Next Word Prediction Using LSTM and RNN

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

4th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

5th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

6th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—Natural language processing (NLP) is about predicting the next word in a sentence given context. This technology is used widely in applications like auto correction for emails and text messages, as well as in tools like Google Search and MS Word, and uses commonly used language in order to improve user experience by anticipating possible next words or phrases. Natural language processing which can be helpful to predict the next word using neural networks like LSTM and RNN. In neural networks RNN has a capability of finding the unhidden sequence of patterns and the main problem of the RNN was it has a disadvantage when it comes to the long term dependencies but with the use of LSTM we can overcome this problem . LSTM has a Memory Cell unit where it can have a capability to store the long term dependencies.**

***Index Terms*—Deep Learning, Machine Learning, LSTM, RNN, Natural Language Processing**

I. Introduction

Machine Learning has an ability to build systems that can be getting better automatically experience . And here you can get experience by the model has been used to train the machines using built. Through that training, the machine will learn by. the patterns. From a new language. then the concept of the programming paradigm is different from traditional programming. In traditional then the inputs in programming are rules and data. output is the answer. In machine learning, actual data is seen alongside the theoretical concepts are the and various definitions are given in the First the inputs are answers and data, then the output is rules. This rule will be what we'll use later to build the model In that can recognize certain types of patterns such as if in.

The part of machine learning is itself deep learning that makes use of separate layers of neural networks classification and prediction decision. Some Classification are examples of the use include spam. includes detection, cancer prediction, sentiment analysis, and so on. We also use another example of it then. That is in daily life, and we don't usually realize it when we type.We use gadgets like mobile almost everyday. Computer and and then of course typing ! browsing on google to sending, messages. During those activities we may see that it It recommends the next word it expects us to type based on what we type already. This is called next word prediction because it predicts the next word for our texts. This helped us to grow our writing fluency and save time. An acute problem is that of next word prediction an arena of natural language problem processing.

While coming to the next word prediction tasks the maim key idea in this project was the data has a dependent on each other like when you are predicting the next word it will depend on the previous words it is fine for the small data sets but when it comes to the large datasets we will face the memory issues of the neural network model so that why we are using LSTM and RNN . LSTM has an capability to predict the next word because it has a memory cell unit and Forget gate which will store the information and it will forget the information if it is not used and it has a advantage of the longer term dependencies and RNN also useful for to find the sequence patterns but it has a disadvantage of longer term dependencies

II. Dataset

In this project we are using the medium data set where in this dataset it contains the articles opf the medium which was published in 2019 from the various trends of publications like Startup , Better Humans, Towards Data Science . This data we are having the shape of rows of 6508 and the 10 columns .

III. Data Visualization Data Preprocessing

We are dealing with the medium article dataset which contains the articles which are with the html tags and also the unwanted elements. Here we are using the BeautifulSoup tag to remove the html tags from the data and it will give the plain text . and next we are using the re.sub to remove the unwanted characters which are not letters and also it will remove the white spaces and punctuations and it will remove the symbols which are special also . In this text we are using the emoji removal to remove the emojis from the articles and it will replace the text by removing the emojis and also removes

Identify applicable funding agency here. If none, delete this.

the Unicode replacements it will replaces non breaking spaces with the regular spaces and i this project we are also removing the whitespaces.In this we have written also this under one function text preprocessing we are passing all our data to remove the unwanted information from the data and unwanted special characters . For the data visualization task we arre represnting the top words which are repeated across all the data and we are plotting the frequency of the topwords

