Next Word Wizard: A Dynamic LSTM-based Autocomplete System for Fluid and Intuitive Writing

Amey Shankar Basangoudar
Khoury College of Computer Sciences
Northeastern University
Boston, MA
basangoudar.a@northeastern.edu

Ataish Nehra
Khoury College of Computer Sciences
Northeastern University
Boston, MA
nehra.at@northeastern.edu

Mahesh Babu Kommalapati
Khoury College of Computer Sciences
Northeastern University
Boston, MA
kommalapati.m@northeastern.edu

Mohit Chodisetti
College of Engineering
Northeastern University
Boston, MA
chodisetti.m@northeastern.edu

Sameer Prasad Koppolu
Khoury College of Computer Sciences
Northeastern University
Boston, MA
koppolu.s@northeastern.edu

***Abstract*—We have implemented Next Word Prediction, a type of Natural Language Processing (NLP) model that predicts the most likely word to follow a given sequence of words. The model uses various statistical and neural network techniques such as Markov Model, LSTM, Bi-LSTM, and Transformers to analyse large amounts of text data and identify patterns and relationships between words to make accurate predictions. Next word prediction has practical applications such as auto-completion of text, machine translation, and content creation, making it an essential tool for productivity and efficiency in the digital age. This article provides an overview of the various techniques used in NLP models and their practical applications, highlighting the importance of Next Word Prediction in NLP.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

In today's digital age, Natural Language Processing (NLP) has become increasingly popular due to its ability to enable efficient and accurate communication between humans and machines. According to [1], one of the most useful applications of NLP is the Next Word Prediction model, which predicts the most likely word to follow a given sequence of words. As in [2], this model utilizes both statistical and neural network techniques to analyze large amounts of text data and identify patterns and relationships between words. With practical applications like auto-completion of text, machine translation, and content creation, Next Word Prediction is an essential tool for anyone looking to enhance their productivity and communication skills.

Next Word Prediction is an indispensable application of Natural Language Processing (NLP) that enables machines to predict the most likely word to follow a given sequence of words as in [3]. This model utilizes both statistical and neural network techniques to analyze vast amounts of text data to identify patterns and relationships between words, enabling machines to make accurate predictions. According to [4], With numerous practical applications such as auto-completion of text, machine translation, and content creation, Next Word Prediction has become an essential tool for people who are looking to improve their writing skills and communication capabilities.

Next Word Prediction has become a vital application of Natural Language Processing (NLP), enabling machines to predict the most likely word to follow a given sequence of words accurately. As in [5], This model uses a combination of statistical and neural network techniques to analyze massive amounts of text data to identify patterns and relationships between words, allowing machines to make accurate predictions. As in [6], With practical applications such as auto-completion of text, machine translation, and content creation, Next Word Prediction is an essential tool for individuals who are looking to increase their efficiency and productivity while working with text data.

II. Method

We then utilized four different Next Word Prediction models, including LSTM, Bi-LSTM, Transformers, and Markov Model. LSTM and Bi-LSTM models are recurrent neural networks that can capture long-term dependencies in the input sequence, while Transformers use self-attention mechanisms to process the entire sequence of words at once. As in [7], Markov Model is a probabilistic model that assumes the probability of a word depends only on the preceding n-words. By evaluating and comparing the performances of these models, we can gain insights into their strengths and weaknesses and determine their suitability for different use cases.

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

A. Dataset and Data Pre-processing

The dataset New York Times comments obtained from Kaggle[1]. was used to train and test the LSTM and BiLSTM models, as well as the Markov model. The dataset consists of comments on articles from The New York Times published in April 2017. The dataset contains a total of 1,411,956 comments. The entire set of articles was used for the Markov model. The LSTM and BiLSTM models were trained and tested on a portion of the dataset (specifically the News Articles from April 2017) due to processor limitations. The models were evaluated based on their perplexity and loss values. The LSTM model achieved a perplexity of 6.86 and a loss of 8.836, while the BiLSTM model achieved a perplexity of 15.29 and a loss of 9.635. The Markov model achieved a perplexity of 90.21 and a loss of 4.502. The results of these models were compared to evaluate their performance in generating text.

