

Advanced Natural Language Processing

Practical Report



Student Name: Sagar Maheshwari

Student ID: 251043

Table of Contents

Introduction	2
Word Prediction using the technology of the neural language model	3
Word prediction using an N-grams model	8
Conclusion	13
Bibliography	14
Appendices	15

Table of figures

Figure 1: Library of python which is used in the Neural Programming Model	4
Figure 2: Sequence generating function	5
Figure 3: Size of vocabulary	6
Figure 4: final output	7
Figure 5: Python library used for N-grams method	8
Figure 6: word count function	9
Figure 7: Function returns the count of dictionary value	10
Figure 8: main function	11
Figure 9: Final part	12

Introduction:-

In the modern world, there is a very broad role for electronic writing so Microsoft designed a sentence completion challenge for designing a system that can predict the upcoming word during the writing in order to complete the sentence. This challenge was designed in 2011 for developing a sense of programming among the students. This model can be performed with the help of various models using the python language. The report discussed the two technologies which are helpful in completing the task. The first one is the neural language model and the other one is the n-gram method. In these two methods, various python mathematical tools and libraries are used. Python Libraries which are used in completing the task are matplotlib, NumPy, Keras, random, TensorFlow, and counter. Two methods are described by measuring the several steps. In order to perform tasks, machine learning and data science properties are used for the analysis and visualisation of the data. A mathematical tool such as sequential, tokenizer, and programming logic such as embedded python is used for predicting the word or sentence in order to complete the sentence. As a python code, input data is taken in the form of docstring and then docstring is split and gives the output according to the randomly generated number. It can be done by using an external text file by importing the OS module of python.

Word Prediction using the technology of the neural language model:-

The N-gram model deals in mathematical operations such as probability. This model predicts the N-gram according to the sequence of any specific word. The N-gram model predicts the value according to the sequence of the word and predicts the next word and sentence. It is the easiest concept to understand the process of machine learning, it is defined as the sequence of a series of words. This model can be used for the checking spelling of a word, can be able to identify the sequencing of DNA, and generally, it is used in the natural processing of languages. Because natural language such as English follows the repetition of words and characters in the sequence. A problem statement is to develop a code that predicts the word for completion of the sentence. The language model is defined as the key model of a larger model for the programming languages. Word prediction of the technology is understood by using the 10 steps (Wang et al. 2019).

The first step in writing the code is to import the python libraries. Python is a high-level, interpreted, structural language that is used in various departments like machine learning, deep learning, data science, and designing the framework with help of a python inbuilt library. For mathematical operations, the NumPy library is majorly used in the domain. There is a need for a number of libraries in order to develop the python code of next word prediction with the help of a neural language model. The first library is Matplotlib to analyse the data by measuring the graph and the probability. Matplotlib is a platform by which graphical plotting and data visualisation can occur. Matplotlib is the numerical extension of the NumPy library which is majorly used for data science. Developers also use matplotlib for the application programming interface. A python-based array is also used in the model; it is a collection of the unique data type. Array stores the data; an array is not a direct data type of the python. In order to use the array in python, developers have to import the python module as arr. Another majorly used library in the code is Keras which is an open-source and free platform for developing the model of deep learning. Keras also performs numerical operations by the use of sub-libraries such as TensorFlow and Theano. Keras is a powerful library because it allows a large number of data to be performed in the python language. In order to break the raw data into small chunks, tokenization is used; tokenization is the base for the neural programming language model. It also gives an understanding of the development of the model of NLP and proposes an overview to write the program. Tokenization works for identifying the next word by analysing the sequence

of the last word. It is used for splitting the larger body into the smaller text line by using the syntax of python. It can even break words and also can create words. A sub-library of Keras, Embedding is also used for implementing the functionality of the application by using python. Python file extensions are (.py and .ipynb). It is useful writing the script in python in an efficient way. There is a need to give a large input to the machine learning operation in the form of sparse vectors; embedding makes them easier. This model considers the conversation between the two people in the form of data. The responsibility of the code is to identify the next word by the written word as a prediction. The complete model of the numeral programming model is understood below by using the python script. For writing the code, visual studio code is used and output is performed on the git-bash terminal (W3schools.com 2022).

```
from numpy import array, argmax
from keras.preprocessing.text import Tokenizer
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Embedding
```

