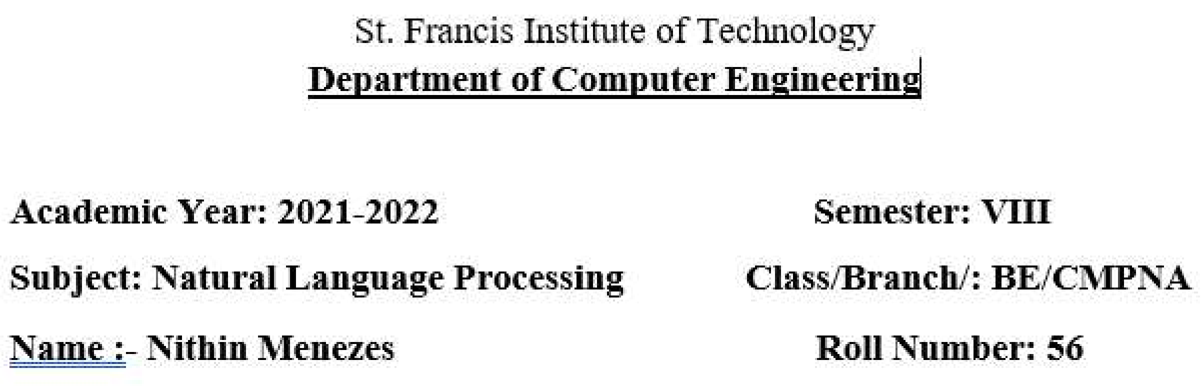

St. Francis Institute of Technology

Department of Computer Engineering

Academic Year: 2021-2022 **Semester: VIII**

Subject: Natural Language Processing **Class/Branch/: BE/CMPNA**

Name :- Nithin Menezes **Roll Number: 56**

Double-click (or enter) to edit

Name: Nithin Menezes

Roll no:56

PID: 182066
BECMPNA

Experiment No: 4

Aim : To implement the Ngram model from a text corpus and do adjacent word prediction in Python

Saved successfully!

```
corpus = '''the fruit is red the fruit is green the fruit is blue the rock is brown.
Nithin is a student.'''


# Writing the corpus into a text file
with open('corpus.txt', 'w+') as f:
    f.write(corpus)


import re
from pprint import pprint


def _conditional_prob(s, n):
    return float(s.count(f'{n[0]} {n[1]}') / s.count(n[0]))


def get_conditional_prob(s, ngrams):
    conditional_prob = {}
    for ngram in ngrams:
        conditional_prob[ngram] = _conditional_prob(s, ngram)
    return conditional_prob

def tokenize(corpus):
    return [token for token in corpus.split(' ') if token != '']


def get_n_grams(s, tokens, n):
    s = s.lower()

    # Replace all none alphanumeric characters with spaces
    s = re.sub(r'[^a-zA-Z0-9\s]', ' ', s)

    # Use the zip function to help us generate n-grams
    # Concatentate the tokens into ngrams and return
    ngrams = zip(*[tokens[i:] for i in range(n)])
    # return [" ".join(ngram) for ngram in ngrams]
    return list(set([ngram for ngram in ngrams]))


def predict_next_word(s, conditional_prob, tokens, word):
    predictions = {}
    for token in tokens:
        n = tuple([word, token])
        predictions[n] = _conditional_prob(s, n)
    return predictions


with open(r'corpus.txt', 'r') as f:
    all_lines = f.readlines()
corpus = ''.join([x.replace('\n', ' ') for x in all_lines])
```

Saved successfully! ×

```
for line in all_lines:
    line = line.replace('\n', '')
    tkns = tokenize(line)
    for t in tkns:
        tokens.append(t)
    n_gram = get_n_grams(line, tokens,2)
    for n in n_gram:
        n_grams.append(n)

conditional_prob = get_conditional_prob(corpus, n_grams)
print('Conditional Probabilities: '); pprint(conditional_prob)
word = input('Enter a word: ')
predictions = predict_next_word(corpus, conditional_prob, tokens, word)
print('All predictions with probability are: ')
pprint(predictions)
```

```
Conditional Probabilities:
{('Nithin', 'is'): 1.0,
 ('a', 'student.'): 1.0,
 ('blue', 'the'): 1.0,
      ('brown.', 'Nithin'): 1.0,
      ('fruit', 'is'): 1.0,
      ('green', 'the'): 1.0,
      ('is', 'a'): 0.2,
      ('is', 'blue'): 0.2,
      ('is', 'brown.'): 0.2,
      ('is', 'green'): 0.2,
      ('is', 'red'): 0.2,
      ('red', 'the'): 1.0,
      ('rock', 'is'): 1.0,
      ('the', 'fruit'): 0.75,
      ('the', 'rock'): 0.25}
     Enter a word: fruit
     All predictions with probability are:
     {('fruit', 'Nithin'): 0.0,
      ('fruit', 'a'): 0.0,
      ('fruit', 'blue'): 0.0,
      ('fruit', 'brown.'): 0.0,
      ('fruit', 'fruit'): 0.0,
      ('fruit', 'green'): 0.0,
      ('fruit', 'is'): 1.0,
      ('fruit', 'red'): 0.0,
      ('fruit', 'rock'): 0.0,
      ('fruit', 'student.'): 0.0,
      ('fruit', 'the'): 0.0}
```

```
word = input('Enter a word: ')
predictions = predict_next_word(corpus, conditional_prob, tokens, word)
print('All predictions with probability are: ')
pprint(predictions)
```

