Next Word Prediction using Multi-turn Dialogue Dataset

Kirthana Shri Chandra Sekar
chandrasekar.k@northeastern.edu

Srinidhi Aduri
aduri.s@northeastern.edu

Suja Ganesh Murugan
ganeshmurugan.s@northeastern.edu

***Abstract* — Next-word prediction plays a key role in bridging the gap between human communication and technological interaction, making digital experiences more seamless and intuitive. In this paper, we utilized advanced language models to construct an auto-complete system using daily conversation text as a dataset to predict the next word in a sentence. Our research involved analyzing 13,118 multi-turn dialogues to anticipate the most probable next word based on the preceding text. We implemented the Long short-term memory network to perform this text generation. To enhance performance, we undertook comprehensive fine-tuning, making adjustments to critical parameters, including the optimizer, batch normalization, and learning rate, in addition to other hyperparameters. Our review rigorously assesses the capabilities and limitations of the LSTM Model applied to a Multi-turn Dialogue Dataset, directing the path for future improvements in next word prediction technology.**

I. Introduction

A. Overview

Next word prediction is crucial in a fast-paced world where quick and effective communication is valued. Our research aims to use deep neural networks, specifically Long short-term memory (LSTM) networks known for their efficacy in handling sequential data and capturing long-term dependencies, to predict the next word accurately. The dataset we used captures the conversations that happen in daily life settings, with around eight speaker turns per dialogue on average and around 15 tokens per turn, reflecting the diversity encountered in real-time communication.

B. Motivation

Our research aims to enrich user interaction by providing immediate suggestions rooted in conversation context, thereby enhancing typing efficiency and minimizing errors in messaging apps and other text interfaces. Current next word prediction models struggle to understand the context of a conversation or text. Models often struggle to grasp the nuances of human language, including sarcasm, idioms, and cultural references, which can lead to inappropriate or inaccurate predictions. To address this issue, we train the model using a high-quality multi-turn open-domain English dialog dataset to significantly improve the accuracy and contextual relevance of predictive text systems in extended conversations.

C. Approach

Our main goal is to build a model using advanced machine learning algorithms, particularly focusing on the LSTM network. We aim to compare the performances of the models by training them with the Daily Dialog dataset for our experiments to determine the most robust and performant architecture that can predict simultaneous words in a conversation with high precision. We will construct and train the models using the prepared dataset, optimizing it for context-aware next word prediction in multi-turn dialogues. Finally, we test the model's performance using a set of metrics, such as accuracy, perplexity, and contextual relevance. Fig. 1 demonstrates the pipeline of our approach.

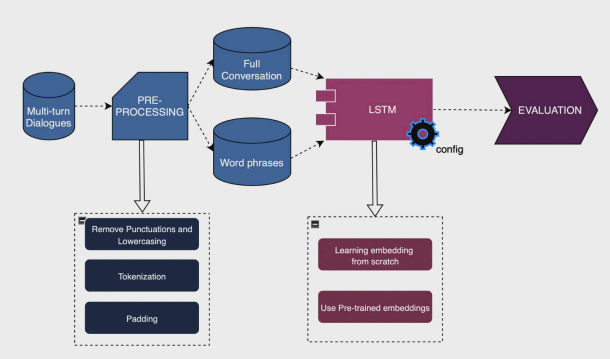


Fig. 1. *Methodology of Next-word Prediction Process*

II. Related Work

In recent years, next-word prediction has evolved from traditional methods like Support Vector Machines and Markov models to deep learning algorithms such as Recurrent Neural Networks (RNN) and Long-Short-Term Memory Networks (LSTM).

Many State-of-the-art word prediction techniques exist in NLP that leverage methods to enhance accuracy and efficiency, like Language Modelling, Part-of-speech (POS) tagging, morpho-syntactic agreement, and linear combination algorithms, to name a few [1]. There are many deep learning methods that are of high interest in NLP, such as - LSTMS, transformers, GRU, and BERT. Some LSTM methods include the Seq2Seq model based on Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM) cells [2] are widely used in tasks like machine translation as it is trained by maximizing the log probability of generating the correct translation given the source sentence. RNN Encoder-Decoder models, based on LSTM cells, use an attention mechanism that permits flexibility, where most relevant vectors are given the highest weights.