For the Transformer models, the same dataset was used for training and the Yelp dataset from the Kaggle[2]. was used for testing. The Yelp dataset contains a total of 5,261,668 reviews. The Transformer models used in this project include DistilBert, AlBert, Bert, and RoBerta. These models were trained on a portion of the NYT dataset and tested on the Yelp dataset. The perplexity and loss values were used to evaluate the performance of the models. The DistilBert model achieved a perplexity of 30.14 and a loss of 3.4, while the AlBert model achieved a perplexity of 15.78 and a loss of 2.75. The Bert model achieved a perplexity of 19.86 and a loss of 2.98, and the RoBerta model achieved a perplexity of 4.24 and a loss of 1.44. The results of these models were also compared to evaluate their performance in generating text. Overall, the combination of datasets and models allowed for a thorough analysis of text generation techniques.

In our data preprocessing phase, we cleaned the text data by converting it to lowercase and splitting it into individual lines. The Keras Tokenizer function was then used to create a tokenizer object that converted the text data into sequences of integers. N-gram sequences were generated from the integer sequences using a nested loop, and these sequences were padded and split into input data and labels to train the next word prediction model. We used pre-trained GloVe embeddings, and a dictionary called 'embeddings_index' was created to store the word and its embedding coefficients as key-value pairs. We then created a numpy matrix called 'embeddings_matrix' with the same number of rows as the total number of unique words in the input data and the same number of columns as the embedding dimension. Finally, the embeddings matrix was populated with the GloVe embeddings for the words in the input data using the embeddings index dictionary to look up the embedding coefficients for each word.

[1]https://www.kaggle.com/datasets/aashita/nyt-comments.

[2]https://www.kaggle.com/datasets/omkarsabnis/yelp-reviews-dataset.

B. Modelling

1) Markov Chain: We implemented the Markov Model to generate new text based on a given corpus by analysing sequences of words. This model is based on modelling the transition between states in a stochastic process. We created methods for suggesting the next word in a sentence based on the number of preceding words given as input. The Markov Model was trained on a corpus of 10,000 articles, which allowed it to learn and predict the likelihood of words following specific sequences. This method was evaluated on a test dataset and proved to be effective in generating coherent and grammatically correct sentences. If a word or sentence outside the corpus is encountered, the Markov chain model will not have a probability value assigned to it. To overcome the limitation of the Markov chain model we have used subsequent models.

2) LSTM: Firstly, For the LSTM model, we specified a hidden layer with 128 units, a dropout rate of 0.5, and a learning rate of 0.01. We then trained the model on the input data for 50 epochs, with a batch size of 256 and a verbosity level of 1, meaning that progress updates were displayed during training. The hidden layer units determine the number of memory cells in the LSTM, which can help the model capture long-term dependencies in the input sequence. The dropout rate is a regularization technique that randomly drops out connections between neurons during training to prevent overfitting. The learning rate determines the step size taken in the direction of the gradient during training. By adjusting these hyperparameters and evaluating the resulting model performance, we can tune the LSTM to achieve optimal results. LSTM can suffer from vanishing gradients or exploding gradients during training. So we have implemented BiLSTM to overcome this issue.

3) BiLSTM: Secondly, For the BiLSTM model, we used similar hyperparameters to the LSTM model, including a hidden layer with 128 units, a dropout rate of 0.5, and a learning rate of 0.01. We trained the model on the input data for 50 epochs, with a verbosity level of 1. The key difference between the BiLSTM and the LSTM is that the BiLSTM processes the input sequence in both directions, allowing the model to capture not only previous but also future context. This can be particularly useful in next word prediction tasks, where the context surrounding a given word is crucial for accurate predictions. By adjusting the hyperparameters and comparing the performance of the BiLSTM to other models, we can evaluate its effectiveness for the given task. However, BiLSTM is computationally expensive, to address the shortcoming we have used Transformers.

4) Transformers: Finally, we utilized attention mechanisms to process sequential data and employed different transformer models such as DistilBert, AlBert, Bert, and RoBerta. These models were trained on a dataset of 10,000 news articles over 10 epochs and evaluated on an external test dataset of 5,000 Yelp reviews. The evaluation process was done to measure the accuracy and effectiveness of the transformer models in predicting the next word in a sequence. By using these

transformer models, we aimed to improve the performance of our next word prediction model and achieve better results in various NLP applications such as content creation and machine translation.

