Next Word Prediction

Ganga[1], Praveena[2] , Dr.J.B.Jona[3]
[1, 2] Dept of Master of Computer Applications,
[3]Associate Professor, Dept of Master of Computer Applications,
[1, 2, 3] Coimbatore Institute of Technology, Coimbatore (India)

Abstract- *Next word prediction, also known as language modeling, lies at the heart of natural language processing (NLP). It tackles the fundamental challenge of anticipating the most likely word to follow a given sequence of words. This seemingly simple task unlocks a diverse range of applications, impacting user interfaces, machine translation, text generation, and even dialogue systems. Replicating this ability in machines poses numerous challenges, including modeling long-range dependencies, handling ambiguity, and adapting to different domains and styles.*

Here the next word prediction has seen a shift from statistical techniques like n-grams to powerful deep learning models, particularly recurrent neural networks (RNNs) and their variants like Long Short-Term Memory (LSTM) networks. These models can learn complex representations of language, incorporating semantic and syntactic information to produce increasingly accurate predictions.

This leverages a combination of Python with Machine Learning. It requires a modern multi-core processor, a minimum of 4 GB RAM with major operating systems for optimal performance.

Keywords- Text Prediction, Machine Learning, N-grams, Long Short-Term Memory (LSTM)

I. INTRODUCTION

Word prediction tools were developed which might facilitate to speak and additionally to assist the individuals with less speed writing. During this paper, a language model based mostly framework for fast electronic communication, which will predict probable next word given a group of current words are briefed. Word prediction technique will the task of guesswork the preceding word that's probably to continue with few initial text fragments. Our goal is to facilitate the task of instant electronic communication by suggesting relevant words to the user. The proposed word prediction technique involves the task of anticipating the subsequent word likely to follow a given sequence of initial text fragments. By harnessing the power of language models, this framework aims to empower users with a more intuitive and responsive communication experience. The ultimate goal is to contribute to the advancement. This is because language is creative new sentence.

1.1 Problem statement

The problem at hand involves developing and refining language models to accurately predict the next word in a sequence, within a given text or sentence. The objective is to enhance these models' contextual understanding, syntactic structure comprehension, and semantic interpretation, aiming to generate precise and contextually relevant predictions. This task addresses the challenge of improving language generation by predicting subsequent words, thereby advancing natural language processing capabilities for various applications like autocomplete suggestions, predictive text entry, and chatbot interactions, ultimately striving for more coherent and accurate text generation based on contextual input.

1.2 Objective

The primary objective of next word prediction is to advance models in accurately forecasting the subsequent word following a given sequence within a sentence or text. This task focuses on enhancing language models' contextual comprehension, syntax understanding, and semantic interpretation, aiming to generate coherent and contextually fitting predictions. By achieving accurate predictions, the overarching goals include improving language understanding, text coherence, and user experience in language-based applications like autocomplete suggestions, predictive text entry, and chatbots. Ultimately, the aim is to refine language models' predictive abilities to generate more natural, coherent, and contextually appropriate text based on preceding input sequences.to accurately predict the next word in a sequence, within a given text or sentence. The objective is to enhance these models' contextual understanding, syntactic structure comprehension, and semantic interpretation, aiming to generate precise and contextually relevant predictions.

1.3 Literature Review

Existing systems work on word prediction model, that suggests subsequent immediate word supported this out their word. These systems work victimization machine learning algorithms that has limitation to form correct syntax. Multi-window convolution (MRN N) formula is enforced, additionally they need created residual-connected lowest gated unit(MGU) that is brief version of LSTM during this cnn try and skip few layers whereas coaching end in less coaching time and that they have sensible accuracy out and away victimization multiple layers of neural networks will cause latency for predicting n numbers of words .Developing technologies has been manufacturing additional correct outcomes than the prevailing system technologies, models developed victimization bidirectional LSTM algorithms area unit capable of handling additional knowledge expeditiously and predicts higher.

II. TECHNOLOGIES USED

This project the prediction for the next set of words uses the below methodologies for prediction.

Continual neural networks area unit generalizations of an instantaneous transmission neural network that has internal memory. The RNN is repetitive in nature as a result of it performs constant operate for every knowledge entry, however at constant time, this output depends on the previous calculation. the choice relies on associate degree analysis of this input and also the output from the previous input. RNNs will use their internal state (memory) to method input sequences once direct communication neural networks cannot. All RNN inputs area unit interconnected. This state formula is that the following :- hf = f(ht-1,xt) Application of the activation operate :- Ht = tanh(Whhht-1 + Wxhxt)

Implementing next word prediction in machine learning typically involves using natural language processing (NLP) techniques and recurrent neural networks (RNNs) or transformer models. TensorFlow and Keras are popular libraries for building and training neural networks. In the context of implementing next word prediction using TensorFlow, NumPy, and Keras, several key functionalities are employed.