Figure 1: Library of python which is used in the Neural Programming Model

(Source: Author 2022)

Step 1: The first step describes the sequence generation by declaring the function. The function takes the attribute in the form of a tokenizer and user-defined variables such as seed_text and n_words. In python, some action is defined by using the function like if the user wants to perform additional functions then the user is able to define the functionality of the addition by using the def keyword. Def keyword stands for the definition. The function is defined by any name; if the user passes the two values in the function then it will return the addition of the two numbers. A sequence is defined as the list of the elements with a particular defined order; it is useful for performing mathematical operations for the study of spaces, functions, and other mathematical structures using the property of convergence. In the model, the first step defines the generating sequence action. A for loop is applied in the function after declaring the two variables such as in_text and result. Two variables are declared in the for loop in which one is attached with the tokenizer and array which were described earlier. Tokenizer evaluates the text

as an integer and the model is used for the production of words in the vocabulary, then the print function gives the output of the predicted word. Another for loop is described in the first for loop for performing the class of mapping. The map is a predefined function in python that is used for defining the position of a variable in the memory. If the statement is declared in the second loop for testing the condition, if the defined condition is true then the loop is broken; otherwise the function gives the output in the form of a result. The result is declared as the appending of the input; appending is the process of adding the items in the list or any type of data type (Programiz.com 2022).

```
def generate_seq(model, tokenizer, seed_text, n_words):
    in_text, result = seed_text, seed_text
    # generate a fixed number of words
    for _ in range(n_words):
        # encode the text as integer
        encoded = tokenizer.texts_to_sequences([in_text])[0]
        encoded = array(encoded)
        # predict a word in the vocabulary
        a = model.predict(encoded, verbose=0)
        yhat = argmax(a,axis=1)
        print('ythat', yhat)
        # map predicted word index to word
        out_word = ''
        for word, index in tokenizer.word_index.items():
            if index == yhat:
                out_word = word
                break
        # append to input
        in_text, result = out_word, result + ' ' + out_word
    return result
```

Figure 2: Sequence generating function

(Source: Author 2022)

Step 2: Step two defines the source of the text which was taken as the docstring and declared in the variable named “data”. Docstring is denoted by triple quotes; here, multiple statements are defined. The use of docstring is defined as a document of a module of python, classes, function, and method. After the first step, Tokenizer is defined as a function of the tokenizer, then a

docstring is attached by using the indexing. For this scenario a predefined function `fit_on_text` is used for attaching the data to the tokenizer. Tokenize splits the data in small forms. Another property of python is attached to the tokenizer by which texts are broken in the form of a proper sequence. It will be started from the 0th index and run at the place of the end position which is declared in the “encoded” variable. For determining the size of vocabulary, the `len` function is used. `len` is a predefined length function that calculates the length of any string and other types of data types. And a value is increased for running the loop. Generated sequence is converted into the list to store the data. The list is a mutable data type that can contain all data types like integer, string, tuple, set, and array. The print function size of the vocabulary comes as an output (Simplilearn 2021).

```
# integer encode text|
tokenizer = Tokenizer()
tokenizer.fit_on_texts([data])
encoded = tokenizer.texts_to_sequences([data])[0]
# determine the vocabulary size
vocab_size = len(tokenizer.word_index) + 1
print('Vocabulary Size: %d' % vocab_size)
# create word -> word sequences
sequences = list()
```

Figure 3: Size of vocabulary

(Source: Author 2022)

A loop is applied in the range from 1 to the length of the variable “encoded”. The range function defines the repetition of the for a loop. And in the for loop sequence is described as the variable encoded with the index from $i-1$ to $i+1$. That sequence is appended in the newly designed list sequences. The for loop gives the output in the form of the length of newly generated list sequences. Total sequences are printed in the form of `%d` and It is a format of python2 and earlier versions of python3. Sequences which are the data type of the list are now converted in the array. Array stores only one type of data and then values of x and y are split as the index of respectively 0 and 1. The index number of zero values is assigned in the variable x and then the index number of 1 in the sequence is assigned in the variable y . `to_categorical` is used for converting the class vector which is the part of the binary class that is converted into the matrix of the binary class. The value of y and vocab size are assigned to the y as a form of the matrix. After that the actual