III. Results

In this study, we employed different models to predict the next word in each sequence of text. Our evaluation metrics were perplexity and mean test score. The "Fig. 1" shows the ablation study on LSTM and BiLSTM Models. The LSTM model had a hidden layer with 128 units, a dropout rate of 0.5, and a learning rate of 0.01, and it achieved a perplexity of 6.86 and a mean test score of -8.836. The BiLSTM model also had a hidden layer with 128 units, a dropout rate of 0.5, and a learning rate of 0.01, and it achieved a perplexity of 15.29 and a mean test score of -9.635. The transformer models that we used in this study were DistilBert, AlBert, Bert, and RoBerta. DistilBert achieved a perplexity of 30.14 and a loss of 3.4, AlBert achieved a perplexity of 15.78 and a loss of 2.75, Bert achieved a perplexity of 19.86 and a loss of 2.98, and RoBerta achieved the best results with a perplexity of 4.24 and a loss of 1.44.

dropoutRate_learnRate_lstmUnits_Model	perplexity	mean_test_score
0.5_0.01_128_BILSTM	6.861027	-8.833612
0.5_0.01_64_BILSTM	8.731209	-9.074659
0.5_0.001_64_BILSTM	9.591643	-9.168648
0.5_0.001_128_BILSTM	12.432793	-9.428093
0.5_0.001_64_LSTM	15.293210	-9.635164
0.2_0.01_128_BILSTM	20.562212	-9.931210
0.5_0.01_128_LSTM	23.238059	-10.053547
0.5_0.01_64_LSTM	27.547236	-10.223657
0.5_0.001_128_LSTM	28.139688	-10.244936
0.2_0.01_64_BILSTM	39.882721	-10.593698
0.2_0.001_64_LSTM	48.157840	-10.782239
0.2_0.001_128_BILSTM	57.125324	-10.953003
0.2_0.01_128_LSTM	60.022999	-11.002483
0.2_0.01_64_LSTM	61.678286	-11.029687
0.2_0.001_64_BILSTM	64.849076	-11.079818
0.2_0.001_128_LSTM	66.385650	-11.103236

Fig. 1. Ablation Study of LSTM and BiLSTM Models

Model	LSTM	BiLSTM	DistilBert	AlBert	Bert	RoBerta
Perplexity	6.86	15.29	30.14	15.78	19.86	4.24
Loss	8.836	9.635	3.4	2.75	2.98	1.44

Fig. 2. Ablation Study of LSTM Models and Transformers

The results in "Fig. 2" shows us that the LSTM model outperformed the BiLSTM model in terms of perplexity and mean test score. However, the transformer models performed better than both LSTM and BiLSTM models, with RoBerta achieving the best results in terms of perplexity and loss. These findings suggest that transformer models are more effective than traditional recurrent neural network models in predicting the next word in a sequence of text.

Overall, our study demonstrates the effectiveness of different models in predicting the next word in a sequence of text. The results highlight the importance of choosing the right model architecture and parameters to achieve optimal performance. These findings have important implications for natural language processing and can be applied in a range of applications, including text prediction, machine translation, and chatbot development.

IV. Discussions

In this project, we explored various deep learning models for language modeling and next word prediction tasks. We initially dealt with the Markov model – a naïve model that predicts the next word based on the input sequences of words present in their exact manner in the corpus. Thereafter, we applied LSTM, BiLSTM, and transformer models such as DistilBert, AlBert, Bert, and RoBerta. We also used a Markov model to generate new text based on a given corpus. Our results showed that the LSTM and BiLSTM models outperformed the transformer models on the next word prediction task, with the LSTM model achieving the lowest perplexity and the highest test score. On the other hand, RoBerta achieved the best performance among the transformer models with the lowest perplexity and loss. Our findings suggest that while transformers have shown great promise in various NLP tasks, traditional models such as LSTMs and BiLSTMs still perform well in language modeling and next word prediction tasks. Furthermore, the Markov model was able to generate coherent and meaningful text based on the input corpus. Overall, our project highlights the importance of selecting the appropriate model for a given NLP task and the potential of Markov models in text generation.