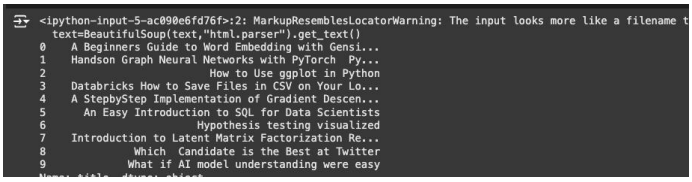


Fig. 1: text preprocessing

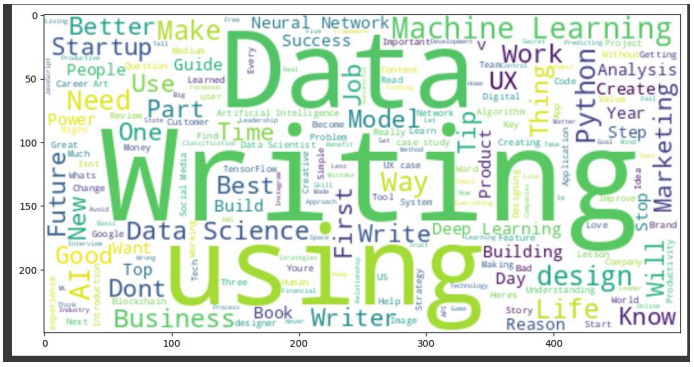


Fig. 2: word2vec

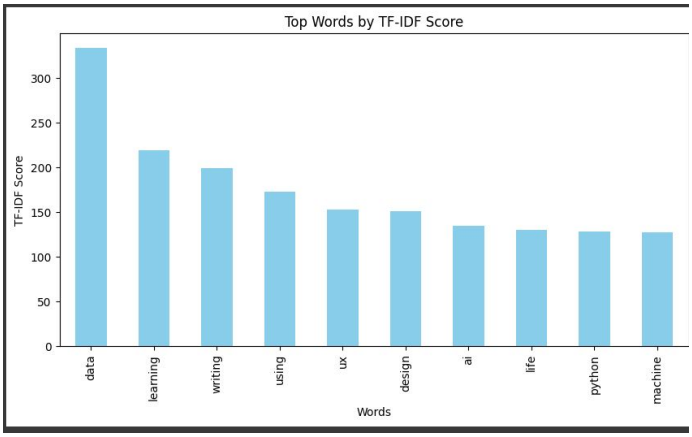


Fig. 3: Histogram of pixes covid images

IV. Existing Methods

N-gram Models: The next word prediction was done by using the probabilistic approaches where the current word is predicted by using the previous words. In this method it will calculate the probability of of a sequence of the words which are used widely for simple and which have the low resources applications . Whenever you are going with the longer term dependencies it will unable to perform and it also lacks the unseen hidden patterns while doing the next word prediction.

Convolution Neural Networks: Though the cnn networks are designed for the image enhancement tasks but cnn adapted for the Natural language processing tasks by treating the sequences of the word data as a spatial data and it has a ability to capture the local features.when ever you are dealing with the short term dependencies data it can be useful.

Gated Recurrent Units : These units are similar like LSTM but it has a architecture which was specified using fewer parameters and achieving the similar performances and it was designed to train the data fast as when compare to the LSTM and it mainly designed for the situations where the accuracy and computational efficiency are imporatnt

Transformers : It was a major booster method for the NLP tasks where it has a self attention mechanism where it can capture the complex relationships between the words compare where it doesn't matter the distance of the word . Bert model , GPT and T5 there transformer based model will predict the next word with high accuracy but it was a computational efficiency tasks it needs lot of computational power .

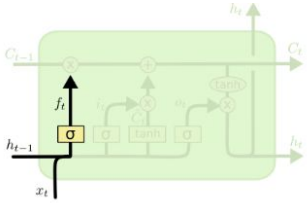
V. Algorithms

A. *Long short Term Memory*

LSTM neural networks are mainly designed for the longer term dependencies make it ready for the tasks which are mainly for the language modeling and time series forecasting. Main problem with this kind of the neural networks was it will always get the vanishing gradient problem but LSTM will overtake this problem that the RNN are facing by using the gates that it has and this gates will control the information flow.

B. *forget gate*

- This gate will decide that the information should be removed from the cell state and forget gate takes the input from the previous hidden state and also from current state input and in this architecture and in this gate it has the sigmoid activation function it will keep the data values in between 0 and 1 in each cell state . if the value or closer to the 0 that information will be forget and if the value is either 1 or closer to the 1 then the information will be regained

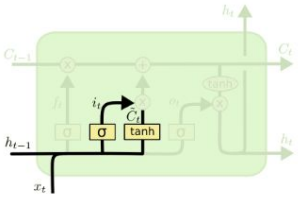


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Fig. 4: covid images

C. *Input gate*

- This gate will determined weather the new information will be stored in the cell state and mainly it has two parts and one of the main part is like that sigmoid the sigmoid layer needs to decide which state has to update and the activation tanh will create a vector that has the new value of candiates .