model of the neural programming language is defined. Then the property of embedding is applied to the size of the model by using the property of add. Embedding is used by defining the length of the input in the next line property of long-short ten Memory. Then the summary of the model appears as a form of output by using the print function. Then the work of the model is to compile all networks which are designed by the developer by using the loss, optimizer, and metrics. Then the final result is printed in the sequence of the model, tokenizer, name, and an integer (GeeksforGeeks 2018).

```
for i in range(1, len(encoded)):
    sequence = encoded[i-1:i+1]
    sequences.append(sequence)
print('Total Sequences: %d' % len(sequences))
# split into X and y elements
sequences = array(sequences)
X, y = sequences[:, 0], sequences[:, 1]
# one hot encode outputs
y = to_categorical(y, num_classes=vocab_size)
# define model
model = Sequential()
model.add(Embedding(vocab_size, 10, input_length=1))
model.add(LSTM(50))
model.add(Dense(vocab_size, activation='softmax'))
print(model.summary())
# compile network
model.compile(loss='categorical_crossentropy',
              optimizer='adam', metrics=['accuracy'])
# fit network
model.fit(X, y, epochs=500, verbose=2)
# evaluate
print(generate_seq(model, tokenizer, 'Jack', 6))
```

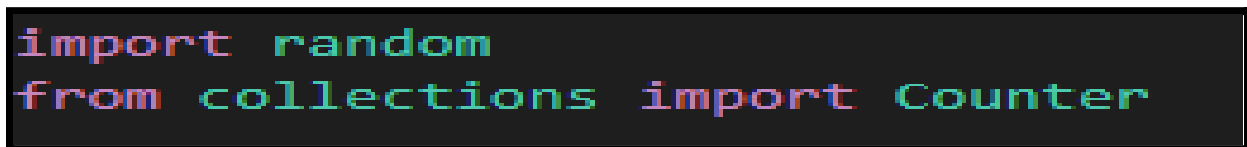
Figure 4: final output

(Source: Author 2022).

```
1/1 - 0s - loss: 0.2326 - accuracy: 0.8750 - 0s/epoch - 0s/step
Epoch 491/500
1/1 - 0s - loss: 0.2324 - accuracy: 0.8750 - 0s/epoch - 0s/step
Epoch 492/500
1/1 - 0s - loss: 0.2321 - accuracy: 0.8750 - 8ms/epoch - 8ms/step
Epoch 493/500
1/1 - 0s - loss: 0.2319 - accuracy: 0.8750 - 8ms/epoch - 8ms/step
Epoch 494/500
1/1 - 0s - loss: 0.2316 - accuracy: 0.8750 - 8ms/epoch - 8ms/step
Epoch 495/500
1/1 - 0s - loss: 0.2314 - accuracy: 0.8750 - 8ms/epoch - 8ms/step
Epoch 496/500
1/1 - 0s - loss: 0.2312 - accuracy: 0.8750 - 8ms/epoch - 8ms/step
Epoch 497/500
1/1 - 0s - loss: 0.2309 - accuracy: 0.8750 - 8ms/epoch - 8ms/step
Epoch 498/500
1/1 - 0s - loss: 0.2307 - accuracy: 0.8750 - 8ms/epoch - 8ms/step
Epoch 499/500
1/1 - 0s - loss: 0.2305 - accuracy: 0.8750 - 8ms/epoch - 8ms/step
Epoch 500/500
1/1 - 0s - loss: 0.2302 - accuracy: 0.8750 - 8ms/epoch - 8ms/step
ythat [1]
ythat [3]
ythat [4]
ythat [5]
ythat [6]
ythat [7]
Jack and jill went up the hill
```