V. Conclusion

In conclusion, this project aimed to build and compare different language models for next word prediction using various techniques such as LSTM, BiLSTM, Transformers, and Markov models. The results show that each model has its strengths and weaknesses and performs differently depending on the type of data and task at hand. The Markov model, though simple, performed reasonably well and could be used as a baseline model for next word prediction. But, it does not provide satisfactory results as it is limited by the vocabulary and the order of the sequences in the corpus. Hence to predict a word for a large number of input sequences would require a very large corpus. On the other hand, LSTM and BiLSTM models performed well on the given text dataset, while Transformers showed promising results on the training data with a perplexity of 4.24 and performed very well on the external Yelp reviews dataset. The findings of this project demonstrate the importance of selecting the appropriate language model based on the data and task requirements. Future work could focus on incorporating other advanced language modeling

techniques and evaluating them on different types of data. Overall, this project provides insights into the development and evaluation of next word prediction models and can be useful in various NLP applications.

REFERENCES

[1] Barman, P.P. and Boruah, A., 2018. A RNN based Approach for next word prediction in Assamese Phonetic Transcription. Procedia computer science, 143, pp.117-123.

[2] Ganai, A.F. and Khursheed, F., 2019, November. Predicting next word using RNN and LSTM cells: Stastical language modeling. In 2019 Fifth International Conference on Image Information Processing (ICIIP) (pp. 469-474). IEEE.

[3] Ambulgekar, S., Malewadikar, S., Garande, R. and Joshi, B., 2021. Next Words Prediction Using Recurrent NeuralNetworks. In ITM Web of Conferences (Vol. 40, p. 03034). EDP Sciences.

[4] Sutskever, I., Martens, J. and Hinton, G.E., 2011. Generating text with recurrent neural networks. In Proceedings of the 28th international conference on machine learning (ICML-11) (pp. 1017-1024).

[5] Siami-Namini, S., Tavakoli, N. and Namin, A.S., 2019, December. The performance of LSTM and BiLSTM in forecasting time series. In 2019 IEEE International Conference on Big Data (Big Data) (pp. 3285-3292). IEEE.

[6] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M. and Davison, J., 2020, October. Transformers: State-of-the-art natural language processing. In Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations (pp. 38-45).

[7] Jang, B., Kim, M., Harerimana, G., Kang, S.U. and Kim, J.W., 2020. Bi-LSTM model to increase accuracy in text classification: Combining Word2vec CNN and attention mechanism. Applied Sciences, 10(17), p.5841.

[8] Tenney, I., Das, D. and Pavlick, E., 2019. BERT rediscovers the classical NLP pipeline. arXiv preprint arXiv:1905.05950.

[9] Tarwani, K.M. and Edem, S., 2017. Survey on recurrent neural network in natural language processing. Int. J. Eng. Trends Technol, 48(6), pp.301-304.

[10] Mikolov, T. and Zweig, G., 2012, December. Context dependent recurrent neural network language model. In 2012 IEEE Spoken Language Technology Workshop (SLT) (pp. 234-239). IEEE.

APPENDIX A

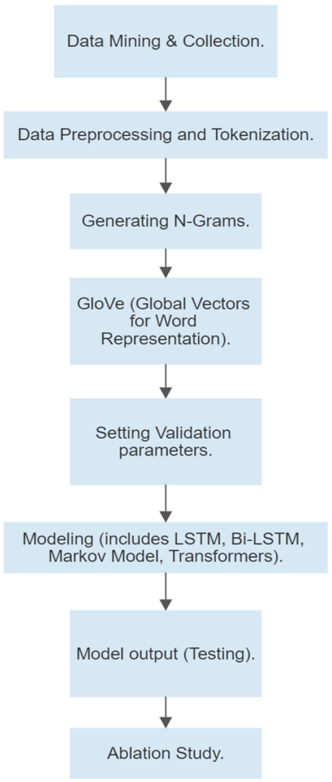

Fig. 3. Flowchart of the Methodology

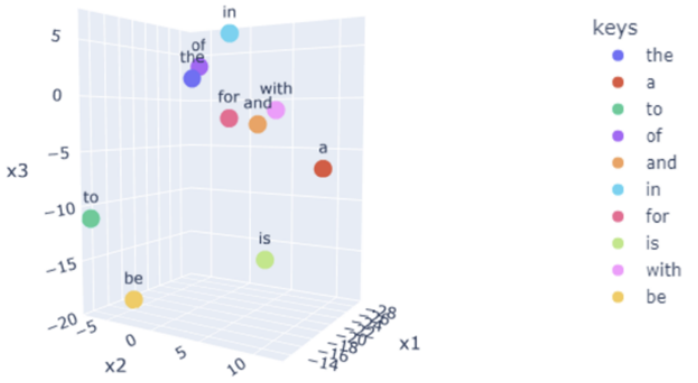

Fig. 4. TSNE Visualization of Glove Embeddings for the Top 10 Most Frequent Words