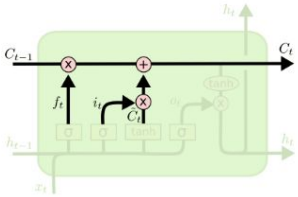


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Fig. 5: covid images

D. *Output gate*

- The hidden state is what part of the cell state will be output, and therefore controls the next time step. The final output is from the cell state, the output from the sigmoid layer, it creates.

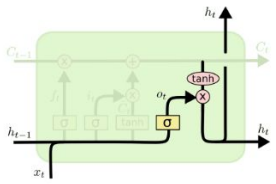


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Fig. 6: covid images

E. *final output*

- Using the forget gate , input gate , output gate the final cell state update will be done



$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Fig. 7: covid images

F. *Bidirectional LSTM*

In a bidirectional LSTM we have two LSTM networks which were designed to sequin the process for the beginning to end like in forward direction and next the other sequences it will be processing form in backward direction which is from end to beginning . Meanwhile in this process it will capture the information from both ends like from past and the future context for one process at each time . In this we have forward pass and the backward process and the concatenation. In this the forward pass will be the the information will be pass through the one layer of LSTM from left to the right end and meanwhile the backward pass will be the another LSTM layer will pass the information from right to the left end and the hidden states form the both of the directions will be combined to form the final output .

G. *Recurrent Neural Networks*

For data in sequence, we have Recurrent Neural Networks (RNNs), a specicalized type of neural network that incorporates previous context remember by a unique feedback mechanism. RNNs use hidden state, a hidden state filled with the information coming from the past time steps, and use them to solve tasks like language modeling, speech recognition and time series prediction. parameter sharing is achieved through the iterative hidden state updates that incorporated

Whenever we have the long sequences of the data it will cause a major problem in language modeling tasks like next word prediction the major problem was the vanishing gradient . However, RNNs are unable to learn long-term dependencies and have short term memory limitations as a result of this problem. To take care of this, complicated variants like LSTMs (Long Short-term Memory networks) as well as GRUs (Gated Recurrent Units) come in with doorways that take care of coming back data inside of lengthy collections, alleviating the vanishing gradient predicament and increasing long expression memory.

In RNN the BTTP backpropagation through time will functions like it will roll the network forward and it will backprop at each time step meanwhile RNN can run in the both directions with from the forward and backward and it will give a model with a chance to take in both the past and future context to increase prediction accuracy. . RNNs are still good enough to be the default in sequence modeling despite of some limitations, however newer architectures are transformers, which parallelize dependencies rather than sequentially.

VI. Methodology

The methodology for this project involves the following steps:

- LSTM (Long Short Term Memory) networks work through a special kind of Recurrent Neural Network (RNN) that is used to exploit long range dependencies in sequences. The last networks of this study use LSTM networks, composed of memory cells that apply input, forget and output gates to guard the flow of information so that the model can remember or forget information from long sequences. This structure deals with the vanishing gradient problem of traditional RNNs and let the model learn and store long term dependencies. Fine Tuning the performance of LSTM models depends heavily on hyper parameters. Other key hyperparameters include the number of layers, the number of units (hidden state size) in each layer, learning rate, dropout rate for regularization, and batch size—it matters how we feed the data to the

network and so on. To increase the efficiency and to converge the training we are using the SGD optimizer and also the learning rate scheduler . These hyperparameters are usually tuned through experiment or grid search for a best matching for a particular task.

