

# **NEXT WORD PREDICTOR**

Kavya Sree Chirumamilla, Trilok Vaddineni, Venkata Dharaneeswara Reddy Maddula, Jaya Sai Mohan Kollu

## **ABSTRACT:**

As the name suggests, it is about predicting the word that comes next. It is a type of recommendation system, because it gives us the suggestion of the words that are useful and make meaningful sense when used in that paraphrase. Here we used DENSE because it takes the input to the current stage from previous stage.

## **ACKNOWLEDGEMENT:**

We would like to thank our professor Dr. Syed Khushal Shah, for helping us in choosing project suitable for implementing AI to a decent level. We would also like to thank our TA Arefa, for constantly helping us with the queries regarding project during implementation. And Thanks to all my teammates for your constant efforts in completing the project.

- Kavya Sree Chirumamilla – 11534538
- Trilok Vaddineni – 11483870
- Venkata Dharaneeswara Reddy – 11549876
- Jaya Sai Mohan Kollu – 11549643

## **INTRODUCTION:**

### **1. PROBLEM SPECIFICATION:**

Our task here is to predict the next possible word that would bring a meaningful sense to the sentence that is being written. Here in this project after training the data with a dataset, we will give some inputs and this will provide us with the best next word.

### **2. DATASET:**

I got the dataset from the website: <https://www.gutenberg.org/files/35/35-0.txt>  
Here we downloaded the data set from the Gutenberg website; it has a total of 197921 words and 5163 unique words without any blank spaces. This is about the book named 'The Time Machine' which was written by H. G. Wells.

### 3. PROBLEM ANALYSIS:

Here we have to first adapt to the user way of texting, as the user might not only use the words which we trained to the program. We will get difficulties in predicting the next word, when the given input sentence has a misspelled word.

## DESIGN AND MILESTONE:

### 1. PROPOSED METHOD:

Here as we mentioned above we had used DENSE and along with that we also use LSTM (Long Short Term Memory). LSTM has the capability to process an entire sequence of data which will help us in predicting the next word. DENSE is also used in this, as neurons from previous layer give the input to the neurons in next layer.

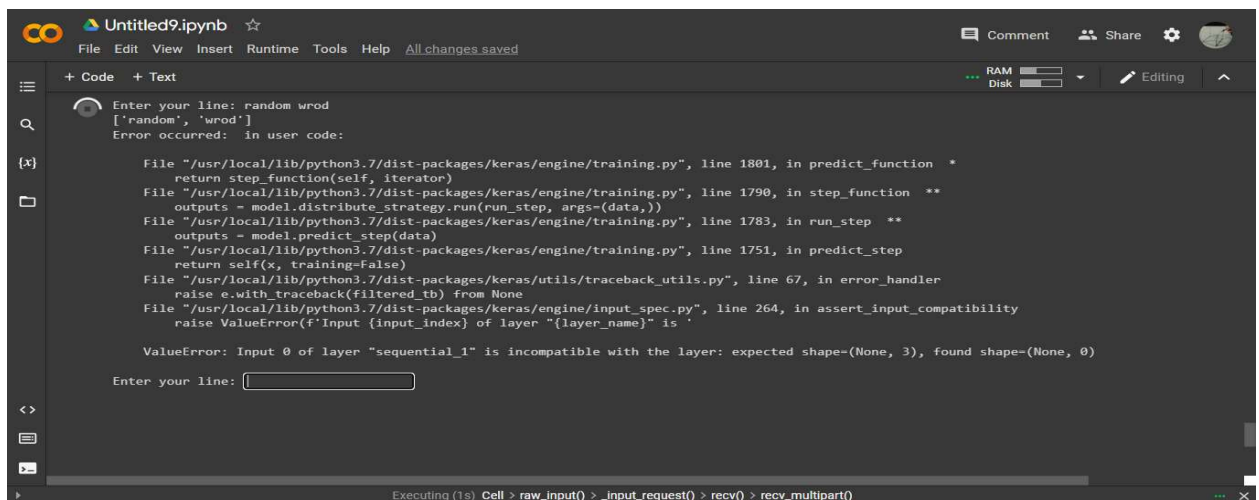
### 2. DATA PROCESSING:

As from the dataset we mentioned above, we process the words present in that data set and based on the previous words in the sentence we predict the best possible word that will come next that will make proper meaning.

