IST-Africa

IST-Africa 2022 Conference Proceedings
Miriam Cunningham and Paul Cunningham (Eds)
IST-Africa Institute and IIMC, 2022
ISBN: 978-1-905824-69-4

Stacked Language Models for an Optimized Next Word Generation

Ednah Olubunmi ALIYU[1] and Eduan KOTZÉ[2]
Department of Computer Science and Informatics, University of the Free State, Bloemfontein, South Africa
Email: Aliyu.EO@ufs.ac.za, kotzeJE@ufs.ac.za

Abstract: Next word prediction task is the application of a language model in natural language generation that deals with generating words by repeatedly sampling the next word conditioned on the previous choices. This paper proposes a stacked language model for optimized next word generation using three models. In stage I, the meaning of a word is captured through learn embedding and the structure of the text sequence is encoded using a stacked Long Short Term Memory (LSTM). In stage II, a Bidirectional Long Short Term Memory (Bi-LSTM) stacking on top of the unidirectional LSTM encodes the structure of the text sequences, while in stage III, a two-layer Gated Recurrent Unit (GRU) is used to capture text sequences of data. The proposed system was implemented using Python 3.7, Tensorflow 2.6.0 with Keras and a Nvidia Graphical Processing Unit (GPU). The proposed deep learning models were trained using the *Pride and Prejudice* corpus from the Project Gutenberg library of ebooks. The evaluation was performed by predicting the next 3 words after considering 10 sets of text sequences. From the experiment carried out, the accuracy of the two-layer LSTM model measured 83%, the accuracy of the Bi-LSTM stacking on unidirectional LSTM model measured 79%, and the accuracy of the two-layer GRU model measured 81%. Regarding predictions, the two-layer LSTM predicted the 10 sequences correctly, the Bi-LSTM stacking on unidirectional LSTM predicted 8 sequences correctly and the two-layer GRU predicted 7 sequences correctly.

Keywords: Natural Language Generation, Long Short Term Memory, Bidirectional, Gated Recurrent Units

1. Introduction

Text predictors or succeeding-word predictors have completely revolutionized the world of text messages and chats by making communication a lot easier. Moreover, they help to avoid unnecessary and excessive typing Tamizharasan *et al*. [1]. The main aim of text prediction, according to HaCohen-Kerner *et al.* [2], was to increase typing speed and reduce writing errors (particularly helpful for persons with dyslexia).

The evolution of language models in recent years has led to the fast-growing amount of data stored in databases, data warehouses and other data repositories Zhou [3]. Since the origin of Natural Language Processing (NLP), which teaches computers how to understand and generate human language, the foundation lies in a number of disciplines, such as computer science, linguistics, mathematics, data science, artificial intelligence and psychology. Modern NLP applications, such as search engines, question answering, natural language assistants such as Siri by Apple, and translation systems use NLP technology to understand a query and find pattern matching documents Chowdhury [4]. The main driver of this change, according to Iosifova *et al.* [5], was the emergence of language models.

One of the methods used to compute the probability of predicting the next word given the sequence of texts is employing n-gram models. Stolcke *et al.* [6] pointed out that one of the limitations of using n-grams paths with Markov Chains is the lack of memory. A

Markov Chain only conditions on a fixed window (that is, N previous words as n-grams), making it impossible to model large word windows. This problem led to research on a neural network model. For instance, Recurrent Neural Networks (RNNs) are capable of conditioning the model on all previous words in the corpus. However, RNNs gradually vanish as they propagate to earlier time-steps Chaubard *et al.* [7].

In the quest to solve the vanishing gradient problem, more variants of RNN, such as Long Short Term Memory (LSTM) and Gated Recurrent Units (GRU), were developed to use deep learning with a network of artificial cells that manage memory, making them suitable for text prediction Tuan *et al.* [8]. Later in 2019, Devlin *et al.* [9] introduced Bidirectional Encoder Representations from Transformers (BERT). BERT uses transformer architecture and attention to learn contextual relations between words in a text, thus presenting state-of-the-art results in a wide variety of NLP tasks. However, the transformer model was first proposed in a paper titled *Attention Is All You Need* by Vaswani *et al.* [10], where end-to-end memory networks are based on a recurrent attention mechanism instead of a sequence-aligned recurrence and have performed well on simple-language question answering and language modeling tasks.