Word prediction using an N-grams model:-

N-grams are defined as a sequence of continuous words, symbols, or tokens in the document. Sequences are defined as the neighbouring items in the document. N refers to the multiple like 1,2,3 and so on. The term n equal to one is defined as unigram; when n refers to two, then the term defined as, n equal to three is defined as trigram, and the term is greater than the three then the term refers to the n-gram. For developing the code of the word prediction system for completing the sentence with the help of the n-gram model there is a need for two libraries to import which are the random and counter. The random module is used for the generation of random numbers it is majorly used to design the number guessing game and for some surveys. The data generated by a random function is not random according to the scientific sense; it can be considered a pseudorandom number generator (PRNG). Because the algorithm generated the number randomly but the number as an output of reproducible data. Counter in python is used for holding the data in the unordered collection. The main use of counters is to operate add, subtraction, multiplication, union, and intersection. An iterable list is created for holding the data in a counter and is able to count the elements in another counter. The counter can be perfumed with string, list, dictionary, and the tuple. For initialising the counter there is a need to pass the value in terms of string, list, dictionary, or the tuple. The counter can be updated easily after initialising the by using the update function in the counter (Borisov 2020).

A screenshot of a code editor with a dark background. The text is written in a monospaced font with syntax highlighting: 'import random' is on the first line, and 'from collections import Counter' is on the second line. The word 'import' is highlighted in red, 'random' and 'Counter' are in green, and 'from collections' is in blue.

```
import random
from collections import Counter
```

Figure 5: Python library used for N-grams method

(Source: Admin 2022)

The first step in the N-grams method is to def a function that takes the argument as an array and sentence in order to define the frequency of the written statements. A blank list is defined inside the function names word_list and another variable splits the sentence and evaluates the length of the split sentence. This line also shows that two variables are declared in the same line and differentiated by the comma. After declaring these variables a for loop is worked in the range of zero to the length of the array. For-loop defines the code to execute for length +1 time. If the

length of the array is 5 then the code will run six times because indexing starts from zero in the python. If the for loop does not have a start argument manually then it will be run by the zeroth position. It is the default position (GeeksforGeeks 2021). After that, if the condition is used to check the length of the split sentence. If the condition denotes that the index of the word plus the length of the split sentence is less than the length of the array minus one then the array is appended to the blank list. The index of an array is defined as the addition of the word index and length of a split sentence. This frequency calculator function returns the counter of the word list in the form of a dictionary. Dictionary is the unordered collection of the key and value pair. Basically, this function returns the count of the words which were evaluated earlier (Borisov 2020).

```
# function calculate the freq of the corse by i+1
# were i is the index of nth words.

def next_word_freq(array, sentence):

    sen_len, word_list = len(sentence.split()), []

    for i in range(len(array)):

        if ' '.join(array[i : i + sen_len]).lower() == sentence.lower():

            if i + sen_len < len(array) - 1:

                word_list.append(array[i + sen_len])

    # this function return the count of the word

    return dict(Counter(word_list))
```

Figure 6: word count function

(Source: Author 2022)

In the next step, a function is defined for counting the CDF by the user of the counter dictionary. Then two variables are declared for performing the mathematical function. The first variable

assigned the value zero and another variable assigned the sum of the dictionary values. Then a for loop is executed on the key and value pair of the dictionary by using the dict items method. That calculates the pmf by evaluating the frequency of the total words and addition is performed at the time of ith word. And this overall calculation gives the return of the CDF dictionary (GeeksforGeeks 2021).

```
def CDF(d):  
    prob_sum, sum_vals = 0, sum(d.values())  
    for k, v in d.items():  
        # evaluate the PMF by measuring each  
        # the freq. by total of all frequencies then add  
        # all the PMFs till ith word which is the CDF of  
        # the ith word.  
        pmf = v / sum_vals  
        prob_sum += pmf  
        d[k] = prob_sum  
    # Return cdf dictionary
```

Figure 7: Function returns the count of dictionary value

(Source: Author 2022)

Then the third function is declared with the name of the main function and has the attributes which are then sent, x, and, n. For performing the task anyone is able to use the text file or input data in the form of a docstring. Some data sets are defined in the corpus which is split by the function of the split. Docstring which is split by the split method is saved as the string in l. A variable is declared which has the blank string which will be used for storing the partial sentence. After that it will be stored in the sent and out is declared for saving the final output.

```

    return d

# The main function reads the sentence/word as input
# from user and reads the corpus file. For faster processing,
# we have taken only the first 1000 words.

def main(sent, x, n):

    # it describe the simple text file.
    # anyone can use the text file for executing the code
    # for rading the file from corpus is attached below.

    # corpus = open('AH19334.txt','r').read()

    corpus = ''' is chance of project failure if code not executed properly.
is described the programming language.
is a chance of sucess.
is define as the biggest industry.
is known as the subsidiary of the company.
is a diversified country.
is a chance of sucess of project if algorithms are executed properly.
is a largest raw mataril supplier company in the world.
...

l = corpus.split()

# "temp_out" will be used to store each partial sentence
# which will later be stored into "sent". "out" is used to store
# the final output.

temp_out = ''
out = sent + ' '