Here we will open the data set in read mode and store the words in the form of list. Then we convert that list to string. Then we replaced the unnecessary symbols such as (), ",", "'", '\n', '\r with the white blank space. Then we removed the unnecessary blank spaces.

### 3. EXPERIMENTAL SETTINGS:

Here we have also shown a case where we will not get any output. As the algorithm will run and run, but will not be able to predict the next word, because we are training with one data set. If we train the program with multiple dataset it will work. Also we will get the error if spelt wrong, like the below one:



The screenshot shows a Jupyter Notebook interface with a dark theme. The title bar says "Untitled9.ipynb". The menu bar includes File, Edit, View, Insert, Runtime, Tools, and Help. The toolbar shows RAM and Disk usage, and an Editing mode button. The code cell contains the following text:

```
Enter your line: random wrod
['random', 'wrod']
Error occurred: in user code:

File ~/usr/local/lib/python3.7/dist-packages/keras/engine/training.py, line 1801, in predict_function *
    return step_function(self, iterator)
File ~/usr/local/lib/python3.7/dist-packages/keras/engine/training.py, line 1790, in step_function **
    outputs = model.distribute_strategy.run(run_step, args=(data,))
File ~/usr/local/lib/python3.7/dist-packages/keras/engine/training.py, line 1783, in run_step **
    outputs = model.predict_step(data)
File ~/usr/local/lib/python3.7/dist-packages/keras/engine/training.py, line 1751, in predict_step
    return self(x, training=False)
File ~/usr/local/lib/python3.7/dist-packages/keras/utils/traceback_utils.py, line 67, in error_handler
    raise e.with_traceback(filtered_tb) from None
File ~/usr/local/lib/python3.7/dist-packages/keras/engine/input_spec.py, line 264, in assert_input_compatibility
    raise ValueError(f'Input {input_index} of layer "{layer_name}" is '

ValueError: Input 0 of layer "sequential_1" is incompatible with the layer: expected shape=(None, 3), found shape=(None, 0)
```

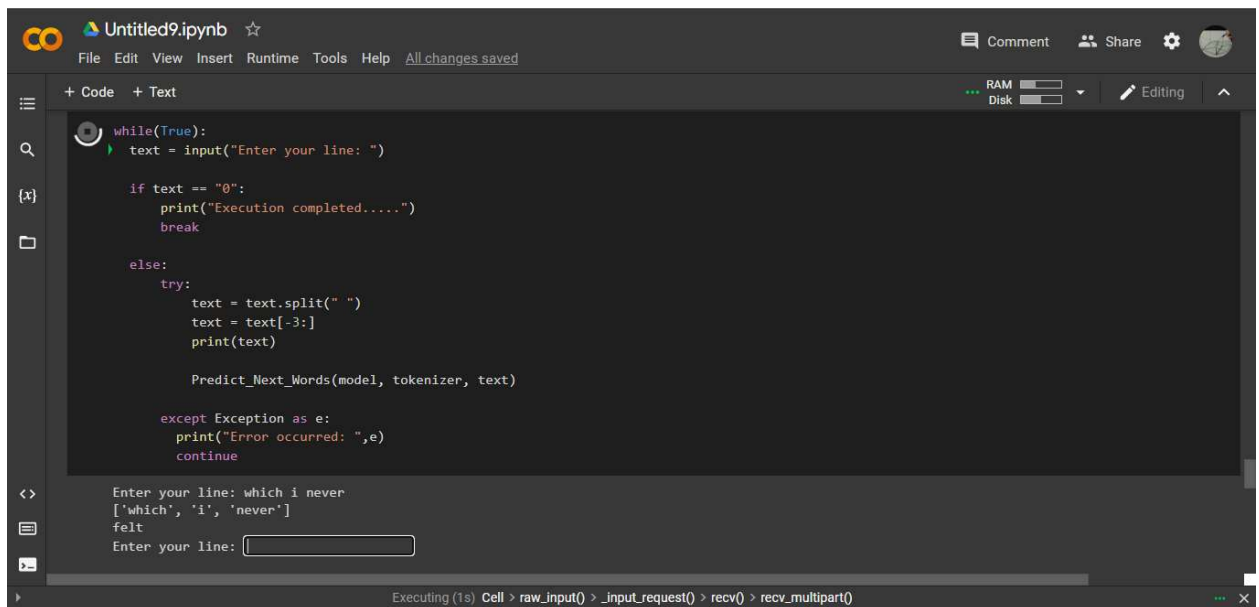
Below the error message, there is a text input field with the placeholder text "Enter your line:". The status bar at the bottom indicates "Executing (1s) Cell > raw\_input() > \_input\_request() > recv() > recv\_multipart()".