Saved successfully! ×

```
                                  y are:

      ('Nithin', 'a'): 0.0,
      ('Nithin', 'blue'): 0.0,
      ('Nithin', 'brown.'): 0.0,
      ('Nithin', 'fruit'): 0.0,
      ('Nithin', 'green'): 0.0,
      ('Nithin', 'is'): 1.0,
      ('Nithin', 'red'): 0.0,
      ('Nithin', 'rock'): 0.0,
      ('Nithin', 'student.'): 0.0,
      ('Nithin', 'the'): 0.0}
```

Trigram

```
def trigram_conditional_prob(s, n):
    return float(s.count(f'{n[0]} {n[1]} {n[2]}') / s.count(f'{n[0]} {n[1]}'))


def get_conditional_prob_tri(s, ngrams):
    conditional_prob = {}
    for ngram in ngrams:
        conditional_prob[ngram] = trigram_conditional_prob(s, ngram)
    return conditional_prob

def predict_next_word_tri(s, conditional_prob, tokens, word):
    predictions = {}
    for token in tokens:
        n = tuple([word[0],word[1], token])
        predictions[n] = trigram_conditional_prob(s, n)
    return predictions


with open(r'corpus.txt', 'r') as f:
    all_lines = f.readlines()
corpus = ''.join([x.replace('\n', ' ') for x in all_lines])
n_grams = []
tokens = []
for line in all_lines:
    line = line.replace('\n', '')
    tkns = tokenize(line)
    for t in tkns:
        tokens.append(t)
    n_gram = get_n_grams(line, tokens,3)
    for n in n_gram:
        n_grams.append(n)

conditional_prob = get_conditional_prob_tri(corpus, n_grams)
print('Conditional Probabilities: '); pprint(conditional_prob)
word = input('Enter two words to input ').split(" ")
                                          corpus, conditional_prob, tokens, word)
                                          ity are: ')
pprint(predictions)
```

```
Conditional Probabilities:
{('Nithin', 'is', 'a'): 1.0,
 ('blue', 'the', 'rock'): 1.0,
 ('brown.', 'Nithin', 'is'): 1.0,
 ('fruit', 'is', 'blue'): 0.3333333333333333,
 ('fruit', 'is', 'green'): 0.3333333333333333,
 ('fruit', 'is', 'red'): 0.3333333333333333,
 ('green', 'the', 'fruit'): 1.0,
 ('is', 'a', 'student.'): 1.0,
 ('is', 'blue', 'the'): 1.0,
 ('is', 'brown.', 'Nithin'): 1.0,
 ('is', 'green', 'the'): 1.0,
 ('is', 'red', 'the'): 1.0,
 ('red', 'the', 'fruit'): 1.0,
 ('rock', 'is', 'brown.'): 1.0,
 ('the', 'fruit', 'is'): 1.0,
 ('the', 'rock', 'is'): 1.0}
Enter two words to input the rock
All predictions with probability are:
```

```
{('the', 'rock', 'Nithin'): 0.0,
 ('the', 'rock', 'a'): 0.0,
 ('the', 'rock', 'blue'): 0.0,
 ('the', 'rock', 'brown.'): 0.0,
 ('the', 'rock', 'fruit'): 0.0,
 ('the', 'rock', 'green'): 0.0,
 ('the', 'rock', 'is'): 1.0,
 ('the', 'rock', 'red'): 0.0,
 ('the', 'rock', 'rock'): 0.0,
 ('the', 'rock', 'student.'): 0.0,
 ('the', 'rock', 'the'): 0.0}
```

Saved successfully!