```

Figure 8: main function

(Source: Author 2022)

A for loop is executed in the main function in the range of 0 to n-x and then calls the method of next_word_freq, CDF, which was declared earlier. A dictionary is created to store those values which are produced by the counter. The counter generates a random number to predict the next word which was calculated by using the CDF method. In order to generate the random number, a uniform number is called if the dictionary is empty, which means that the entered word is not part of the dataset, then the loop has to terminate and there is a need for a loop that will run from

the initial stage. There is also a chance for encountering an error when there is not enough input for unpacking the value. If the occurrence value by the dict item method is greater than and equal to the random number, then a try block is executed; otherwise a block is executed that terminates all loops and the loop is initialised again in the next step. The value is defined according to the key of the dictionary and stored in the temp_out variable as the key, and the key index is defined as the value of pos. after that output is printed randomly by taking input from the different number of samples (Simplilearn 2021).

```
for i in range(n - x):

    # calling the next_word_freq method that returns the frequency of each word next to sent in the whole word corpus

    func_out = next_word_freq(1, sent)

    # cdf_dict stores the cdf of each word in the above map that is calculated using method CDF.
    cdf_dict = CDF(func_out)

    # We use a random number to predict the next word. The word having its CDF greater than or equal to rand and less than or equal to 1.
    rand = random.uniform(0, 1)

    # If cdf dict is empty, it means the word.sentence entered by you does not exist in the corpus. Hence, break the loop and just print
    # the word entered by you. To implement this we use try-except block. If an error occurs it implies there aren't enough values to unpack
    # and this can happen only when your input is absent from the corpus.
    try: key, val = zip(*cdf_dict.items())
    except: break

    # Iterate through the cdf values and find the smallest value greater than or equal to the random number. That value is the
    # cdf of your predicted word. Add the key of the value to the output string and update the "sent" variable as "temp_out".
    for j in range(len(val)):

        if rand <= val[j]:
            pos = j
            break

    temp_out = key[pos]
    out = out + temp_out + ' '
    sent = temp_out
    print(out, end = '\n\n')

if __name__ == '__main__':
```

Figure 9: Final part

(Source: Author 2022)

Error analysis – Error analysis is a machine learning process by which observation, isolation, and diagnosis is predicted by the understanding of the high and low performance of the project. Error analysis model defines the difference in between the actual output and the output of the written code. Error analysis is performed by importing the warnings and ait considers the csv file which have the test data of the laboratory. Some blank lists are declared for appending the data set to the defined list. Three lists are zipped in the enumerate function. Enumerate function is used for defining the location of the data set. Ensemble is a model which is used for the

prediction of the data by evaluating the outcome of two or more than two models. Then a function is defined by the name of error analysis and a question set is considered for the analysis of the error. And then the graph is plotted by considering all the factors. After that a graphical model is designed by using the graphical function.

The apparent coherence of the sentences formed using the n-gram models trained on 8 and 40 texts is similarly increased when visualised. The leftmost value corresponds to the size of the n-gram model employed, as seen below:

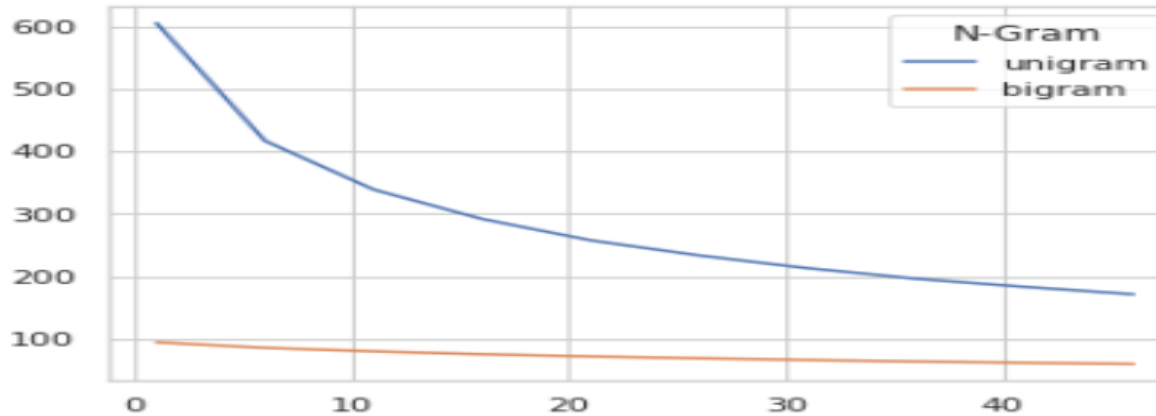
Document Size: 8

1. -the the , , the the the , ,
2. - and at the same time , and the.
3. - alone knew where Anne was.
4. - 'they shut or opened their gates with a trembling hand ,

Document Size: 40

1. - the,the.
2. - the same as the first.
3. - careless people should think.
4. - had spoken its simple reason through the lips of Dejah Thoris ' prison before long.

Figure 10: Error analysis by using various model



(Source: Author 2022)

Conclusion: -

In order to improve the digital and programming skill of the students, Microsoft promotes a challenge of word completion to complete the sentence by using various languages and libraries. Programming is the base for any industry because every industry in the world is getting automated to reduce the workforce. Programming is necessary because humans are highly attached to the machine so the communication between the human and machine is done with the help of the programming language. The report described the practical code which was developed in the lab for completing the sentence by predicting the word or sentence. The report proposed two methods for the completion of the sentence: the first one is the n-gram method and the second one is the neural programming approach. N-gram is a programming language model that depends upon the probability of predicting the next item so this concept is used in describing the model of word prediction by using the various libraries of python. The second method refers to the neural programming model which works by considering the various mathematics tools and libraries to perform the task. Majorly used modules are NumPy, matplotlib, and random. Various machine learning and data science tools such as TensorFlow and tokenizer are learned by using in the coding of the challenge. Learning various programming languages is helpful for students and working professionals to achieve their goals.

Bibliography

Borisov, O. (2020). *Text Generation Using N-Gram Model - Towards Data Science*. [online] Medium. Available at: <https://towardsdatascience.com/text-generation-using-n-gram-model-8d12d9802aa0> [Accessed 25 Apr. 2022].

GeeksforGeeks. 2018. *Python | Word Embedding using Word2Vec - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/python-word-embedding-using-word2vec/#:~:text=Word%20Embedding%20is%20a%20language,matrix%2C%20probabilistic%20models%2C%20> etc. [Accessed 25 Apr. 2022].

GeeksforGeeks. 2021. *How to calculate and plot a Cumulative Distribution function with Matplotlib in Python ? - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/how-to-calculate-and-plot-a-cumulative-distribution-function-with-matplotlib-in-python/> [Accessed 25 Apr. 2022].

Programiz.com. 2022. *Python Functions (def): Definition with Examples*. [online] Available at: <https://www.programiz.com/python-programming/function> [Accessed 25 Apr. 2022].

Simplilearn 2021. *Global Variable in Python With Examples*. [online] Simplilearn.com. Available at: <https://www.simplilearn.com/tutorials/python-tutorial/global-variable-in-python> [Accessed 25 Apr. 2022].

Simplilearn 2021. *Understanding All About Index in Python with Examples*. [online] Simplilearn.com. Available at: <https://www.simplilearn.com/tutorials/python-tutorial/index-in-python> [Accessed 25 Apr. 2022].

W3schools.com. 2022. *Python Modules*. [online] Available at: https://www.w3schools.com/python/python_modules.asp [Accessed 25 Apr. 2022].

Wang, D., Gong, C. and Liu, Q., 2019, May. Improving neural language modelling via adversarial training. In *International Conference on Machine Learning* (pp. 6555-6565). PMLR.

Appendices: -

[neural.py](#)

[ngram.py](#)