**PROGRAM**

```
# ex1
import re

def detect_word_pattern(pattern, text):
      matches = re.findall(pattern, text)

if matches:
      print("Word patterns detected:")
      for match in matches:
            print(match)
else:
      print("No word patterns detected.")

# sample inputs and outputs
sample_inputs = [
      ("[0-9]+","The price is $25 and the quantity is 10."),
      ("[A-Z][a-z]+","John and Alice went to the park."),
      ("[aeiou]+","The quick brown for jumps over the lazy dog."),
      ("[0-9]{2}-[0-9]{2}-[0-9]{4}","The date is 12-31-2022."),
      ("[A-Za-z]+","12345 is a number.")
]

for pattern, text in sample_inputs:
      print("patterns:",pattern)
      print("Text:",text)
      detect_word_pattern(pattern, text)
      print("--------------------")
```

**OUTPUT**

```
>>>
=============== RESTART: /home/student/Jerald/ex1/word_pattern.py
patterns: [0-9]+
Text: The price is $25 and the quantity is 10.
Word patterns detected:
25
10
-------------------
patterns: [A-Z][a-z]+
Text: John and Alice went to the park.
Word patterns detected:
John
Alice
-------------------
patterns: [aeiou]+
Text: The quick brown for jumps over the lazy dog.
Word patterns detected:
e
ui
o
o
u
o
e
e
a
o
-------------------

patterns: [0-9]{2}-[0-9]{2}-[0-9]{4}
Text: The date is 12-31-2022.
Word patterns detected:
12-31-2022
-------------------
patterns: [A-Za-z]+
Text: 12345 is a number.
Word patterns detected:
is
a
number
-------------------
>>>
```

**PROGRAM**

```
#ex2
import nltk
def tokenize_text(text):
    tokens = nltk.word_tokenize(text)
    return tokens
sample_inputs = ["Manoah is a college Student.",
                 "He loves Music.",
                 "His favourite domain is Data Analyst."
                 ]
for input_text in sample_inputs:
    print("Input Text:",input_text)
    tokens = tokenize_text(input_text)
    print("Tokens:",tokens)
    print("---------------")
```

**OUTPUT**

```
Python 3.8.10 (default, May 26 2023, 14:05:08)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========================= RESTART: /home/student/Ex2.py ==================
Input Text: Manoah is a college Student.
Tokens: ['Manoah', 'is', 'a', 'college', 'Student', '.']
---------------
Input Text: He loves Music.
Tokens: ['He', 'loves', 'Music', '.']
---------------
Input Text: His favourite domain is Data Analyst.
Tokens: ['His', 'favourite', 'domain', 'is', 'Data', 'Analyst', '.']
---------------
>>>
```

**PROGRAM**

```
#ex3
import nltk
#nltk.download('stopwords')
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
data="Let your faith be bigger than your fears."
stopWords=set(stopwords.words('english'))
words=word_tokenize(data.lower())
wordFiltered=[]
for w in words:
      if w not in stopWords:
            wordFiltered.append(w)
print("Words=",words)
print("Stopwords=",stopWords)
print("Filtered words=",wordFiltered)
```