Based on the achievement mentioned above, a stacked language model for optimized next word generation is inspired by the work of Elmogy *et al.* [11]. They employed a Bi-LSTM to generate text. Word2vec word embedding and one-hot representation were used to vectorize the input sequences using the datasets of the *Alice in Wonderland* and *Twilight* stories. In addition, Graves *et al.* [12] highlighted that the depth of the network is more important than the number of memory cells in a given layer to model skill.

In this paper, three models will be established, namely the stacking of two layers LSTM, Bi-LSTM stacking on top of unidirectional LSTM layers and two layers GRU with a learn word embedding. Furthermore, optimization parameters such as learning rate, dropout, L1 regularization and adaptive momentum estimation with categorical cross-entropy will be used to compare the three algorithms on the best convergence and the most accurate predictor model for next word prediction.

1.1 Related Works

Different models and architectures have been proposed in literature to address the next word prediction problem. Wang *et al.* [13] suggested a convolutional neural network, named genCNN, for word sequence prediction. Their model exploits both short and long range dependencies. The RNN-based approach for next word prediction in Assamese phonetic transcription was introduced in Barman and Boruah [14] with an accuracy of 88.20% for Assamese text and 72.10% for phonetically transcribed Assamese language. LSTM and Markov Chains with the combination of two approaches were employed by Shakhovska *et al.* [15]. The Markov chains were faster and produced better results. The hybrid model presented adequate results but it was slower. Elmogy *et al.* [11] presented a bidirectional recurrent neural network (Bi-LSTM) to develop a text generation problem with word2vec word embedding Mikolov *et al.* [16] and one-hot representation for the input sequences. Their results show that Bi-LSTM is well-suited for making next word predictions. Ambulgekar *et al.* [17] proposed next word prediction using LSTM. The input sequences given to the language model consisted of 40 characters. The model was trained for 5 epochs with 56% accuracy using Tensorflow Abadi *et al.* [18] and Keras Chollet [19]. In the next section, mathematical concepts behind LSTM and Bi-LSTM will be discussed.

1.2 Long Short Term Memory (LSTM)

Many variants of RNN have been proposed. One of these variants is the Elman Network containing a single, self-connected hidden layer Wen [20]. In practice, the range of context that a vanilla RNN can process is quite limited and results in a vanishing gradient problem.

The methods proposed to solve the problem were Gated Recurrent Units and Long Short Term Memory. In this paper, both models will be employed.

Figure 1 shows a typical LSTM cell with only one memory block, denoted as C_t. Along with the memory block, there is input gate i_t, forget gate f_t, and output gate o_t, respectively. x_t is the current time step input and h_{t-1} is the hidden layer representation at the previous time step.

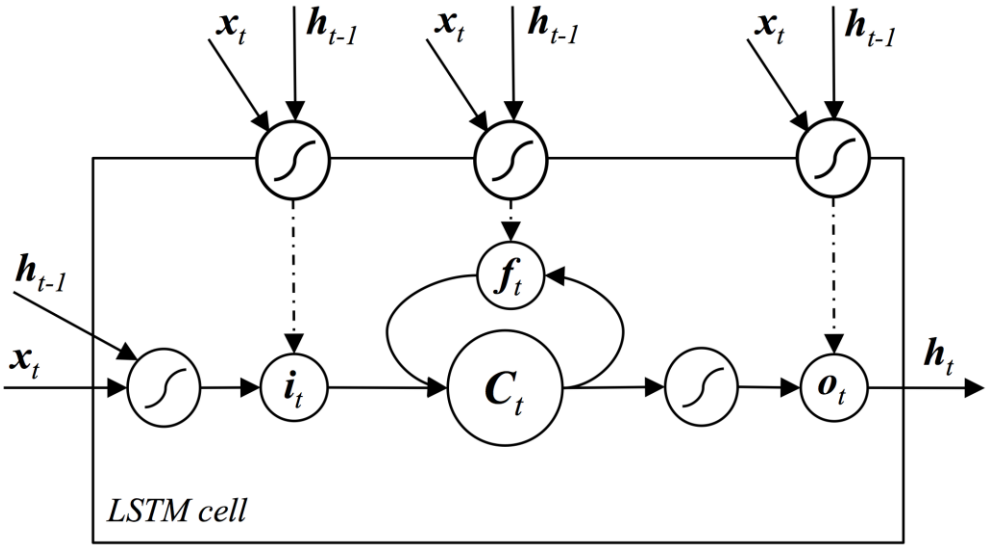

Figure. 1. LSTM Internal Structure [20]

The multiplication units joining each memory block provide the cell with continuous analogies of write, read, and reset. These multiplication gates allow LSTM memory cells to store and access information over long periods, hence alleviating the vanishing gradient problem.