[3] Transformers can replace CNNs and RNNs with attention mechanisms that offer a more flexible and powerful approach for tasks involving sequence modeling and generation, making them preferred in many modern deep-learning architectures. [4] BERT models are widely used language models for text generation and prediction. Its comprehensive approach to contextual understanding through attention mechanisms, coupled with sub-word tokenization, makes it superior for a wide range of natural language processing tasks. [5]

III. Dataset and Methodology

A. Dataset

In this study, we utilize the publicly accessible dataset DailyDialog [6] for our analysis. The dataset contains 13,118 dialogues with an average of eight speaker turns per dialogue. Each dialogue has around 15 tokens per turn, and they are separated by '__eou__' (end of utterance) notation within each conversation.

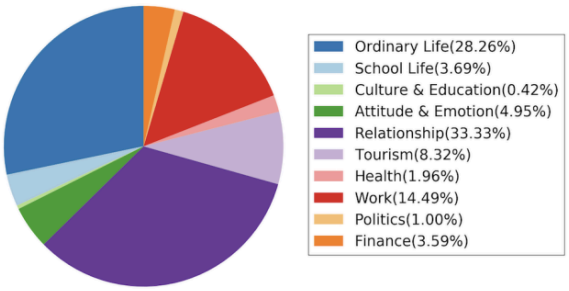
Table 1. *Multi-turn Dialogue Example*

Turns	Dialogue Text
Turn 1	'Would you mind waiting a while'
Turn 2	'Well , how long will it be'
Turn 3	'I'm not sure . But I'll get a table ready as fast as I can'
Turn 4	'OK . We'll wait'

B. Data Preprocessing

The DailyDialog dataset has categorized its conversations under various topics such as Tourism, Health, Work, etc.

Fig. 2. *Topic distribution of DailyDialog*



To train our model, we use conversations from all the topics and predict the last word in each conversation. The model struggles to learn and generalize from the diverse conversational data effectively. Therefore, we primarily focused on the topic with the largest conversations, 'Relationship,' which has around 4391 conversations and 32258 utterances.

We then perform selective preprocessing on the conversations by removing the punctuations, the '__eou__' notations, and case folding in each utterance, which is suitable for autocomplete systems. We further tokenize each processed utterance, looping over all the conversations present in our dataset, creating a corpus of processed words. Finally, we create sequences for model training. These sequences are encoded as integers, where each word in each sequence is assigned a unique integer. Using the Tokenizer class in the Keras library, we then convert the sequences of words to sequences of integers. These new sets of integer sequences are further split into input and output elements using array slicing; we have used the Five-Word-In, One-Word-Out sequence format for our experiments. Hence, out of the 6 elements in a sequence, the first 5 would be considered as input, whereas the 6th word is considered as output.

C. LSTM Model

Our model architecture is composed of several key components tailored to address the challenges of next-word prediction in natural language processing. These components include embedding layers, LSTM layers, dense layers, and an optimization algorithm, each contributing uniquely to the model's performance. The first layer in our model is the embedding layer, which transforms input words into dense vector representations. Two variations of the embedding layer were explored:

1. **Embeddings built from Scratch:**
   Here, embeddings are learned from scratch during the training process, allowing the model to develop word representations specifically tuned to the dataset and task.

2. **Pre-trained Embeddings:**
   We also experimented with pre-trained word embeddings, including Word2Vec and GloVe. These embeddings are advantageous as they bring prior semantic and syntactic knowledge from large text corpora, potentially accelerating the learning process and improving model robustness.

Following the embedding layer, our model includes LSTM layers. LSTMs are particularly suited for sequence prediction tasks due to their ability to maintain long-term dependencies in data. The architecture allows the model to learn complex patterns and deeper relationships across the text, essential for accurate next-word prediction [8], as the model can effectively leverage contextual information.

Post-LSTM processing, the model incorporates two dense layers. The first dense layer uses the ReLU activation function to introduce non-linearity, enabling the model to learn complex patterns and relationships in the data. The final layer employs a softmax activation function, which converts the outputs from the previous layers into a probability distribution over the possible next words in the vocabulary. This layer is critical for determining the most likely subsequent word based on the learned context.

To enhance the performance of our models, we employ the Adam optimization functions [9], with a 0.001 learning rate, to assess their impact. Our objective is to identify the most effective LSTM configuration that optimizes both accuracy and generalizability across the dataset through a comprehensive exploration of architectural dimensions and hyperparameter tuning. The Adam optimizer is employed to adjust the weights of the network during training. Chosen for its efficiency and effectiveness, Adam facilitates faster convergence and handles sparse gradients well, which is beneficial given the variable nature of text data in next-word prediction tasks.