[3]Link to Code : https://github.com/sameerprasadkoppolu/Next-Word-Wizard-Group-18-NLP-Project-.

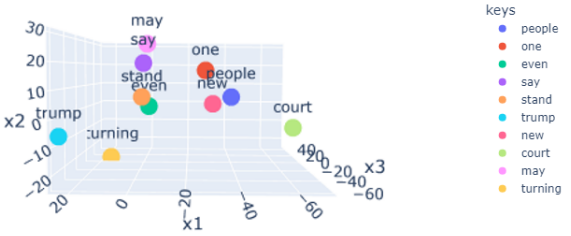

Fig. 5. TSNE Visualization of Glove Embeddings for the Top 10 Most Frequent Words that are not Stop Words

```
Enter the sentence to generate next word prediction: What
Possible next words are:
Next word suggestions: [('do', 58), ('you', 50), ('is', 49)]
------------------------------------------------------------
Enter the sentence to generate next word prediction: What do you want
Possible next words are:
Next word suggestions: [('to', 1)]
------------------------------------------------------------
```

Fig. 6. Markov Model sample output

embedding_49_input	input:	[(None, 34)]
InputLayer	output:	[(None, 34)]
embedding_49	input:	(None, 34)
Embedding	output:	(None, 34, 100)
lstm_98	input:	(None, 34, 100)
LSTM	output:	(None, 34, 256)
dropout_73	input:	(None, 34, 256)
Dropout	output:	(None, 34, 256)
lstm_99	input:	(None, 34, 256)
LSTM	output:	(None, 256)
dropout_74	input:	(None, 256)
Dropout	output:	(None, 256)
dense_98	input:	(None, 256)
Dense	output:	(None, 128)
dense_99	input:	(None, 128)
Dense	output:	(None, 1731)

Fig. 7. LSTM Model Structure

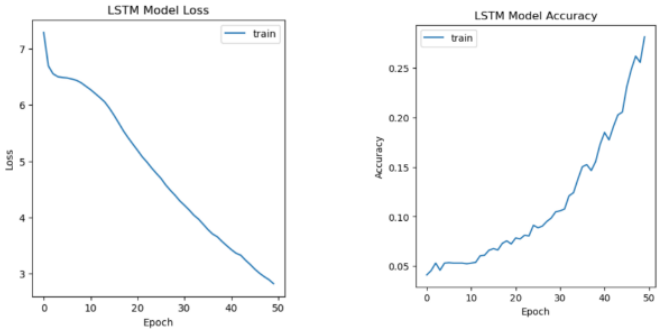

Fig. 8. LSTM Model Loss and Accuracy

embedding_50_input	input:	[(None, 34)]
InputLayer	output:	[(None, 34)]
embedding_50	input:	(None, 34)
Embedding	output:	(None, 34, 100)
bidirectional_50(lstm_100)	input:	(None, 34, 100)
Bidirectional(LSTM)	output:	(None, 34, 512)
bidirectional_51(lstm_101)	input:	(None, 34, 512)
Bidirectional(LSTM)	output:	(None, 512)
dense_100	input:	(None, 512)
Dense	output:	(None, 128)
dropout_75	input:	(None, 128)
Dropout	output:	(None, 128)
dense_101	input:	(None, 128)
Dense	output:	(None, 1731)

