            1000 non-null   int64
 5   Bonus %           1000 non-null   float64
 6   Senior Management  933 non-null   object
 7   Team              957 non-null    object
dtypes: float64(1), int64(1), object(6)
memory usage: 62.6+ KB
```

df.describe(include='object')

Out[8]:

| | First Name | Gender | Start Date | Last Login Time | Senior Management | Team |
|---|---|---|---|---|---|---|
| **count** | 933 | 855 | 1000 | 1000 | 933 | 957 |
| **unique** | 200 | 2 | 972 | 720 | 2 | 10 |
| **top** | Marilyn | Female | 10/30/1994 | 1:35 PM | True | Client Services |
| **freq** | 11 | 431 | 2 | 5 | 468 | 106 |

df.describe()

Out[9]:

| | Salary | Bonus % |
|---|---|---|
| **count** | 1000.000000 | 1000.000000 |
| **mean** | 90662.181000 | 10.207555 |
| **std** | 32923.693342 | 5.528481 |
| **min** | 35013.000000 | 1.015000 |
| **25%** | 62613.000000 | 5.401750 |
| **50%** | 90428.000000 | 9.838500 |
| **75%** | 118740.250000 | 14.838000 |
| **max** | 149908.000000 | 19.944000 |

# EXPERIMENT 12 (EMPLOYEE DATASET)

12. Convert "Start Date" column to datetime data type and check the number of unique data in dataframe.

**Program**

```
df['Start Date'] = pd.to_datetime(df['Start Date'])
```

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   First Name        933 non-null    object
 1   Gender            855 non-null    object
 2   Start Date        1000 non-null   datetime64[ns]
 3   Last Login Time   1000 non-null   object
 4   Salary            1000 non-null   int64
 5   Bonus %           1000 non-null   float64
 6   Senior Management 933 non-null    object
 7   Team              957 non-null    object
dtypes: datetime64[ns](1), float64(1), int64(1), object(5)
memory usage: 62.6+ KB
```

df.nunique()

```
Out[12]: First Name          200
         Gender              2
         Start Date        972
         Last Login Time   720
         Salary            995
         Bonus %           971
         Senior Management   2
         Team               10
         dtype: int64
```

# EXPERIMENT 13 (EMPLOYEE DATASET)

13. Check the sum of all null values in dataframe and fill the ‘Gender’ column with ‘No Gender’ value where value is empty or null.

**Program**

df.isnull().sum()

```
Out[13]: First Name            67
         Gender               145
         Start Date             0
         Last Login Time        0
         Salary                 0
         Bonus %                0
         Senior Management     67
         Team                  43
         dtype: int64
```

df['Gender'].fillna('No Gender',inplace=True)

df['Gender'].unique()

```
Out[23]: array(['Male', 'Female', 'No Gender'], dtype=object)
```

