Next word Predictor using Deep Learning and NLP

Premkumar Vankudoth (Win: 104766296)*, Sai Raghav Karumanchi(Win:042113306), Likhita Pampana (464105177)
Master of Data Science,WMU
premkumar.vankudoth@wmich.edu
sairaghav.karumanchi@wmich.edu
likhita.pampana@wmich.edu

***Abstract*—This is part of the Data Science Course project, the goal of the project is to build a predictive text i,e what comes next which is similar to the technologies like google keyboards, emails, google search bar suggests relevant next words when we type something, In this report, text data from the kaggle have been taken and processed using Natural language processing methods before building a model. The frequencies of words in uni-gram terms were identified to understand the nature of the data for development and used techniques of deep learning like RNN(Recurrent Neural networks) and LSTM(Long short term Memory) to train the model. The final output of the model is user inputs a sentence and system predicts the next word for the given sentence.**

.

***Index Terms*—Kaggle, Unigram, Recurrent neural Networks, Long short term memory, Deep learning, natural language processing.**

I. Introduction

Around the Globe, people are spending an lots of time on their mobile devices for email, social networking, banking and many more activities. Typing on mobile devices can be easier if the keyboard can present options like what the next word might be.This model is similar to how a predictive text keyboard works on apps like What's App, Facebook Messenger, Instagram, e-mails, or even Google searches. for instance take google search bar and type in what is the weather we already receive some prediction like in (city depends on area you like), or today or now. building a such a model would save a lot of time by understanding the user's patterns of texting.This model will consider the last word of a particular sentence and predict the next possible word.

Natural language processing is one the most important technologies in the area of artificial intelligence, examples like chat bots and robots can automatically give answers when questioned similar like apple SIRI or google assistance.Today's natural language processing systems can analyze unlimited amounts of text-based data without fatigue and in a consistent, unbiased manner.

Deep learning is considered an evolution of machine learning. It chains together algorithms that aim to simulate how the human brain works, otherwise known as an artificial neural network

II. Aim

The main objective of the project was to develop a automated next word predictor with good model accuracy.

III. Problem Statement

Predicting the text what comes next before we type would be great whenever we use applications for texting or emailing. This applications trains the sequence data as uni gram sequence (one word at a time) and predicts the next word from the trained data it just makes man work easy by giving the suggestions.This could be also used by our virtual assistant to complete certain sentences

IV. Data Description

The Data sets for text data are easy to find and we can consider Project Gutenberg which is a voluntarily efforts, to "encourage the creation and distribution of eBooks". From here we can get many stories, documentations.We will use the text from the book Metamorphosis by Franz Kafka.The text data is in(.txt) format with which we load and processes and build the model.

V. Data prepossessing

Text data cleaning has series of tasks the steps we performed for the text data we user are:

- Removed all the tab spaces or extra symbol spaces with single spaces
- split the sentences into the series of string as a list format makes easy to work.

A. Text analysis with TensorFlow

Text-based applications are very popular use cases of deep learning. Text-based applications such as sentiment analysis and threat detection.
Keras is a high-level library that's built on top of TensorFlow. It provides a scikit-learn type API for building Neural Networks. Developers can use Keras to quickly build neural networks without worrying about the mathematical aspects of tensor algebra, numerical techniques, and optimization methods. Keras supports almost all the models of a neural network – fully connected.

1) Tokenization: W e will explore Keras tokenizer through which we will convert the texts into sequences that can be further fed to the predictive model.Tokenization is essentially splitting a phrase, sentence or an entire text document into smaller units, such as individual words or terms. Each of these smaller units are called tokens The tokens could be words, numbers or punctuation marks.. For instance take (

Natural language Processing after tokenization converts to ['natural','Language','Processing']
Before processing a natural language, we need to identify the words that constitute a string of characters. That's why tokenization is the most basic step to proceed with NLP (text data). This is important because the meaning of the text could easily be interpreted by analyzing the words present in the text.