Long Short-Term Memory (LSTM) networks and Recurrent Neural Networks (RNNs) are both sequential models used in the realm of deep learning for handling sequential data like time series or natural language. RNNs, being the foundational architecture, process sequential data by passing information sequentially from one step to another, allowing feedback loops to maintain information persistence. However, RNNs suffer from vanishing or exploding gradient problems during training, limiting their ability to retain long-term dependencies.

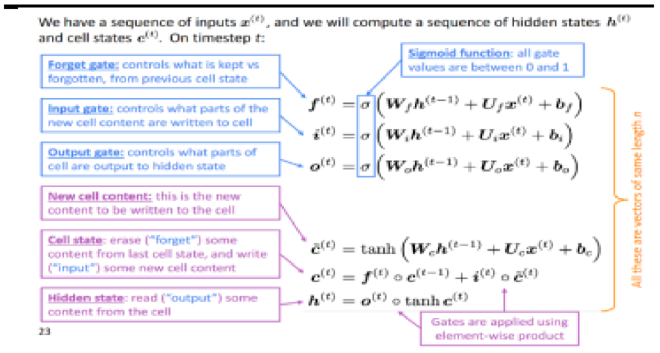

We have a sequence of inputs $x^{(t)}$, and we will compute a sequence of hidden states $h^{(t)}$ and cell states $c^{(t)}$. On timestep t:
Forget gate: controls what is kept vs forgotten, from previous cell state
Input gate: controls what parts of the new cell content are written to cell
Output gate: controls what parts of cell are output to hidden state
New cell content: this is the new content to be written to the cell
Cell state: erase ("forget") some content from last cell state, and write ("input") some new cell content
Hidden state: read ("output") some content from the cell
Sigmoid function: all gate values are between 0 and 1
$f^{(t)} = \sigma\left(W_f h^{(t-1)} + U_f x^{(t)} + b_f\right)$
$i^{(t)} = \sigma\left(W_i h^{(t-1)} + U_i x^{(t)} + b_i\right)$
$o^{(t)} = \sigma\left(W_o h^{(t-1)} + U_o x^{(t)} + b_o\right)$
$\tilde{c}^{(t)} = \tanh\left(W_c h^{(t-1)} + U_c x^{(t)} + b_c\right)$
$c^{(t)} = f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)}$
$h^{(t)} = o^{(t)} \circ \tanh c^{(t)}$
All these are vectors of same length n
Gates are applied using element-wise product
23

Fig -1 LSTM Architecture

III. PROPOSED METHODOLOGIES

3.1 Data Preprocessing

These area unit easy clean-up procedures that makes it easier to use the information in sequent steps. This method is administered with the assistance of Tensor flow library. The subsequent area unit few pre-processing steps typically done:-

1. Marking white areas
2. Lower-case conversions
3. Removing numbers
4. Removing punctuation
5. Removing unwanted words
6. Removing non-English words

3.2 Text Analysis

To know the speed of occurance of terms, Term Document Matrix operate was accustomed to produce term matrixes to achieve the account of term frequencies.

3.3 Pad Sequences

When changing sentences to numerical values, there's still a difficulty of providing equal length inputs to our neural networks. Not each sentence is constant length. pad_sequences operate is employed for artefact the shorter sentences with zeroes, and truncating a number of the longer sequences to be shorter. Additionally, because it is often specified whether or not to pad and truncate from either the start or ending, relying upon the pre settings and post settings for the padding arguments and truncating arguments. By default, artefact arguments and truncation can happen from the start of the sequence.

3.4 Data Preparation

Gather a large dataset of text, such as books, articles, or other sources of written language.

- Preprocess the text data by tokenizing it into words or subword units.
- Create sequences of words or subwords to serve as input-output pairs for training the model.

Word Embeddings:

- Convert words or subwords into numerical vectors using word embeddings. Popular word embedding techniques include Word2Vec, GloVe, or more advanced methods like contextual embeddings (e.g., BERT embeddings).

Sequence Modeling with LSTM:

- Build an LSTM-based neural network architecture. This typically involves defining an embedding layer, one or more LSTM layers, and a dense output layer.
- Train the model on your prepared dataset. The objective is to minimize the difference between the predicted next word(s) and the actual next word(s) in the training data.