Fig. 9. BiLSTM Model Structure

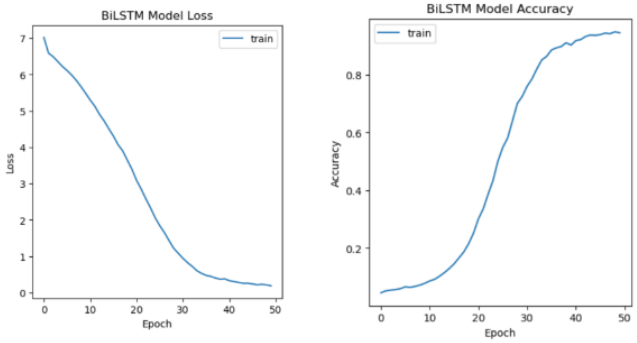

Fig. 10. BiLSTM Model Loss and Accuracy

Distill Bert Model Training Loss

```
In [164]: result_df_dis_bert
```

Out[164]:

	Epochs	Loss	perplexity
0	1	11.004016	60115.108706
1	2	9.949603	20943.906482
2	3	8.930533	7559.293090
3	4	8.403626	4463.221352
4	5	7.322774	1514.399539
5	6	6.500705	665.610599
6	7	5.766952	319.562250
7	8	4.791068	120.429940
8	9	4.455917	86.135119
9	10	3.405928	30.142253

Fig. 11. Distill Bert Model perplexity and loss

Albert Model Training Loss

```
result_df_albert
```

	Epochs	Loss	perplexity
0	1	9.171308	9617.197755
1	2	8.951017	7715.731495
2	3	7.863577	2600.807256
3	4	7.215315	1360.101679
4	5	7.263971	1427.914913
5	6	6.002040	404.252499
6	7	4.862526	129.350549
7	8	4.649932	104.577846
8	9	3.777681	43.714558
9	10	2.758860	15.781843

Fig. 12. Albert Model perplexity and loss

Bert Model Training Loss

```
result_df_bert
```

	Epochs	Loss	perplexity
0	1	9.940946	20763.370562
1	2	9.331098	11283.518625
2	3	8.534450	5087.031005
3	4	7.489810	1789.712552
4	5	6.845112	939.278245
5	6	6.100550	446.103164
6	7	5.297325	199.801636
7	8	4.671700	106.879286
8	9	4.001611	54.686179
9	10	2.989039	19.866576

Fig. 13. Bert Model perplexity and loss

Roberta Model Training Loss

```
: result_df_roberta
:
```

	Epochs	Loss	perplexity
0	1	7.140338	1261.854682
1	2	7.758777	2342.039501
2	3	6.209717	497.560503
3	4	6.036339	418.358813
4	5	5.136130	170.056319
5	6	3.752814	42.640897
6	7	3.157083	23.501941
7	8	3.760667	42.977096
8	9	3.178656	24.014448
9	10	1.446373	4.247682