Now here as I mentioned in the limitations section, that we will get error if the word is spelt wrong.

#### 4. VALIDATION METHODS:

To validate whether the output we got through this method is correct or not we had an entry text box at the end of the program when we execute the code. We can then determine whether the output which is shown is good or not.

#### 5. RESULTS AND ANALYSIS:



The screenshot shows a Jupyter Notebook titled 'Untitled9.ipynb'. The code cell contains a Python script that runs in a loop. It prompts the user to 'Enter your line: '. If the input is an empty string, it prints 'Execution completed.....' and breaks. Otherwise, it splits the input by spaces, takes the last three words, and prints them. It also calls a function 'Predict\_Next\_Words(model, tokenizer, text)'. An exception handler prints the error if one occurs. The output shows the input 'which i never' and the output 'felt'.

```
while(True):
    text = input("Enter your line: ")

    if text == "":
        print("Execution completed.....")
        break

    else:
        try:
            text = text.split(" ")
            text = text[-3:]
            print(text)

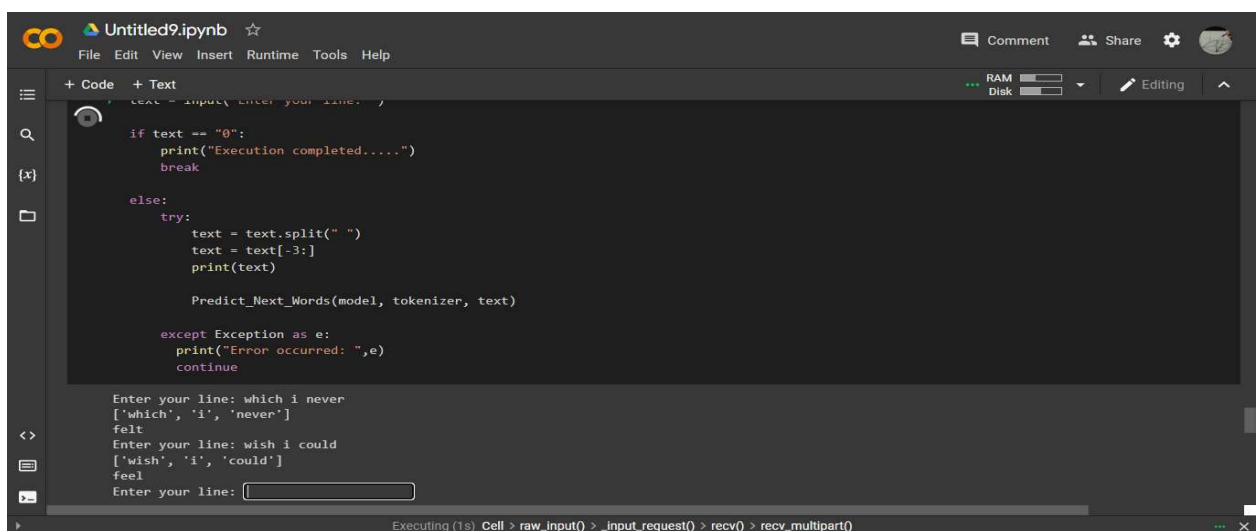
            Predict_Next_Words(model, tokenizer, text)

        except Exception as e:
            print("Error occurred: ",e)
            continue
```

Enter your line: which i never  
['which', 'i', 'never']  
felt  
Enter your line:

Here we show the output clearly by giving the input as “which I never”, the code gave as output as “felt”.