To compute the forward pass of an LSTM, the gate activation level needs to be calculated as given below:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1}) \quad (1)$$

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1}\right) \quad (2)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1}) \quad (3)$$

where σ is a sigmoid function and W_x the weight matrices. The proposed cell value at the current time step is given by:

$$\widehat{c_t} = \tanh(W_{xc}x_t + W_{hc}h_{t-1}) \quad (4)$$

By combining equation (4) with equations (1) and (2), the new cell value can be calculated via element-wise multiplication of the gate and cell values to be:

$$c_t = f_t \odot c_{t-1} + i_t \odot \widehat{c_t} \quad (5)$$

Finally, the hidden layer representation is computed by multiplying the output gate and cell value element-wise:

$$h_t = o_t \tanh \odot (c_t) \quad (6)$$

Working backwards to compute the error derivative of the hidden layers, the Back-Propagation Through Time (BPTT) algorithm is adopted to calculate the gradients.

1.3 Bidirectional Network

In Wen [20], the Bidirectional Network encodes the input sequence in both directions by two separate LSTMs and connects both the forward and backward hidden layers to a single output layer, as shown in Figure 2. The output at time step t is calculated as:

$$Y_t = \sigma(w[h_{tf}, h_{tb}] + b) \tag{7}$$

where h_{tf} and h_{tb} represent the hidden state at the forward and backward directions, respectively, w represents the weights associated with them, b represents the bias, which helps the model to fit to the given data, and σ is the activation function used to add non-linearity to the network.

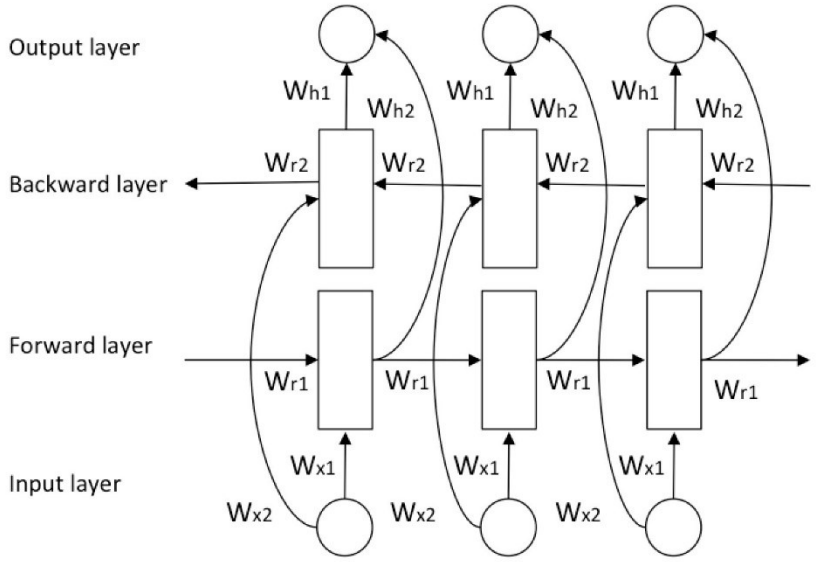

Figure 2. The Unfolded Bidirectional Network

However, *Cui et al.* [21] further explained that the concept of bidirectional LSTMs (biLSTM) originated from bidirectional RNNs (biRNN). Wen [20] also asserted that the forward pass of the network is the same as an RNN. However, there is one difference, namely that the input sequence is presented in the opposite direction so that the output layer can only be updated when both the forward and backward hidden layers are obtained. Similarly, the backward pass is initiated in the same manner that the RNN trained with BPTT; the only difference is that the gradients of the output layer are computed first, then fed back to the two RNNs.

1.4 Stacked Bidirectional and Unidirectional LSTM Networks

Stacked LSTMs or Deep LSTMs were introduced by Graves *et al.* [12] in their application of LSTMs to speech recognition. They found that to model skill, the depth of the network was more important than the number of memory cells in a given layer. According to Cui *et al.* [21], a stacked LSTM architecture can be defined as networks with several stacked LSTM hidden layers, in which the output of an LSTM hidden layer will input into the subsequent LSTM hidden layer. Figure 3 depicts a typical two-stacked LSTM layer.