Model Training:

- Split the dataset into training and validation sets.
- Train the model on the training set and validate its performance on the validation set.
- Adjust hyperparameters, such as the number of LSTM layers, the size of the hidden layers, and the learning rate, to optimize performance.

Prediction:

- Use the trained model to predict the next word(s) given a sequence of input words.
- Sampling techniques (like softmax sampling) can be employed to generate
- diverse and realistic predictions.

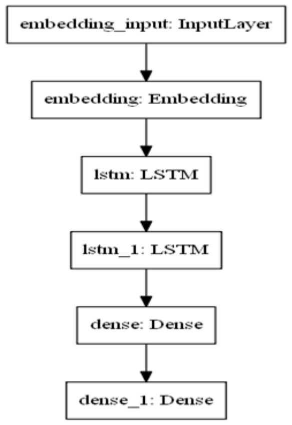


Fig-2. Flow of system

IV. CREATING MODEL

We are not using traditional RNN in predicting words we should know the context of the sentence rather than few previous words. For e.g.The sky is _________.

Here both RNN and LSTM will predict the word 'blue' but if the sentence is like:This is the 10th day of wildfires in the San Francisco bay area. There is smoke everywhere, it is snowing ash and the sky is _______.Here the RNN will predict 'blue' but the correct answer is 'red', that is the reason LSTM comes in use to tackle the long-term dependency problem because it has memory cells to remember the previous context.

1. LSTM (Long Short-Term Memory) Layer:

Objective: Capture sequential patterns and dependencies in the input sequences.
Implementation:Add an LSTM layer to the model, allowing it to learn long-term dependencies in the data.

2. Categorical Labels:

Objective: Prepare the labels in a categorical format to match the model's output layer.
Implementation:Convert labels into one-hot encoded vectors using to_categorical from Keras.

3. Model Compilation:

Objective: Specify the loss function and optimizer for training the model.
Implementation:Choose an appropriate loss function (e.g., categorical crossentropy) and optimizer (e.g., Adam) when compiling the model.

4. Training Hyperparameters:

Objective: Fine-tune hyperparameters to optimize the model's performance.
Implementation: Experiment with different values for epochs, batch size, and learning rate during model training.

5.Model Evaluation:

Objective: Assess the model's performance on a separate validation set.
Implementation:Split the data into training and validation sets. Monitor metrics such as accuracy and loss on the validation set during training.

6.Model Saving and Loading:

Objective: Save the trained model for future use without retraining.
Implementation:Save the model using model.save() after training.Load the saved model using tf.keras.models.load_model().

7. Generating Multiple Predictions:

Objective: Generate a range of potential next words along with their probabilities.
Implementation:Instead of selecting only the most probable word, consider sampling from the distribution of predicted probabilities for a more diverse output.

8. Integration with Pre-trained Models:

Objective: Leverage pre-trained word embeddings or language models for improved performance.
Implementation:Integrate pre-trained word embeddings (e.g., GloVe, Word2Vec) or use a pre-trained language model (e.g., GPT, BERT) as a starting point.
Use techniques like subword tokenization or handle unknown words gracefully in the prediction logic.

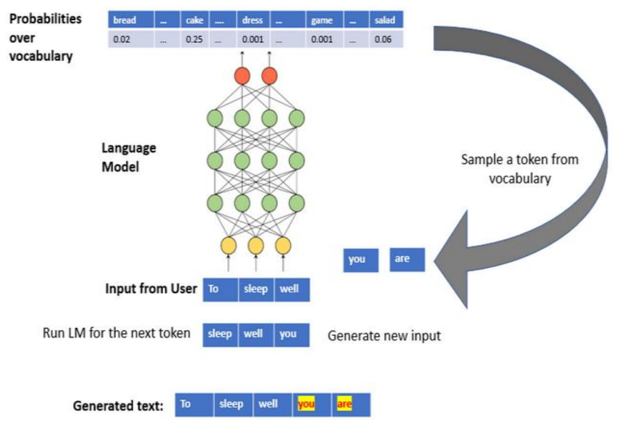
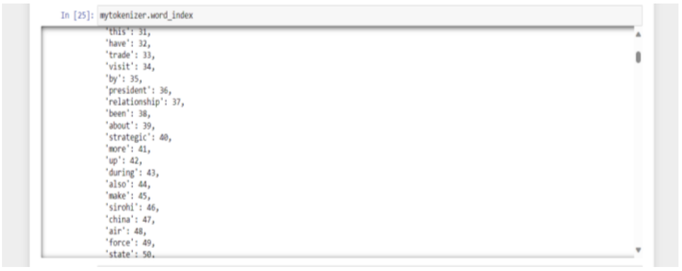


Fig3- Word Prediction

V. RESULTS AND ANALYSIS

The implementation and the result of the model is depicted below.