Below are some more examples as such:



The screenshot shows the same Jupyter Notebook interface. The code cell is the same as in the previous screenshot. The output shows two examples of input and output. The first example is 'which i never' resulting in 'felt'. The second example is 'wish i could' resulting in 'feel'.

```
text = input("Enter your line: ")

if text == "":
    print("Execution completed.....")
    break

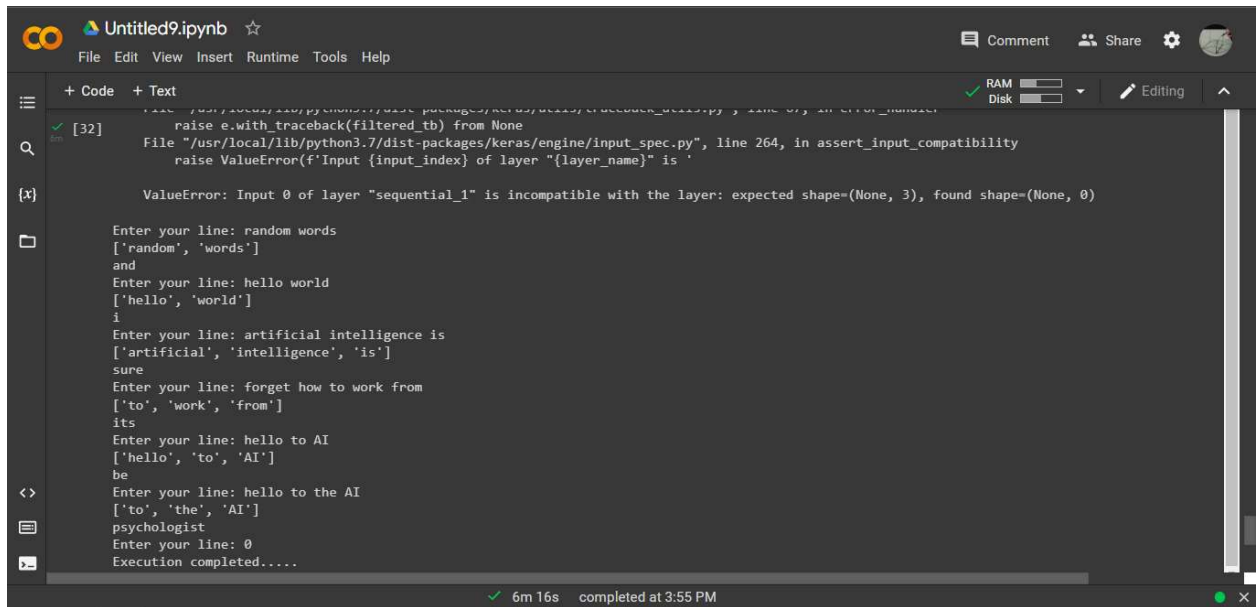
else:
    try:
        text = text.split(" ")
        text = text[-3:]
        print(text)

        Predict_Next_Words(model, tokenizer, text)

    except Exception as e:
        print("Error occurred: ",e)
        continue
```

Enter your line: which i never  
['which', 'i', 'never']  
felt  
Enter your line: wish i could  
['wish', 'i', 'could']  
feel  
Enter your line:

Here as you can see that we had implemented with another example that is: “wish I could” and the output is “feel”.



The screenshot shows a Jupyter Notebook titled 'Untitled9.ipynb'. The code cell [32] contains a traceback for a `ValueError` and a series of user inputs for a word completion model. The error message is: `ValueError: Input 0 of layer "sequential_1" is incompatible with the layer: expected shape=(None, 3), found shape=(None, 0)`. The user inputs are: 'random words', 'hello world', 'artificial intelligence is', 'forget how to work from', 'hello to AI', 'hello to the AI', and an empty input. The notebook status bar at the bottom indicates '6m 16s completed at 3:55 PM'.

```
[32]: raise e.with_traceback(filtered_tb) from None
      File "/usr/local/lib/python3.7/dist-packages/keras/engine/input_spec.py", line 264, in assert_input_compatibility
      raise ValueError(f'Input {input_index} of layer "{layer_name}" is '

ValueError: Input 0 of layer "sequential_1" is incompatible with the layer: expected shape=(None, 3), found shape=(None, 0)

Enter your line: random words
['random', 'words']
and
Enter your line: hello world
['hello', 'world']
i
Enter your line: artificial intelligence is
['artificial', 'intelligence', 'is']
sure
Enter your line: forget how to work from
['to', 'work', 'from']
its
Enter your line: hello to AI
['hello', 'to', 'AI']
be
Enter your line: hello to the AI
['to', 'the', 'AI']
psychologist
Enter your line: 
Execution completed.....
```