**OUTPUT**

```
Python 3.8.10 (default, May 26 2023, 14:05:08)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============== RESTART: /home/student/Jerald/nlp/ex3/stopwords.py ==============
Words= ['let', 'your', 'faith', 'be', 'bigger', 'than', 'your', 'fears', '.']
Stopwords= {'yourself', 'where', 'each', 'how', 'here', "you've", 'it', 'why', "isn't", "hadn't", 'other', 'haven', "shan't", "she's", 'out',
'being', 'can', 'doesn', 'didn', 'this', 'having', 'isn', 'needn', 'same', "haven't", 'does', 'him', "wouldn't", 'about', 'hadn', 'against',
'me', 'those', 'from', 'i', 'do', 'few', 'o', 'so', 'further', 've', 'm', 'is', 'are', 'shouldn', 'will', 'myself', 'don', 'she', 'own', 'whi
le', "you're", 'such', 'there', 'and', 'won', 'very', 'before', 'himself', 're', "doesn't", "should've", 'should', 'your', 'because', 'no', '
y', 'you', 'doing', 'aren', 'them', 'but', 'any', 'these', 'couldn', 'itself', "won't", 'am', 'its', 'that', 'down', 't', 'wasn', 'under', 'b
een', 'below', 'more', "mightn't", 'then', 'by', "that'll", 'd', 'were', "shouldn't", 'my', 'if', "mustn't", 'until', 'a', 'who', 'some', 'on
', 'after', 'whom', 'when', "weren't", "didn't", 'theirs', 'has', 'during', 'he', 'they', 'to', 'hasn', 'or', 'which', 'above', 'only', 'your
s', 'at', 'over', 'weren', 'wouldn', 'as', 'again', "needn't", 'than', 'have', 'be', 'had', "aren't", "wasn't", 'yourselves', 'his', 'now', '
their', "couldn't", 'shan', 'just', "you'd", 'mustn', 'with', 'ourselves', 'our', 'too', 'the', 'we', 'did', 'in', 'once', 'off', 'mightn', '
hers', 'for', 'not', "it's", 'ain', 'ma', "hasn't", 'up', 'herself', 'between', 'into', 'nor', 'an', 'themselves', 'her', 'was', 'of', "don't
", 'what', 's', "you'll", 'through', 'both', 'll', 'most', 'ours', 'all'}
Filtered words= ['let', 'faith', 'bigger', 'fears', '.']
>>>
```

**PROGRAM**

```
#ex4
import nltk
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
ps = PorterStemmer()
sentence = "cars, drawing, played"
words = word_tokenize(sentence)
for word in words:
        print(word,":",ps.stem(word))
```

**OUTPUT**

```
Python 3.8.10 (default, May 26 2023, 14:05:08)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=============== RESTART: /home/student/Jerald/nlp/ex4/tokenize.py ==============
cars : car
, : ,
drawing : draw
, : ,
played : play
>>>
```

**PROGRAM**

```
#ex5
#word n-grams
from nltk import ngrams
sentence = "This is my sentence and I want to ngramize it."
n = 6
w_6grams = ngrams(sentence.split(), n)
for grams in w_6grams:
      print(grams)



#character n-grams
from nltk import ngrams
sentence = "This is my sentence and I want to ngramize it."
n = 6
c_6grams = ngrams(sentence, n)
for grams in c_6grams:
      print(grams)
```

## OUTPUT

```
Python 3.8.10 (default, May 26 2023, 14:05:08)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============= RESTART: /home/student/Jerald/nlp/ex5/word_n-grams.py ============
('This', 'is', 'my', 'sentence', 'and', 'I')
('is', 'my', 'sentence', 'and', 'I', 'want')
('my', 'sentence', 'and', 'I', 'want', 'to')
('sentence', 'and', 'I', 'want', 'to', 'ngramize')
('and', 'I', 'want', 'to', 'ngramize', 'it.')
>>>
```

```
Python 3.8.10 (default, May 26 2023, 14:05:08)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: /home/student/Jerald/nlp/ex5/character_n-grams.py ==========
('T', 'h', 'i', 's', ' ', 'i')
('h', 'i', 's', ' ', 'i', 's')
('i', 's', ' ', 'i', 's', ' ')
('s', ' ', 'i', 's', ' ', 'm')
(' ', 'i', 's', ' ', 'm', 'y')
('i', 's', ' ', 'm', 'y', ' ')
('s', ' ', 'm', 'y', ' ', 's')
(' ', 'm', 'y', ' ', 's', 'e')
('m', 'y', ' ', 's', 'e', 'n')
('y', ' ', 's', 'e', 'n', 't')
(' ', 's', 'e', 'n', 't', 'e')
('s', 'e', 'n', 't', 'e', 'n')
('e', 'n', 't', 'e', 'n', 'c')
('n', 't', 'e', 'n', 'c', 'e')
('t', 'e', 'n', 'c', 'e', ' ')
('e', 'n', 'c', 'e', ' ', 'a')
('n', 'c', 'e', ' ', 'a', 'n')
('c', 'e', ' ', 'a', 'n', 'd')
('e', ' ', 'a', 'n', 'd', ' ')
(' ', 'a', 'n', 'd', ' ', 'I')
('a', 'n', 'd', ' ', 'I', ' ')
('n', 'd', ' ', 'I', ' ', 'w')
('d', ' ', 'I', ' ', 'w', 'a')
(' ', 'I', ' ', 'w', 'a', 'n')
('I', ' ', 'w', 'a', 'n', 't')
(' ', 'w', 'a', 'n', 't', ' ')
('w', 'a', 'n', 't', ' ', 't')
('a', 'n', 't', ' ', 't', 'o')
('n', 't', ' ', 't', 'o', ' ')
('t', ' ', 't', 'o', ' ', 'n')
(' ', 't', 'o', ' ', 'n', 'g')
```