- Using Recurrent Neural Network (RNN) architecture, this provided code has been implemented to perform text generation. Data preprocessing is first, where the text is ready cleaned with beautiful soup and regular expressions, say removing special characters, emoji or extra spaces. A custom Tokenizer class is then used to tokenise the cleaned text, and convert it into sequences of integers. All sequences are padded to achieve uniform length over all samples. The architecture of the neural network is defined in the TextGenerationModel class, which—the model is supposed to be—determines the core of the model. The architecture has an embedding layer and a bidirectional RNN layer and a fully connected layer. The tokens for the input has gone to the embedding layer which convert that to dense vectors which will represent the semantic relationship in between the words here in this bidirectional RNN will process the sequence information in between from the both directions from forward and backward with this process the model can learn from the previous and future rewards
- Using of the optimizer it will increase the efficiency and it will help the model to get high accuracy and here we are using the Adam optimizer and the learning rate of 0.001 this learning rate will help to reach the gradient fater and we are using the weight decay of l2 regularization to train the model . We use 512 batch size and 200 epochs training to get the accuracy , precision ,recall the model will be trained . A learning rate scheduler is created which reduces the learning rate when the loss plateaus to assist in convergence towards a better solution. Finally, we plot loss, accuracy, precision and recall on train epochs to view the model performance after training. These give you a view on how the model learns as well as spotting problems like overfitting, other fitting conditions. It also includes a function of our trained model that accepts an input seqencc and predicts the next word, that is the text generation tasks.

VII. LSTM Results

A. Accuracy Over Epochs

In the initial phase (0-50 epochs), the accuracy improves rapidly as the model begins to learn. During the middle phase (50-150 epochs), there are slight fluctuations, but accuracy remains around 0.85, indicating a high rate of correct predictions. By the later phase (150-300 epochs), accuracy levels off at approximately 0.8520, showing that the model has optimized its performance and is no longer gaining significant improvements.

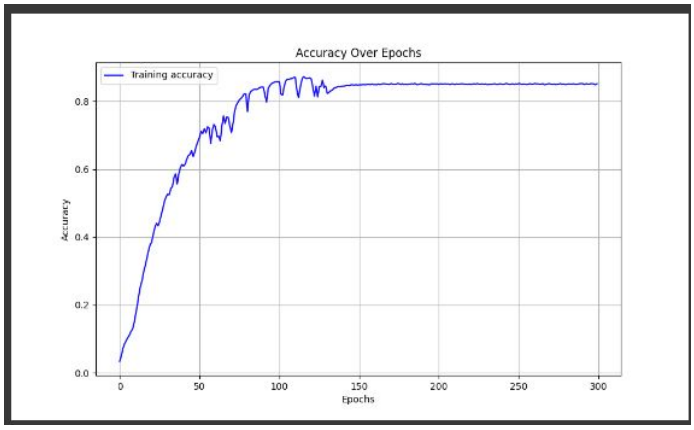


Fig. 8: accuarcy

B. Loss Over Epochs

During the initial phase of training, the loss decreases rapidly, indicating that the model is effectively minimizing errors and learning key patterns. In the middle phase (after approximately 100 epochs), the rate of loss reduction slows significantly. By the later phase, the model has minimized errors to a low level, stabilizing around a low loss value, signifying effective convergence.

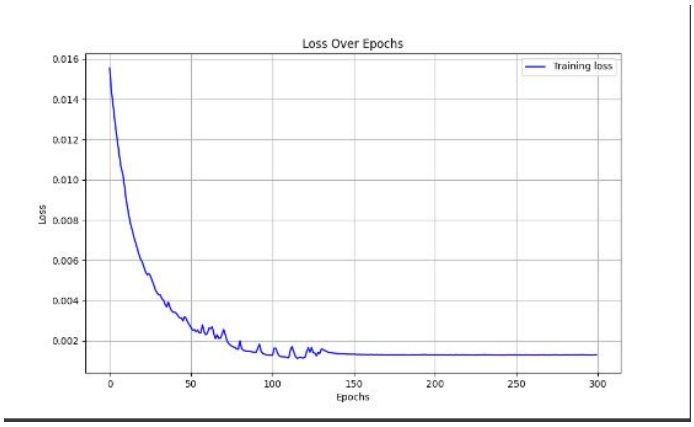


Fig. 9: loss

C. Recall Over Epochs

In the early epochs, recall improves quickly, indicating that the model is rapidly learning to identify positive instances. During the middle phase (50-150 epochs), recall fluctuates slightly but stabilizes around 0.8520, demonstrating the model's consistency in identifying relevant instances. In the later phase, recall plateaus at around 0.8520, suggesting the model has reached a point of diminishing returns for learning new patterns in positive instance detection.