IV. Experimentation and Results

A. *Experimental Setup*

We implement all our models using TensorFlow from the ground up, training them on the training set over 30 epochs with batch_size set to 32 and applying Categorical Cross-Entropy as the loss function. We assess the model's efficacy by reporting perplexity across each configuration in the test set. To prevent overfitting, we incorporate early stopping, evaluating model performance against the validation set. Our aim is to create models that not only achieve high accuracy but also display robustness and consistency across various configurations, ensuring their applicability and reliability in practical settings.

To efficiently track the progress of our training and pinpoint the most effective configurations, we utilize MLFlow. This tool records parameters and key metrics such as training accuracy, loss, validation accuracy, validation loss, test accuracy, top-5 accuracy and perplexity. By monitoring these metrics, MLFlow aids in managing overfitting and identifying frameworks that deliver the best performance.

B. *Results*

i. In our initial experiments, we train our LSTM model by considering the entire conversation as the input, and the model is trained to predict the concluding word. We implement the embedding from scratch and observe the performance using the parallel coordinates plot. It is observed that combinations of embedding dimensions, number of units and layers led to different levels of perplexity. The models with higher embedding dimensions and a moderate number of units performed better than others but not an overall good performance in general as Fig 3. depicts.

Larger embedding dimensions usually provide a richer representation of the words, potentially capturing more nuances in meaning. However, it seems that beyond a certain point, increasing the embedding dimension does not significantly decrease perplexity, which could indicate a plateau effect where additional dimensions do not contribute to performance improvements.

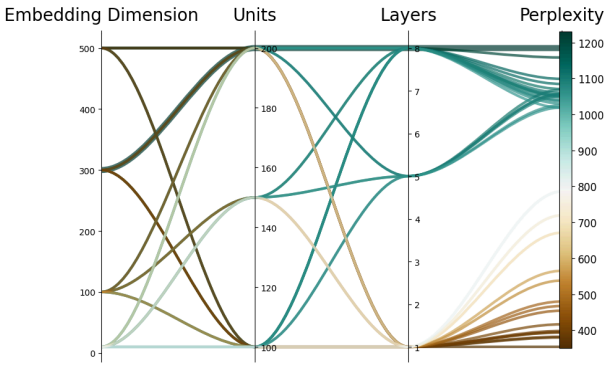


Fig. 3. *Performance of model with the embedding layer made from scratch on the compiled dataset of utterance.*

ii. To achieve lower perplexity and better model performance, careful tuning and understanding of the complex relationships between these model parameters is necessary. To improve the robustness of the model, we adopt the pre-trained word embedding Word2Vec of embedding dimensions 300 and 500 and observe its performance.

Word2Vec embeddings are typically pre-trained on a vast corpus of text data. This pre-training allows the embeddings to capture a wide range of semantic and syntactic relationships between words, which a model with embeddings trained from scratch may not learn unless trained on similarly large and diverse datasets.

However, Fig. 4. shows that there is no significant improvement in the model's performance as compared to the previous model, where the embedding layer was made from scratch. Therefore, we decided to employ different input/output pairs, specifically five word-in and one word-out sequences, to further improve the language modeling prediction.

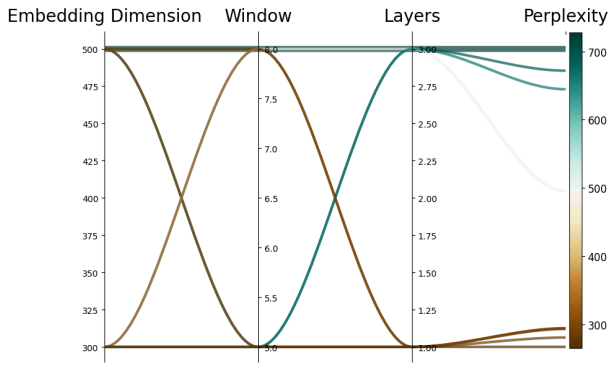


Fig. 4. *Performance of model using pretrained word embedding Word2Vec on the compiled dataset of utterance.*

iii. The Five-Word-In, One-Word-Out sequence method is part of a broader set of techniques for training language models to predict the next element in a sequence.

Fig. 5. illustrates the performance of a next-word prediction model using the Five-Word-In, One-Word-Out sequence method with an embedding layer crafted from scratch, charting the influence of the embedding dimension, the number of LSTM layers, and the number of units on model perplexity.