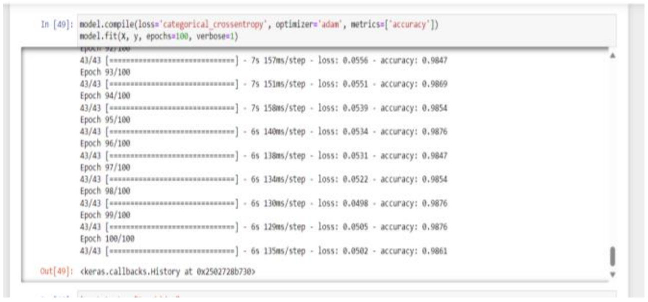
Fig-4 shows the separated dataset into individual units called as tokenization.



```
In [25]: mytokenizer.word_index
 'this': 31,
 'have': 32,
 'trade': 33,
 'visit': 34,
 'by': 35,
 'president': 36,
 'relationship': 37,
 'been': 38,
 'about': 39,
 'strategic': 40,
 'more': 41,
 'up': 42,
 'during': 43,
 'also': 44,
 'make': 45,
 'sirohi': 46,
 'china': 47,
 'air': 48,
 'force': 49,
 'state': 50,
```

Fig-4 Tokenization

Fig-5 shows the accuracy predicted during each epoch which is found to be 0.98 at the end of 100th epoch.



```
In [49]: model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
         model.fit(X, y, epochs=100, verbose=1)
         43/43 [==============================] - 7s 157ms/step - loss: 0.0556 - accuracy: 0.9847
         Epoch 93/100
         43/43 [==============================] - 7s 151ms/step - loss: 0.0551 - accuracy: 0.9869
         Epoch 94/100
         43/43 [==============================] - 7s 158ms/step - loss: 0.0539 - accuracy: 0.9854
         Epoch 95/100
         43/43 [==============================] - 6s 140ms/step - loss: 0.0534 - accuracy: 0.9876
         Epoch 96/100
         43/43 [==============================] - 6s 138ms/step - loss: 0.0531 - accuracy: 0.9847
         Epoch 97/100
         43/43 [==============================] - 6s 134ms/step - loss: 0.0522 - accuracy: 0.9854
         Epoch 98/100
         43/43 [==============================] - 6s 130ms/step - loss: 0.0498 - accuracy: 0.9876
         Epoch 99/100
         43/43 [==============================] - 6s 129ms/step - loss: 0.0505 - accuracy: 0.9876
         Epoch 100/100
         43/43 [==============================] - 6s 135ms/step - loss: 0.0502 - accuracy: 0.9861
Out[49]: <keras.callbacks.History at 0x2502728b730>
```

Fig-5 Epoch accuracy

VI. CONCLUSION

Next word prediction systems represent a vital component of natural language processing, facilitating smoother communication and enhancing user productivity across various applications. Through the utilization of machine learning algorithms, including sophisticated models like bidirectional LSTMs and innovative approaches such as Multi-window Convolution (MRNN) and Residual-connected Minimum Gated Units (MGU), these systems have made significant strides in improving prediction accuracy and efficiency.

REFERENCES

[1] Bengio, Y., Simard, P., Frasconi, P., 1994.Learning long dependencies with gradient descent is troublesome. IEEE transactions on neural networks five, 157– 166.
[2] R. Kneser and H. Ney, "Improved backing-off for n-gram language modeling”, Conference on Acoustics speech and Signal Process, pp. 181-184, 1995.

[3] Mohd. Majid and Piyush Kumar, Language Modelling: Next word Prediction, 2019.
[4] J. Yang, H. Wang and K. Guo, "Natural language Word Prediction Model supported Multi-Window Convolution and Residual Network," in IEEE Access, vol. 8, pp. 188036-188043, 2020, doi: 10.1109/ACCESS.2020.3031200.
[5] Sukhbaatar, S., Weston, J., Fergus, R., et al., 2015. End-to-end memory networks, in: Advances in neural information processing systems, pp. 2440–2448.
[6] Zhou, C., Sun, C., Liu, Z., Lau, F., 2015. A c-lstm neural network for text classification. arXiv preprint arXiv:1511.08630.
[7] Joel Stremmel, Arjun Singh. (2020). Pretraining Federated Text Models for Next Word Prediction using GPT2.