**PROGRAM**

```
#ex4
#n-grams smoothing
from collections import defaultdict
def calculate_ngram_probabilities(corpus):
	ngrams = defaultdict(int)
	context = defaultdict(int)
	for sentence in corpus:
		words = sentence.split()
		for i in range(len(words) - 2):
			trigram = tuple(words[i:i+3])
			ngrams[trigram] += 1
			context[trigram[:2]] += 1
		print(ngrams)
		print("------------------------------------------------------------")
		print(context)
		probabilities = defaultdict(float)
	for trigram, count in ngrams.items():
		context_count = context[trigram[:2]]
		probabilities[trigram] = (count + 1)/(context_count +
len(ngrams))
		return probabilities
corpus = [
	"I love to code",
	"Python is a popular programming language",
	"Coding is fun",
	"I enjoy coding in Python",
	"I love to dance"
]
trigram_probabilities = calculate_ngram_probabilities(corpus)
for trigram, probability in trigram_probabilities.items():
	print(f"Trigram: {trigram}, Probability: {probability:.4f}")
```

# OUTPUT

```
Python 3.8.10 (default, May 26 2023, 14:05:08)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: /home/student/Jerald/nlp/ex6/n-grams_smoothing.py ==========
defaultdict(<class 'int'>, {('I', 'love', 'to'): 1, ('love', 'to', 'code'): 1})
---------------------------------------------------------
defaultdict(<class 'int'>, {('I', 'love'): 1, ('love', 'to'): 1})
defaultdict(<class 'int'>, {('I', 'love', 'to'): 1, ('love', 'to', 'code'): 1, ('Python', 'is', 'a'): 1, ('is', 'a', 'popular'): 1, ('a', 'po
pular', 'programming'): 1, ('popular', 'programming', 'language'): 1})
---------------------------------------------------------
defaultdict(<class 'int'>, {('I', 'love'): 1, ('love', 'to'): 1, ('Python', 'is'): 1, ('is', 'a'): 1, ('a', 'popular'): 1, ('popular', 'progr
amming'): 1})
defaultdict(<class 'int'>, {('I', 'love', 'to'): 1, ('love', 'to', 'code'): 1, ('Python', 'is', 'a'): 1, ('is', 'a', 'popular'): 1, ('a', 'po
pular', 'programming'): 1, ('popular', 'programming', 'language'): 1, ('Coding', 'is', 'fun'): 1})
---------------------------------------------------------
defaultdict(<class 'int'>, {('I', 'love'): 1, ('love', 'to'): 1, ('Python', 'is'): 1, ('is', 'a'): 1, ('a', 'popular'): 1, ('popular', 'progr
amming'): 1, ('Coding', 'is'): 1})
defaultdict(<class 'int'>, {('I', 'love', 'to'): 1, ('love', 'to', 'code'): 1, ('Python', 'is', 'a'): 1, ('is', 'a', 'popular'): 1, ('a', 'po
pular', 'programming'): 1, ('popular', 'programming', 'language'): 1, ('Coding', 'is', 'fun'): 1, ('I', 'enjoy', 'coding'): 1, ('enjoy', 'cod
ing', 'in'): 1, ('coding', 'in', 'Python'): 1})
---------------------------------------------------------
defaultdict(<class 'int'>, {('I', 'love'): 1, ('love', 'to'): 1, ('Python', 'is'): 1, ('is', 'a'): 1, ('a', 'popular'): 1, ('popular', 'progr
amming'): 1, ('Coding', 'is'): 1, ('I', 'enjoy'): 1, ('enjoy', 'coding'): 1, ('coding', 'in'): 1})
defaultdict(<class 'int'>, {('I', 'love', 'to'): 2, ('love', 'to', 'code'): 1, ('Python', 'is', 'a'): 1, ('is', 'a', 'popular'): 1, ('a', 'po
pular', 'programming'): 1, ('popular', 'programming', 'language'): 1, ('Coding', 'is', 'fun'): 1, ('I', 'enjoy', 'coding'): 1, ('enjoy', 'cod
ing', 'in'): 1, ('coding', 'in', 'Python'): 1, ('love', 'to', 'dance'): 1})
---------------------------------------------------------
defaultdict(<class 'int'>, {('I', 'love'): 2, ('love', 'to'): 2, ('Python', 'is'): 1, ('is', 'a'): 1, ('a', 'popular'): 1, ('popular', 'progr
amming'): 1, ('Coding', 'is'): 1, ('I', 'enjoy'): 1, ('enjoy', 'coding'): 1, ('coding', 'in'): 1})
Trigram: ('I', 'love', 'to'), Probability: 0.2308
>>>
```