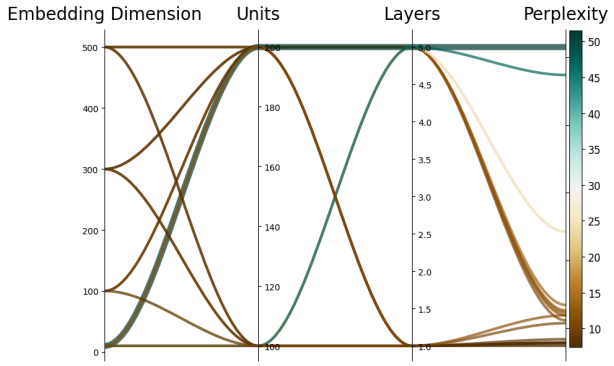


Fig. 5. *Performance of model using word embedding layer made from scratch on a Five-In, One-Out Sequence input format*

We train the model with four different embedding dimensions, which are 10, 100, 300, and 500. There is a trend showing that as the embedding dimension increases, the perplexity tends to decrease to a point, suggesting that a richer word representation (higher dimension) is beneficial up to a certain level. Along with the different embedding dimensions, we also tried different unit sizes, which refer to the number of neurons in each LSTM layer, namely 100 and 200. The model achieves lower perplexity with 100 units; however, this trend does not show a simple linear relationship. There is an optimal middle ground that yields lower perplexity. In contrast, too few or too many units do not result in the best model performance, potentially due to underfitting or overfitting, respectively.

The number of LSTM layers ranges from 1 to 5. With the increase in layers, the model does not appear to contribute significantly to performance improvement, which might be due to the model's increased complexity and the vanishing gradient problem that can arise with many layers in LSTMs.

iv. To reduce the computational load and improve the efficiency of the model, we trained the model using Word2Vec embeddings as opposed to the embedding layer created from scratch in our previous experiment. Fig. 6. elucidates the decreased range of perplexity scores, indicating improved performance of the model.

Word2Vec's ability to provide rich, pre-trained word vectors significantly enhanced the model's learning process, enabling it to predict the next word with higher accuracy and speed. However, the actual improvement in model performance would depend on how well the Word2Vec embeddings align with the specific linguistic characteristics of the text data used for next-word prediction.

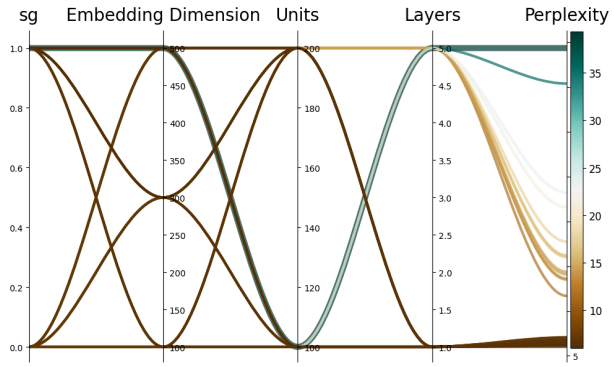


Fig. 6. *Performance of model using Word2Vec embeddings on a Five-In, One-Out Sequence input format.*

v. In the context of next-word prediction, the choice between these embeddings could influence the model's ability to generalize and predict accurately. Word2Vec might offer more nuanced predictions based on the specific contexts it has been trained in, while GloVe might provide a more robust foundation of word associations that can be advantageous when predicting words based on broader usage patterns. Fig. 7 implies that the model trained with GloVe embeddings displays trends similar to those noted with Word2Vec. GloVe embeddings, which are trained on word-word co-occurrence statistics, provide a robust semantic foundation, potentially offering a

better performance on tasks that benefit from such representations than embeddings created from scratch.

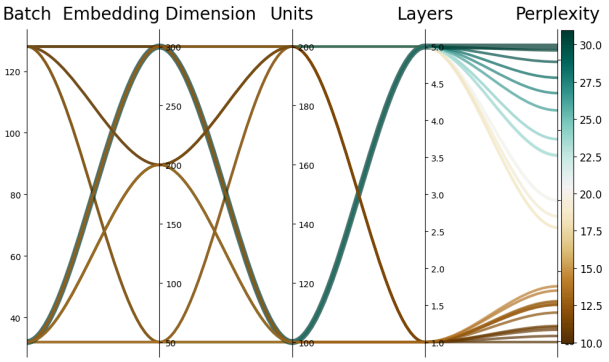


Fig. 7. *Performance of model using GloVe embeddings on a Five-In, One-Out Sequence input format.*

Model	Embed_ dim	Layers	Units	Perplexity	Test Acc.	Top-5 Acc.
Word2Vec (sg=1)	500	1	100	**6.00**	76.78	**83.22**
Word2Vec (sg=1)	300	1	200	6.14	76.30	83.10
Word2Vec (sg=1)	300	1	100	6.20	**77.00**	82.84
Word2Vec (sg=1)	500	1	200	6.34	75.90	82.88
Word2Vec (sg=1)	300	1	200	6.41	74.84	83.08