2) Text-To-Sequences: This part will convert the texts to sequences. This is a way of interpreting the text data into numbers so that we can perform better analyses on them. We will then create the training data set.
These are the steps performed before proceeding with Model.

VI. Methods and Algorithms

We built a sequential model. We will then create an embedding layer and specify the input dimensions and output dimensions.

A. Recurrent Neural networks

Recurrent neural networks are kind of neural networks which are powerful for modelling data such as time series or natural language. In a simple neural network we assume that all inputs/outputs are independent of each other. If you want to predict the next word in a sentence you better know which words came before it.
RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations. In a RNN the information goes through a loop. When it makes a decision, it considers the current input and also what it has learned from the inputs it received previously. A recurrent neural network, however, is able to remember those characters because of its internal memory. The simple RNN can be found in below image.

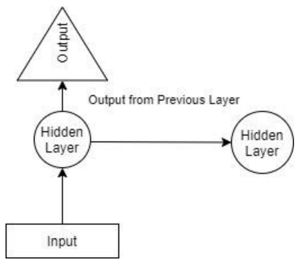

Fig. 1. Sample RNN Structure

B. Long short Term Memory

Long Short Term Memory networks are a special kind of RNN, capable of learning long-term memory from output from previous layer.They are explicitly designed to avoid the long-term dependency problem.Remembering information for long periods of time is behavior of LSTM.
LSTMs also have this chain like structure, but the LSTM has different structure than RNN. Instead of having a single neural network layer.These can solve numerous tasks not solvable by previous learning algorithms for recurrent neural networks (RNNs).
It uses gates to control information flow in the recurrent computations. LSTM networks are very good at holding long term memories. Here we used 2 LSTM layers with 1000 units each as the data is big and we are only using unigram it needs to remember each word.

C. Uni grams

N-grams of texts are extensively used in text mining and natural language processing tasks. An n-gram is a contiguous sequence of n items from a given sample of text. an n-gram of size 1 is referred to as a "unigram"; size 2 is a "bigram"; size 3 is a "trigram".If X=Number of words in a given sentence K, the number of n-grams for sentence K would be: Ngram = X - (N - 1) N is equal to 1 for unigram.
We can use any kind for this We used unigram as it the basic step for the projects like this. In Unigram we assume that the occurrence of each word is independent of its previous word. Hence each word becomes a gram(feature) here,for instance 'I', 'ate', 'banana' and all 3 are independent of each other. Although this is not the case in real languages.

VII. Implementation and Results

Once we made our uni gram sequence then we can convert our data to x (input) and y(output) and y can be transferred as a class variable with all zeros and the location with word as 1.
We will be building a sequential model. We build an embedding layer and specify the input layers and output layers. It is important to specify the input length as 1 since the prediction will be made on exactly one word and we will receive a output for that particular word.
Then we add a LSTM layer with 1000 units and make sequence as true.we will pass it through another LSTM layer. For the next LSTM layer, we will also pass it through another 1000 units no return sequence as it is false. We will pass this through a hidden layer with 1000 units using the dense layer function with relu as the activation. Finally, we pass it through an output layer with the specified vocab size and a softmax activation.
We will be importing the 2 required callbacks for training our model. The 2 important callbacks are ModelCheckpoint (storing the weights of our model after training), ReduceLROnPlateau (reducing the learning rate of the optimizer after a specified number of epochs).
We will be saving the best models based on the metric loss to the file nextword1.h5. This file will be crucial while accessing the predict function and trying to predict our next word.

A. Results and Observations

Once we build the model its time to compile the model, used our loss as categorical cross entropy for categorical and optimizer adam are set and then we will fit the model with 150 epochs.
For each epoch we saved the accuracy and plotted the graph

so that we could get a clear idea weather the model is over fitting or under-fitting or working fine.After 150 epochs we obtained an accuracy of 58 percent can be seen in the below graph.