# EXPERIMENT 14 (EMPLOYEE DATASET)

14. Drop remaining null value, which is, still consist in dataframe and plot a histogram of 'Salary' column.

**Program**

```
df = df.dropna(axis=0,how='any')
```

```
df.isnull().sum()
```

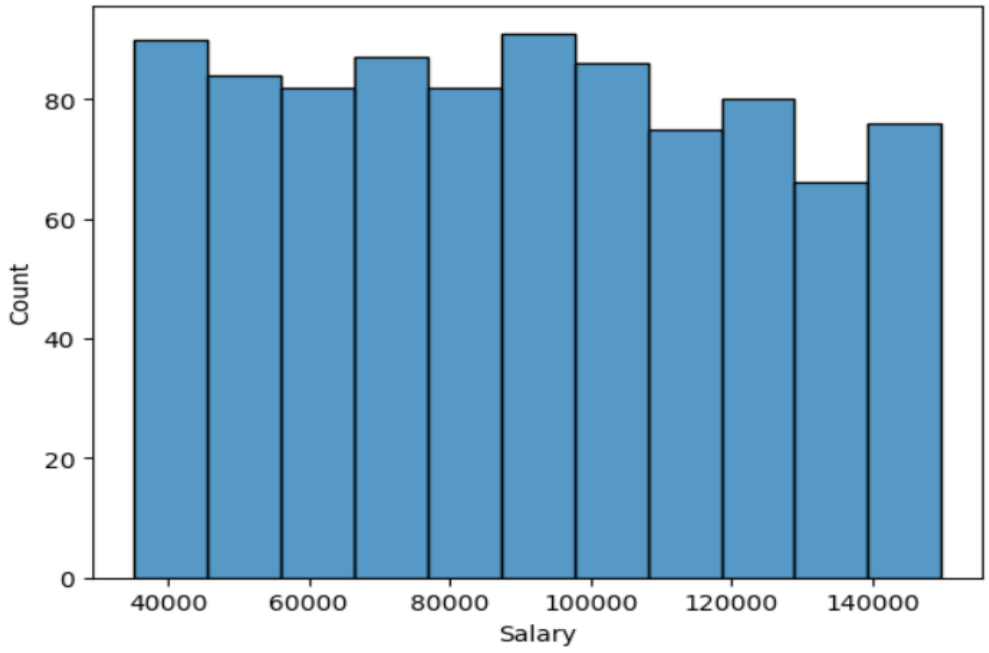
```
Out[17]: First Name          0
         Gender              0
         Start Date          0
         Last Login Time     0
         Salary              0
         Bonus %             0
         Senior Management   0
         Team                0
         dtype: int64
```

```
df.shape
```

```
Out[18]: (899, 8)
```

```
sns.histplot(x='Salary',data=df)
```

```
Out[19]: <AxesSubplot:xlabel='Salary', ylabel='Count'>
```



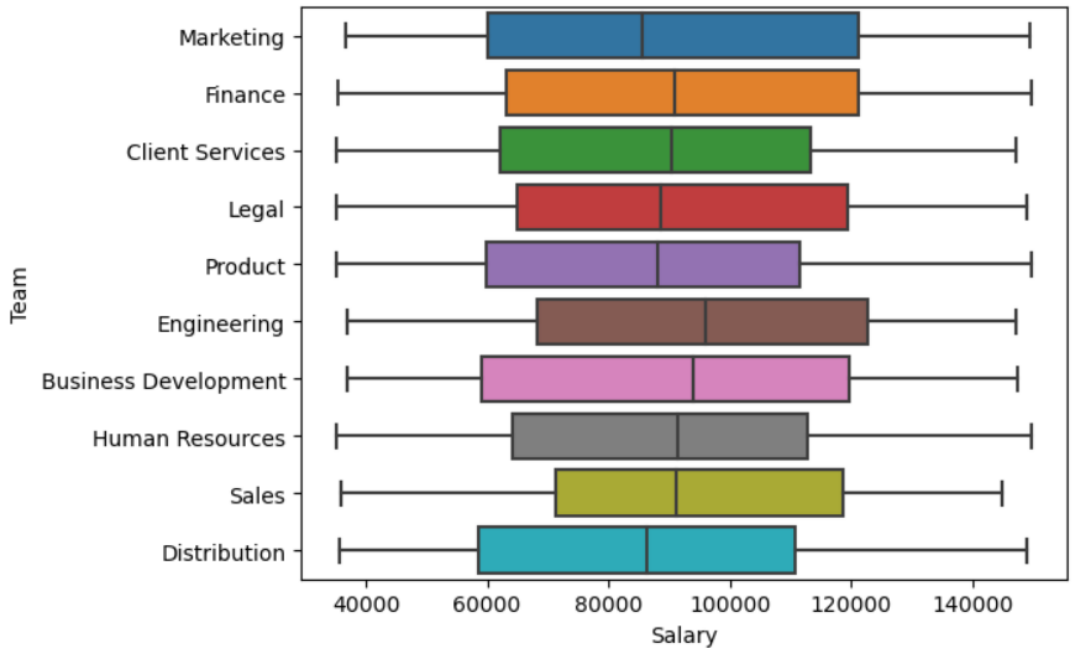
# EXPERIMENT 15 (EMPLOYEE DATASET)

15. Plot a boxplot and scatterplot (with legend) for ‘Salary’ vs ‘Team’ column.

**Program**

```
sns.boxplot(x='Salary',y='Team',data=df)
```

```
Out[20]: <AxesSubplot:xlabel='Salary', ylabel='Team'>
```




```
sns.scatterplot(x='Salary',y='Team',data=df,hue='Gender',size='Bonus %')
plt.legend(bbox_to_anchor=(1,1),loc=2)
```

```
Out[21]: <matplotlib.legend.Legend at 0x276d18d6280>
```