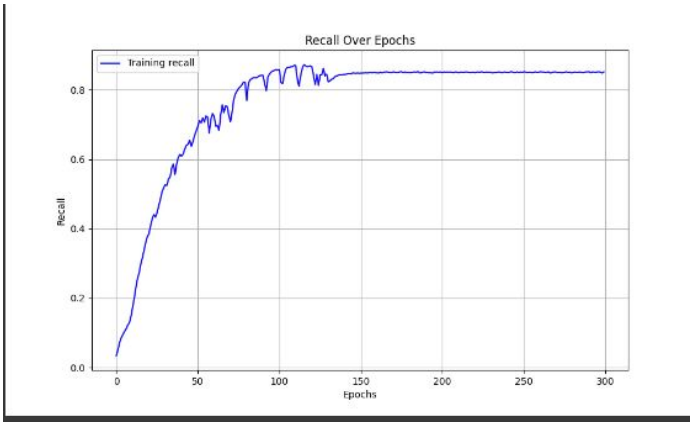


Fig. 10: VGG16 Predicted Outputs

D. Precision Over Epochs

Precision follows a similar trend to recall, with rapid improvement in the initial phase. In the middle phase (50-150 epochs), precision stabilizes around 0.8520 with minor fluctuations, indicating the model's accuracy in making positive predictions. By the later phase, precision plateaus, similar to recall and accuracy, reflecting the model's optimized performance.

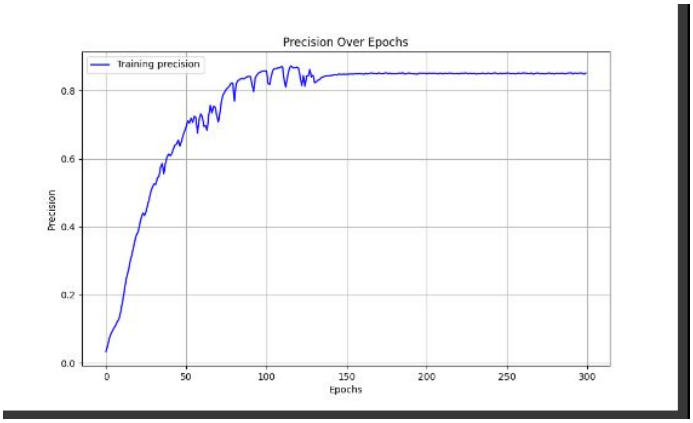


Fig. 11: VGG16 Predicted Outputs

E. Overall Analysis

Following initial learning, the model demonstrates strong performance across recall, accuracy, and precision, all stabilizing around 0.8520. These results indicate minimal errors and effective convergence. The high recall, accuracy, and precision levels suggest that the model is robust and reliable, with no signs of overfitting or underfitting in the later epochs as it maintains a balanced performance across all metrics.

The final next word prediction reulst are here we have given the input in the list and next the model is predicting the next word

```
Input: 'A Step-by-Step Implementation of' -> Predicted next word: 'Data'
Input: 'An Introduction to Deep Learning and' -> Predicted next word: 'Data'
Input: 'Exploring the Basics of Natural Language' -> Predicted next word: 'Processing'
```

Fig. 12: LSTM Next Word predicted

VIII. RNN RESULTS

A. Accuracy Over Epochs

In the initial phase (0-50 epochs), the accuracy improves rapidly as the model begins to learn. During the middle phase (50-150 epochs), there are slight fluctuations, but accuracy remains around 0.85, indicating a high rate of correct predictions. By the later phase (150-300 epochs), accuracy levels off at approximately 0.8505, showing that the model has optimized its performance and is no longer gaining significant improvements.