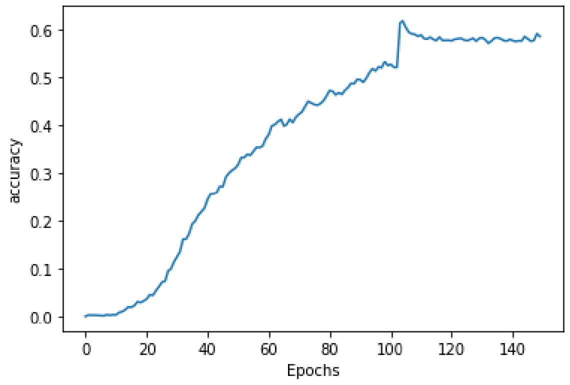

Fig. 2. Accuracy for Each Epoch

After the model is build for predictions we load our saved models and call those functions when a new sentence is passed for prediction, user needs to enter the sentence from the trained document once again the text is split and made sequences and them passed to the loaded model get the predictions and print the next word.There are certain cases where the program might not return the expected result. This is obvious because each word is being considered only once. This will cause certain issues for particular sentences and you will not receive the desired output.
The below image shows how the Prediction output looks like.

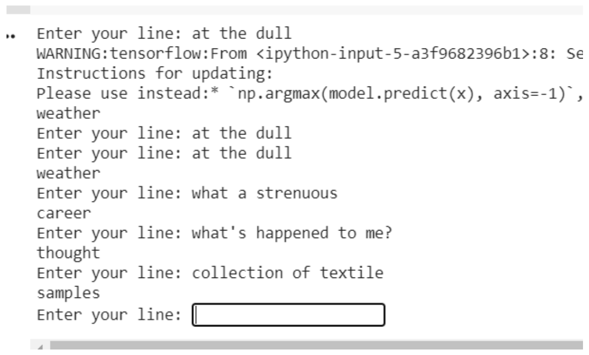
```
Enter your line: at the dull
WARNING:tensorflow:From <ipython-input-5-a3f9682396b1>:8: Se
Instructions for updating:
Please use instead:* `np.argmax(model.predict(x), axis=-1)`,
weather
Enter your line: at the dull
Enter your line: at the dull
weather
Enter your line: what a strenuous
career
Enter your line: what's happened to me?
thought
Enter your line: collection of textile
samples
Enter your line:
```

Fig. 3. Sample Output

VIII. Conclusion and Future works

To conclude, we were finally able to built next word predictor with 58 percent accuracy.We able to reduce the loss significantly with 150 epochs. The next word prediction model is fairly accurate on the provided data set.
For the future we can build a better model We have only used uni-grams for this project. Also, a few more additional steps can be done in the pre-processing steps. Overall, there is a lot of way we can do for betterment of model.we can build this model with other sequences like n gram or quad gram which may result in better accuracy.

IX. References

1.Yin, Wenpeng, et al. ”Comparative study of cnn and rnn for natural language processing.” arXiv preprint arXiv:1702.01923 (2017).

2.Lopez, Marc Moreno, and Jugal Kalita. ”Deep Learning applied to NLP.” arXiv preprint arXiv:1703.03091 (2017)

3.Ganai, Aejaz Farooq, and Farida Khursheed. ”Predicting next Word using RNN and LSTM cells: Stastical Language Modeling.” 2019 Fifth International Conference on Image Information Processing (ICIIP). IEEE, 2019.

4.Yu, Seunghak, et al. ”An Embedded Deep Learning based Word Prediction.” arXiv preprint arXiv:1707.01662 (2017).

5.Miller, John W. ”Word prediction system.” U.S. Patent No. 5,805,911. 8 Sep. 1998.

6.Sundermeyer, Martin, Ralf Schlüter, and Hermann Ney. ”LSTM neural networks for language modeling.” Thirteenth annual conference of the international speech communication association. 2018.

7.Agerri, Rodrigo, Josu Bermudez, and German Rigau. ”IXA pipeline: Efficient and Ready to Use Multilingual NLP tools.” LREC. Vol. 2014. 2014