These were some other test cases that we implemented. Since trying different test cases, help us understand the limitations of the code.

## LIMITATIONS:

Since in the program we are using algorithms such as DENSE and LSTM, we are bound to get limitations.

For LSTM's:

- LSTM's take longer time to train itself
- There is a space complexity issue as this techniques requires a lot of space
- LSTM's are bound to get over-fit

For DENSE:

- These layers will not work properly when trying to detect repetitions and patterns.
- Cannot be used when we require some unique answers for the same questions each and everytime.
- Difficult to process the longer input sequences.
- It is very expensive when it comes to the computation process.

If a word is misspelled then it will be difficult to predict the next possible word.

## FUTURE WORK:

We can elaborate this to a much higher level, by using it to find the next possible word when listening to the noisy speech and can infer the meaning of the sentence by predicting. As for this model time is the biggest issue. As the time goes by the model will train itself via different inputs and increase the efficiency by learning from those data sets.

## PROJECT MANAGEMENT:

### 1. IMPLEMENTATION AND STATUS REPORT:

We had implemented the code completely and successfully executed the code. The below is the attached screenshot which tells that the code implemented is complete and successful.

### 2. WORK COMPLETED:

The project has been completed; we faced some difficulties at the starting while processing the data, then understanding and using the required libraries such as 'tokenizer', 'Embedding', 'Dense'.

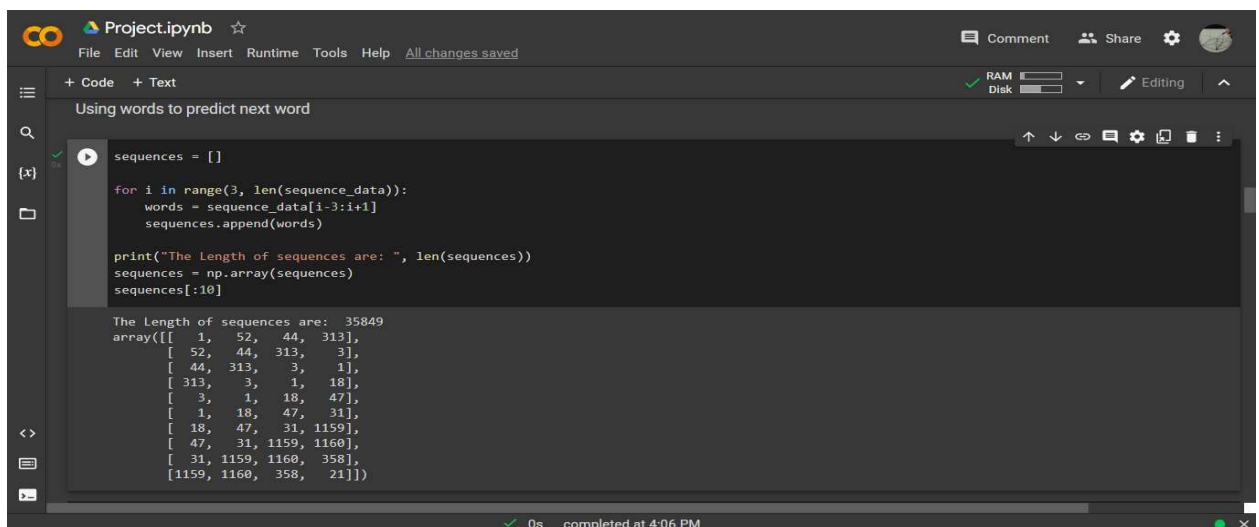
But to understand those libraries we took some help from web and we have attached those links in references section too.