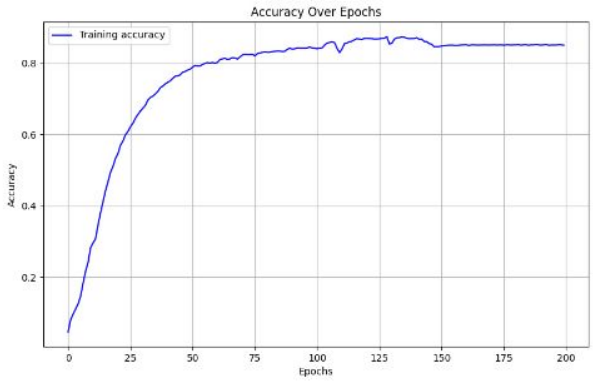


Fig. 13: RNN accuarcy

B. Loss Over Epochs

During the initial phase of training, the loss decreases rapidly, indicating that the model is effectively minimizing errors and learning key patterns. In the middle phase (after approximately 100 epochs), the rate of loss reduction slows significantly. By the later phase, the model has minimized errors to a low level, stabilizing around a low loss value, signifying effective convergence.

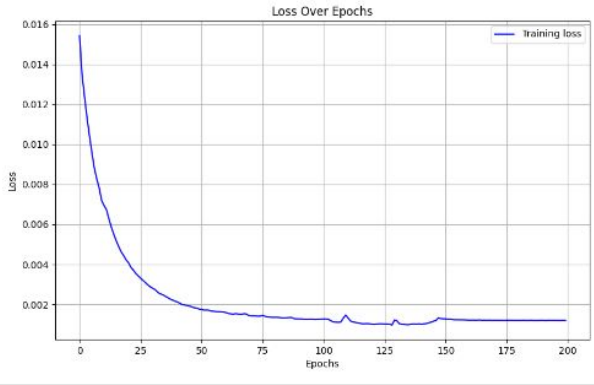


Fig. 14: loss rnn

C. Recall Over Epochs

In the early epochs, recall improves quickly, indicating that the model is rapidly learning to identify positive instances.

During the middle phase (50-150 epochs), recall fluctuates slightly but stabilizes around 0.85, demonstrating the model's consistency in identifying relevant instances. In the later phase, recall plateaus at around 0.8520, suggesting the model has reached a point of diminishing returns for learning new patterns in positive instance detection.

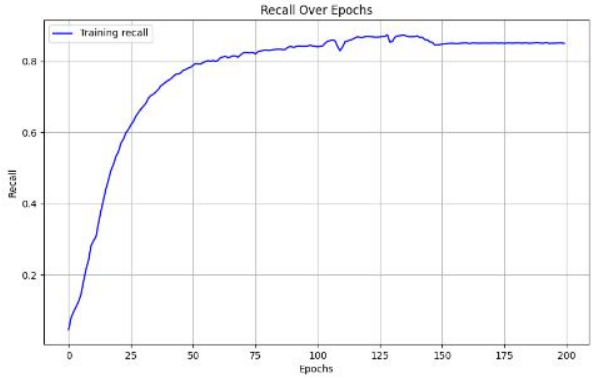


Fig. 15: recall rnn

D. Precision Over Epochs

Precision follows a similar trend to recall, with rapid improvement in the initial phase. In the middle phase (50-150 epochs), precision stabilizes around 0.85 with minor fluctuations, indicating the model's accuracy in making positive predictions. By the later phase, precision plateaus, similar to recall and accuracy, reflecting the model's optimized performance.

