	Bansilal Ramnath Agarwal Charitable Trust's Vishwakarma Institute of Information Technology **Department of** **Artificial Intelligence and Data Science**		
Name: **Siddhesh Dilip Khairnar**			
Class: **TY**	Division: **B**		Roll No: **372028**
Semester: **6th**		Academic Year: **2023-24**	
Subject Name & Code: **Natural Language and Processing & ADUA32203**			
Title of Assignment: **Develop a language model to predict next best word.**			
Date of Performance: **06-03-2024**		Date of Submission: **13-03-2024**	

ASSIGNMENT NO: - 8

Aim: Develop a language model to predict the next best word.

Background Theory:

Language modeling is the process of predicting the probability distribution of words in a sequence of text, often used in natural language processing (NLP) and speech recognition. The goal of language modeling is to estimate the likelihood of a sequence of words. The most common approach to language modeling is to use a statistical model that is trained on a large corpus of text.

Language Model Type	***Description***	***Advantages***	***Disadvantages***
N-Gram Model	Predicts the probability of the next word based on the previous N-1 words in the sentence.	Simple and easy to implement, computationaly efficient	Unable to capture long-range dependencies
Neural Network Model	Uses a neural network to predict the probability of the next word based	Can capture long- range dependencies, can	Computationally expensive,
	on the previous words in the sentence.	handle out-of-vocabulary words	requires large amounts of data
Transformer Language Model	Uses a transformer network to predict the probability of the next word based on the previous words in the sentence	Can capture long- range dependencies, can handle out of-vocabulary words, can be trained faster than traditional neural networks	Computationally expensive, requires large amounts of data, difficult to interpret

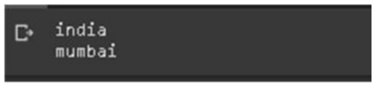
Bayesian Model	Uses Bayesian inference to predict the probability of the next word based on the previous words in the sentence.	Can handle uncertainty and ambiguity, can be updated with new data	Computationally expensive, difficult to implement
Rulebased Model	Uses a set of rules to generate sentences.	Can generate sentences based on specific criteria, can be used to check grammar and syntax	Limited to predefined rules, unable to capture context and meaning

```
import re
from    collections    import
Counter  from  math  import
log import nltk


corpus = """
Mumbai also known as Bombay /bɒmˈbeɪ/— the official name until 1995) is the capital city of the Indian s
tate of Maharashtra.
Mumbai is the de facto financial centre and the most populous city of India with an estimated city proper
p opulation of 12.5 million (1.25 crore).
Mumbai is the centre of Mumbai Metropolitan Region, the sixth most populous metropolitan area in the
world with a population of over 23 million (2.3 crore) livi ng under the Brihanmumbai Municipal
Corporation.
Mumbai lies on the Konkan coast on the west coast of India and has a deep natural harbour. In 2008, Mum
bai was named an alpha world city.
"""


corpus  =  corpus.lower()
corpus = re.sub(r'[^\w\s]', '',
corpus) words =
nltk.word_tokenize(corpus)

unigrams = Counter(words)
bigrams = Counter(zip(words, words[1:]))
trigrams = Counter(zip(words, words[1:], words[2:]))

vocab_size = len(set(words)) smoothed_bigrams = {}
for  bigram  in  bigrams:  word1  =  bigram[0]  prob  =
(bigrams[bigram]    +    1)    /    (unigrams[word1]    +
vocab_size) smoothed_bigrams[bigram] = prob

smoothed_trigrams = {} for
trigram in trigrams: bigram =
(trigram[0], trigram[1])
    prob = (trigrams[trigram] + 1) / (bigrams[bigram] + vocab_size)
    smoothed_trigrams[trigram] = prob

def predict_next_word(previous_words, n=3): previous_words
    = previous_words.lower()
    previous_words = re.sub(r'[^\w\s]', '', previous_words)
```

Code:

```
    if n == 2: probs = [(word, smoothed_bigrams.get((words[-1], word), 1/vocab_size)) for word in
        unigrams]
    elif n == 3:
        probs = [(word, smoothed_trigrams.get((words[-2], words[-
1], word), 1/vocab_size)) for word in unigrams]
    else:
        return "n can only be 2 or 3."
    probs.sort(key=lambda             x:             x[1],
    reverse=True) return probs[0][0]


print(predict_next_word("West          Coast
of"))        print(predict_next_word("centre
of"))
```

Output:



```
india
mumbai
```

Conclusion: In the above assignment, we successfully implemented basic next work prediction model using unigram, bigram, trigram then smoothing them and find out next based on input. Able to get basic idea about how next word prediction mechanism take place in action and predict next word correctly. Able to grasp concepts regarding it very well.