Fig. 14. Roberta Model perplexity and loss

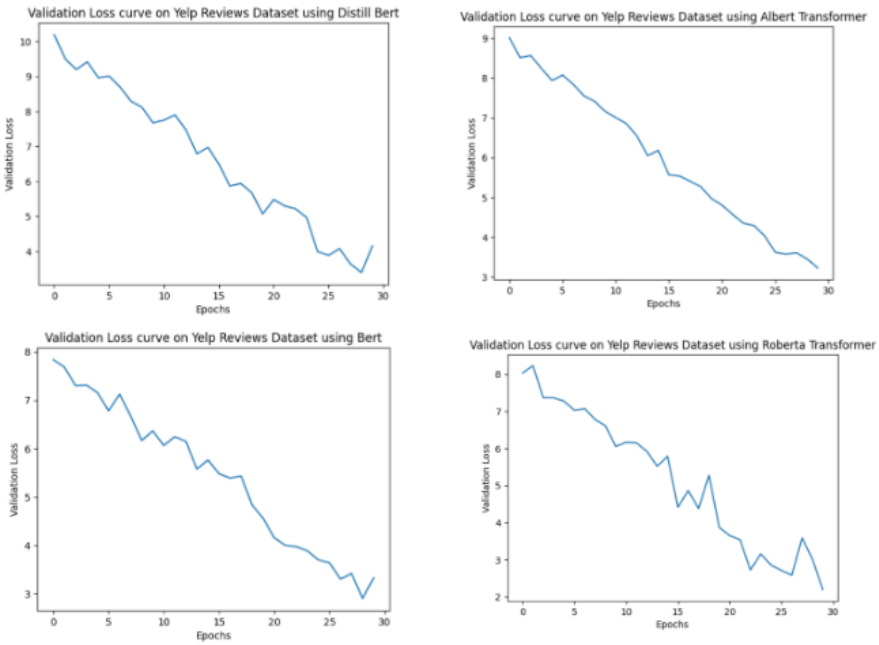

Fig. 15. Validation Losses for Transformer Model

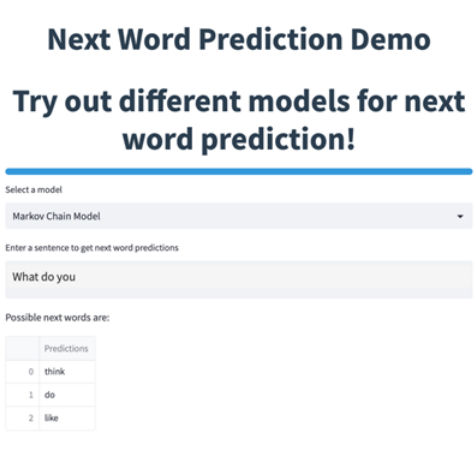

Fig. 16. Demo of the Markov Chain Deployment

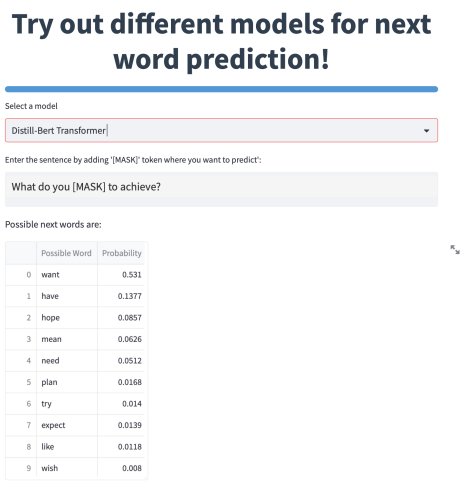

Fig. 17. Demo of the Distill-Bert Deployment

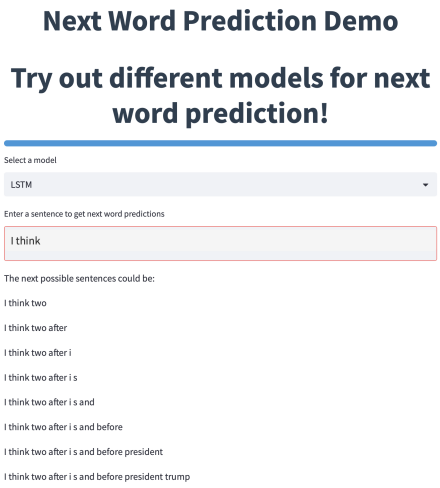

Fig. 18. Demo of the LSTM Deployment

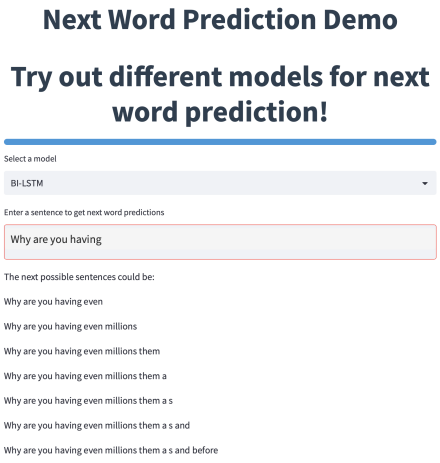

Fig. 19. Demo of the BiLSTM Deployment

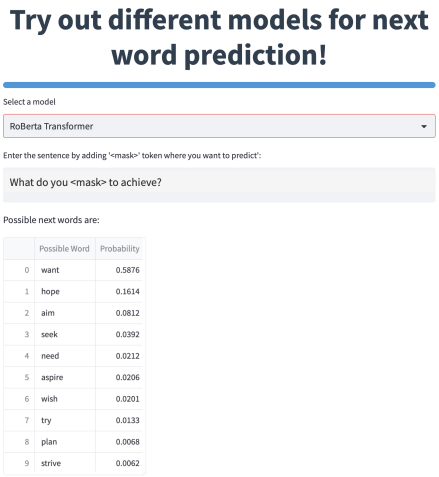

Fig. 20. Demo of the Roberta Deployment