### 3. DESCRIPTION:

Here we predict the next word using some libraries such as 'tokenizer', 'Embedding', 'Dense', 'LSTM', 'sequential' and many more.

We took the data set as input by uploading the file, then we processed the data set by eliminating some repeated words and unwanted spaces.

While predicting the word, in the below screenshot each row has 4 different entries, the first three are inputs and the last one is output:



```
Project.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Using words to predict next word
sequences = []
for i in range(3, len(sequence_data)):
    words = sequence_data[i-3:i+1]
    sequences.append(words)
print("The Length of sequences are: ", len(sequences))
sequences = np.array(sequences)
sequences[:10]

The Length of sequences are: 35849
array([[ 1, 52, 44, 313],
       [ 52, 44, 313, 3],
       [ 44, 313, 3, 1],
       [ 313, 3, 1, 18],
       [ 3, 1, 18, 47],
       [ 1, 18, 47, 31],
       [ 18, 47, 31, 1159],
       [ 47, 31, 1159, 1160],
       [ 31, 1159, 1160, 358],
       [ 1159, 1160, 358, 21]])
```

0s completed at 4:06 PM

#### 4. **RESPONSIBILITY:**

As this is a group project, each and every person did their respective part. We helped each other understand the things which we were not unable to understand. It is a collaborative effort because, we first started off as doing different things in project then slowly we collaborated and shared the knowledge we got to each other and then started to do the project again altogether. We assigned some separate tasks to everyone and those were:

- Kavya: Design of the model and problem solving.
- Trilok: Documentation and problem solving.
- Dharaneeswara: Model designing and helped in code understanding.
- Mohan: Code and learning the knowledge about the libraries and functions that were used

#### 5. **CONTRIBUTIONS:**

<b><u>NAMES</u></b>	<b><u>PERCENTAGE</u></b>
Kavya Sree Chirumamilla	100
Trilok Vaddineni	100
Venkata Dharaneeswara Reddy Maddula	100
Jaya Sai Mohan Kollu	100

#### 6. **ISSUES/CONCERNS:**

We encountered many issues and concerns while doing this project such as:

- Lack of understanding the libraries.
- Using the part of the codes, such as functionality of the keywords which we used.
- No knowledge from where to gather the dataset.

These were some of the key issues that we faced while completing this project.

#### **IMPLEMENTATION:**

Here for the implementation and the outputs of the code, I had uploaded a separate file with the name, 'Implentation&CodeOutputs.pdf'. It will have the complete code along with the outputs.

To explain briefly about the code, we did import the libraries which were required, and then we uploaded the dataset and processed it by storing the file in list and then converting them into the strings to train the code. Then we used the library functions that we imported such as “tokenizer” (this is used to convert text to sequence and sequence to text).

Then we will go through the sequence that was generated using the tokenizer library function (the screenshot is the same in Description). The rest will be in that pdf and I will explain in the video too.

## **REFERENCES:**

- <https://iq.opengenus.org/disadvantages-of-rnn/>
- <https://datascience.stackexchange.com/questions/27392/so-whats-the-catch-with-lstm>
- <http://mattmahoney.net/dc/textdata.html>
- [https://www.researchgate.net/publication/353773522\\_Next\\_Words\\_Prediction\\_Using\\_Current\\_NeuralNetworks](https://www.researchgate.net/publication/353773522_Next_Words_Prediction_Using_Current_NeuralNetworks)
- <https://ieeexplore.ieee.org/document/8985885>
- <https://www.youtube.com/watch?v=fNxaJsNG3-s>
- <https://www.youtube.com/watch?v=r9QjkdSJZ2g>
- <https://analyticsindiamag.com/tutorial-on-keras-tokenizer-for-text-classification-in-nlp/>
- <https://docs.python.org/3/extending/embedding.html#:~:text=Embedding%20provides%20your%20application%20with,writing%20some%20scripts%20in%20Python.>
- [https://www.tensorflow.org/api\\_docs/python/tf/keras/layers/Dense](https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense)
- <https://machinelearningknowledge.ai/keras-dense-layer-explained-for-beginners/>

**GITHUB LINK:** <https://github.com/Mohan-35/Project>