Fig. 8. *Summary of Best-Performing Models*

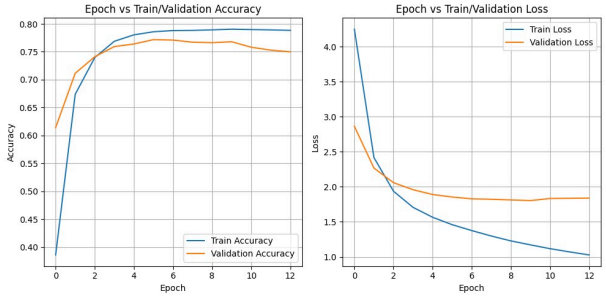


Fig. 9. *Best Performing model's train versus validation plot. Left: Accuracy, Right: Loss*

Summary results. The combined observations suggest that model (iv), which uses Word2Vec embeddings on a Five-Word-In, One-Word-Out Sequence input format, outperforms other models. The data preparation process also has a substantial impact on model training and performance. Sequences created using the Five-Word-In, One-Word-Out format are particularly suitable for next-word prediction tasks, as they provide the model with a clear structure to learn from: given the context of the preceding words (input), it learns to predict the following word (output).

In this context, the sequencing's role is pivotal for training the next-word prediction models. By using five words as the input context for the model to predict the sixth word, the model is trained to understand and learn the probability of a word given its immediate linguistic context. This methodology aligns with the n-gram approach in traditional language modeling but benefits from the depth of neural network processing, enabling the model to capture more complex patterns in language.

The model with 500 embedding dimensions and 100 units achieved the lowest perplexity of 6.00, indicating it predicts the next word with the least uncertainty and highest confidence. On the other hand, the model with 300 embedding dimensions and 100 units has the highest test accuracy at 77.00%. This suggests that while larger embedding dimensions may decrease perplexity, they do not always translate into higher test accuracy. Across all models, the top-5 accuracy does not vary significantly, staying above 82% for all configurations. The model with 500 embedding dimensions and 100 units tops this metric with 83.22% accuracy. It appears that larger embedding sizes may slightly improve the model's ability to list the correct next word within the top five predictions.

V. Conclusion

In this paper, we explore the LSTM model for next word prediction by using different approaches in preprocessing as well as neural network architectures to create an optimal model. We conducted a systemic comparative analysis between utterances and sequences created from the corpus. In next-word prediction tasks, the goal is often to predict the continuation of a text given its beginning. The Five-Word-In, One-Word-Out method trains models to predict subsequent words using pattern recognition from fixed-length inputs. The combined observations suggest that, along with optimal LSTM configurations and embedding settings, it can yield a robust model for next-word prediction.

In future work, we aim to explore the prediction of the next bi-gram and n-gram sequences as an interesting direction to investigate further.

REFERENCES

[1] Aliprandi, Carlo & Carmignani, Nicola & Deha, Nedjma & Mancarella, Paolo & Rubino, Michele & Srl, Synthema & Pisa, & Italy,. (2008). Advances in NLP applied to Word Prediction.

[2] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'14). MIT Press, Cambridge, MA, USA, 3104–3112.

[3] Cho, K., Learning Phrase Representations using RNN Encoder - Decoder for Statistical Machine Translation, arXiv e-prints, 2014. doi:10.48550/arXiv.1406.1078

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need.

[5] Y. Qu, P. Liu, W. Song, L. Liu and M. Cheng, "A Text Generation and Prediction System: Pre-training on New Corpora Using BERT and GPT-2," 2020 IEEE 10th International Conference on Electronics Information and Emergency Communication (ICEIEC), Beijing, China, 2020, pp. 323-326, doi:10.1109/ICEIEC49280.2020.9152352.

[6] Li, Y., Su, H., Shen, X., Li, W., Cao, Z., and Niu, S., “DailyDialog: A Manually Labeled Multi-Turn Dialogue Dataset”, arXiv e-prints, 2017. doi:10.48550/arXiv.1710.03957.

[7] Mikolov, T., et al. (2013). "Efficient Estimation of Word Representations in Vector Space."

[8] Hochreiter, S., & Schmidhuber, J. (1997). "Long Short-Term Memory."

[9] Kingma, D. P., & Ba, J. (2014). "Adam: A Method for Stochastic Optimization."