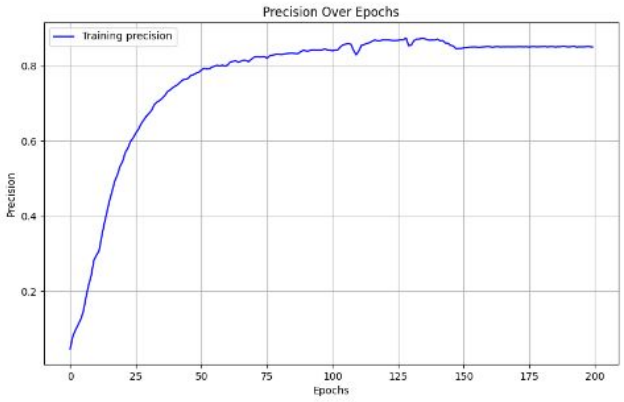


Fig. 16: precison rnn

E. Overall Analysis

Following initial learning, the model demonstrates strong performance across recall, accuracy, and precision, all stabilizing around 0.85. These results indicate minimal errors and effective convergence. The high recall, accuracy, and precision levels suggest that the model is robust and reliable, with no signs of overfitting or underfitting in the later epochs as it maintains a balanced performance across all metrics.

```
Input: ' An Easy Introduction to SQL for Data Scientists' -> Predicted next word: 'Remastered'
Input: 'Hypothesis testing visualized' -> Predicted next word: 'in'
Input: 'Introduction to Latent Matrix Factorization' -> Predicted next word: 'Recommender'
```

Fig. 17: RNN Next Word predicted

The final next word prediction reulst are here we have given the input in the list and next the model is predicting the next word

IX. Conclusion

In this work, we investigate the performance of two types of Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) models on next word prediction tasks. We observe than both architectures can well handle sequential data while the LSTM model clearly dominates the standard RNN. Its ability to capture long term dependencies while overcoming vanishing gradient problems makes it especially suited to language prediction tasks in which getting this information into the LSTM pipe step is crucial.

In all epochs, the LSTM model outperformed the other model with higher values of accuracy, recall and precision. LSTM was found to be effective and reliable for prediction of next word, as these metrics stabilized at optimal levels with no signs of overfitting.

References

[1] Jordan, Michael I., and Tom M. Mitchell. "Machine learning: Trends, perspectives, and prospects." *Science*, vol. 349, no. 6245, 2015, pp. 255-260. doi: https://doi.org/10.1126/science.aaa841510.1126/science.aaa8415.

[2] Sahoo, Abhaya Kumar, Chittaranjan Pradhan, and Himansu Das. "Performance evaluation of different machine learning methods and deep-learning based convolutional neural network for health decision making." In *Nature Inspired Computing for Data Science*, pp. 201-212. Springer, Cham, 2020. doi: https://doi.org/10.1007/978-3-030-36720-$3_1$010.1007/978 − 3 − 030 − 36720 − $3_1$0.

[3] Prajapati, Gend Lal, and Rekha Saha. "REEDS: Relevance and enhanced entropy based Dempster Shafer approach for next word prediction using language model." *Journal of Computational Science*, vol. 35, 2019, pp. 1-11. doi: https://doi.org/10.1016/j.jocs.2019.07.00610.1016/j.jocs.2019.07.006.

[4] Ambulgekar, Sourabh, Sanket Malewadikar, Raju Garande, and Bharti Joshi. "Next Words Prediction Using Recurrent Neural Networks." *ITM Web of Conferences*, vol. 40, p. 03034, EDP Sciences, 2021. doi: https://doi.org/10.1051/itmconf/2021400303410.1051/itmconf/20214003034.

[5] Sarwar, S. M., and Abdullah-Al-Mamun. "Next word prediction for phonetic typing by grouping language models." In *2016 2nd International Conference on Information Management (ICIM)*, pp. 1-6. IEEE, 2016.

[6] Kim, Y., J. An, M. Lee, and Y. Lee. "An Activity-Embedding approach for next-activity prediction in a multi-user smart space." 2017, pp. 1-6.

[7] Rakib, O. F., S. Akter, M. A. Khan, A. K. Das, and K. M. Habibullah. "Bangla word prediction and sentence completion using GRU: An extended version of RNN on N-gram language model." *2019 International Conference on Sustainable Technologies for Industry 4.0 (STI)*, pp. 1-6, 2019.

[8] Sassi, A., M. Brahimi, W. Bechkit, and A. Bachir. "Location Embedding and deep convolutional neural networks for next location prediction." *2019 IEEE 44th LCN Symposium on Emerging Topics in Networking (LCN Symposium)*, pp. 149-157, 2019. .

[9] Bengio, Y., P. Simard, and P. Frasconi. "Learning long dependencies with gradient descent is troublesome." *IEEE Transactions on Neural Networks*, vol. 5, no. 2, 1994, pp. 157-166. doi: https://doi.org/10.1109/72.27918110.1109/72.279181.