**PROGRAM**

```
#ex7
#pos_tags
import nltk
from nltk import word_tokenize
sentence = "I am learning NLP in Python"
tokens = nltk.word_tokenize(sentence)
pos_tags = nltk.pos_tag(tokens)
print(pos_tags)
```

**OUTPUT**

```
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on w
in32
Type "help", "copyright", "credits" or "license()" for more information.

================== RESTART: C:\Users\admin\Downloads\pos_tags.py ================
[('I', 'PRP'), ('am', 'VBP'), ('learning', 'VBG'), ('NLP', 'NNP'), ('in', 'IN'), ('Python',
'NNP')]
```

**PROGRAM**

```
#ex8

#chunks

import nltk

sentence = "The clever fox escaped from the lion"

tokens = nltk.word_tokenize(sentence)

pos_tags = nltk.pos_tag(tokens)

grammar = "NP: {<DT>?<JJ>*<NN>}"

chunk_parser = nltk.RegexpParser(grammar)

chunks = chunk_parser.parse(pos_tags)

print(chunks)

chunks.draw()
```

**OUTPUT**

```
Python 3.8.10 (default, May 26 2023, 14:05:08)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================ RESTART: /home/student/Jerald/nlp/ex8/chunks.py ===============
(S
  (NP The/DT clever/NN)
  (NP fox/NN)
  escaped/VBD
  from/IN
  (NP the/DT lion/NN))
```

**PROGRAM**

```
#ex9i
#cosine_similarity
import re
from collections import Counter
import math
def calculate_cosine_similarity(text1, text2):
    def tokenize(text):
        words=re.findall(r'\w+',text.lower())
        return Counter(words)
    vec1=tokenize(text1)
    vec2=tokenize(text2)
    intersection=set(vec1.keys()) & set(vec2.keys())
    dot_product=sum(vec1[word]*vec2[word] for word in intersection)
    magnitude1=math.sqrt(sum(vec1[word] ** 2 for word in vec1.keys()))
    magnitude2=math.sqrt(sum(vec2[word] ** 2 for word in vec2.keys()))
    cos_similarity=dot_product / (magnitude1*magnitude2)
    return cos_similarity

text1="I am Jerald"
text2="I am from Sivakasi"
similarity=calculate_cosine_similarity(text1, text2)
print("cosine similarity:", similarity)
```

**PROGRAM**

```
#ex9ii
#jaccard_similarity
import re
def calculate_jaccard_similarity(text1,text2):
    def tokenize(text):
        words = re.findall(r'\w+', text.lower())
        return set(words)

    set1=tokenize(text1)
    set2=tokenize(text2)

    intersection = len(set1.intersection(set2))
    union = len(set1.union(set2))
    jaccard_similarity=intersection/union
    return jaccard_similarity

text1="I am Jerald"
text2="I am in Sivakasi"
similarity=calculate_jaccard_similarity(text1, text2)
print("Jaccard Similarity:", similarity)
```

**OUTPUT**

```
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct  5 2020, 15:34:40) [MSC v.1927 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============ RESTART: C:/Users/Best/Downloads/cousine_similarity.py ============
cosine similarity: 0.5773502691896258
>>>
```

**OUTPUT**

```
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct  5 2020, 15:34:40) [MSC v.1927 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: F:/STUDY MATERIALS/SEMESTER 7/Lab/NLP Lab/ex9b/jaccard_similarity.py
Jaccard Similarity: 0.4
>>>
```

**PROGRAM**

```
#ex9i
#bert_similarity
import sentence_transformers
from sentence_transformers import SentenceTransformer


from sklearn.metrics.pairwise import cosine_similarity
def calculate_bert_similarity(text1, text2):
    # Load the pre-trained BERT model
    model = SentenceTransformer('bert-base-uncased')


    # Encode the texts into BERT embeddings
    embedding1 = model.encode([text1])[0]
    embedding2 = model.encode([text2])[0]


    # Calculate cosine similarity between the embeddings
    similarity = cosine_similarity(embedding1.reshape(1, -1),
embedding2.reshape(1, -1))


    return similarity[0][0]


# Example usage
text1 = "I like apples"
text2 = "I love apples"
similarity = calculate_bert_similarity(text1, text2)
print("BERT Similarity:", similarity)
```

**PROGRAM**

```
#ex9ii
#roberta_similarity
import sentence_transformers
from sentence_transformers import SentenceTransformer


from sklearn.metrics.pairwise import cosine_similarity
def calculate_bert_similarity(text1, text2):
    # Load the pre-trained BERT model
    model = SentenceTransformer('roberta-base-nli-mean-tokens')


    # Encode the texts into BERT embeddings
    embedding1 = model.encode([text1])[0]
    embedding2 = model.encode([text2])[0]


    # Calculate cosine similarity between the embeddings
    similarity = cosine_similarity(embedding1.reshape(1, -1),
embedding2.reshape(1, -1))


    return similarity[0][0]


# Example usage
text1 = "I like apples"
text2 = "I love apples"
similarity = calculate_bert_similarity(text1, text2)
print("RoBERTa Similarity:", similarity)
```

**OUTPUT**

```
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct  5 2020, 15:34:40) [MSC v.1927 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: F:\STUDY MATERIALS\SEMESTER 7\Lab\NLP Lab\ex9b\sentence_transform_i.p
y
No sentence-transformers model found with name C:\Users\Best/.cache\torch\senten
ce_transformers\bert-base-uncased. Creating a new one with MEAN pooling.
BERT Similarity: 0.89023733
>>>
```

**OUTPUT**

```
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct  5 2020, 15:34:40) [MSC v.1927 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: F:\STUDY MATERIALS\SEMESTER 7\Lab\NLP Lab\ex9b\sentence_transform_ii.
py
RoBERTa Similarity: 0.9577409
>>>
```

**PROGRAM**

```
#ex10

#lesk_algorithm

from pywsd.lesk import simple_lesk

sentences = ['I went to the bank to deposit my money','The river bank was
full of dead fishes']

# calling the lesk function and printing results for both the sentences

print ("Context-1:", sentences[0])

answer = simple_lesk(sentences[0],'bank')

print ("Sense:", answer)

print ("Definition : ", answer.definition())

print ("Context-2:", sentences[1])

answer = simple_lesk(sentences[1],'bank')

print ("Sense:", answer)

print ("Definition : ", answer.definition())
```

**OUTPUT**

```
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct  5 2020, 15:34:40) [MSC v.1927 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=== RESTART: F:\STUDY MATERIALS\SEMESTER 7\Lab\NLP Lab\ex10\lesk_algorithm.py ==
Warming up PyWSD (takes ~10 secs)... took 5.176192045211792 secs.
Context-1: I went to the bank to deposit my money
Sense: Synset('depository_financial_institution.n.01')
Definition :  a financial institution that accepts deposits and channels the mon
ey into lending activities
Context-2: The river bank was full of dead fishes
Sense: Synset('bank.n.01')
Definition :  sloping land (especially the slope beside a body of water)
>>>
```