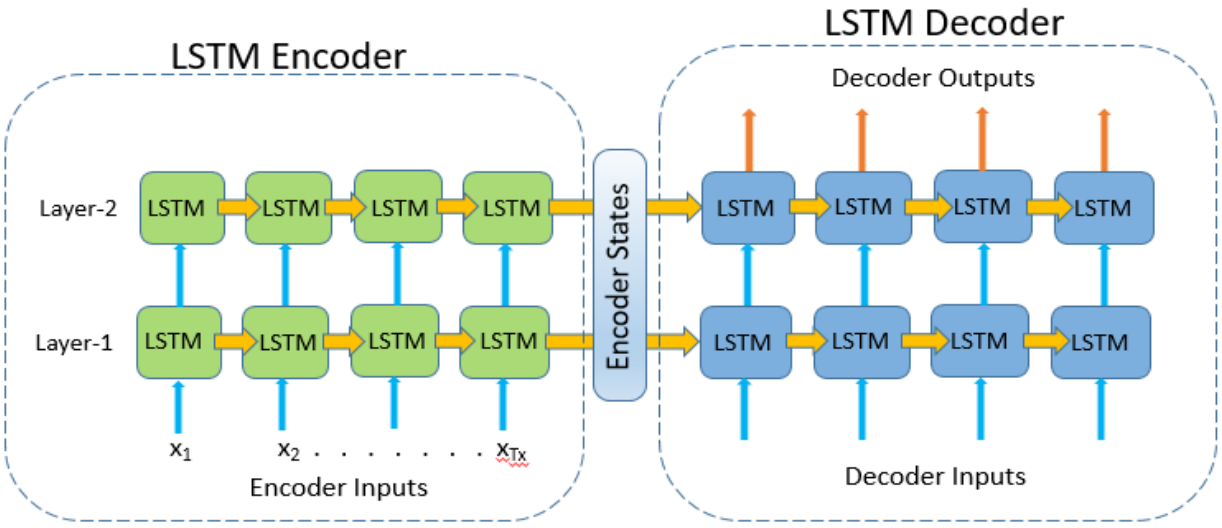

Figure 3. LSTM Network with 2 Stacked Layers [22]

When the layers are stacked, the outputs (cell states) of the first layer LSTM cells of both the encoder and the decoder are passed to the second layer LSTM cells as input Battula [22]. Cui *et al.* [21] found that an LSTM layer, which is fit for capturing forward dependency, is a better choice for the last (top) layer of the model.

1.5 Gated Recurrent Units (GRU)

Xu and Hu [23] proposed a Gated Recurrent Unit (GRU) neural network to process sequence information, such as thermal sequence, as opposed to LSTM neural networks. GRUs contain two gate structures, namely the reset gate and the update gate, where x_t represents the input data at the current time, h_t depicts the hidden state of the GRU at the current moment, and h_{t-1} is the previous hidden state (as shown in Figure 4). The update gate z_t controls current input x_t and the previous hidden state h_{t-1}, where the value of z_t squashes between the range 0 to 1 are defined below:

$$z_t = \sigma(W_{zx}x_t + W_{zh}h_{t-1} + b_z) \quad (8)$$

where σ is a sigmoid activation function, W_{zx} and W_{zh} are the update gate weight values and b_z is the bias.

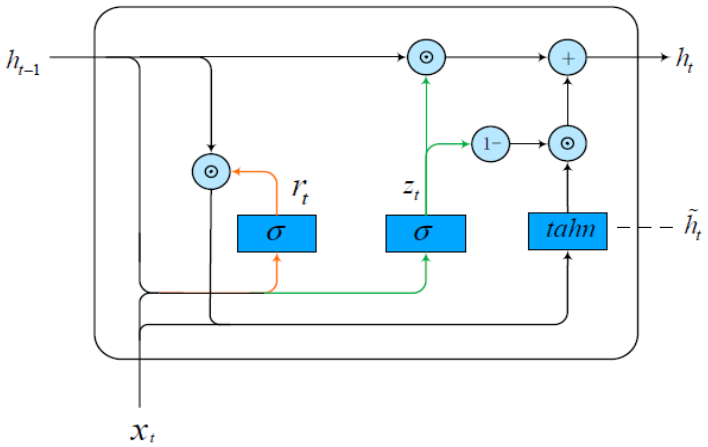

Figure 4. GRU Internal Structure [23]

The reset gate r_t determines the influence of the hidden state h_{t-1} on the hidden state h_t to forget. It is formulated as follows:

$$r_t = \sigma(W_{rx}x_t + W_{rh}h_{t-1} + b_r) \quad (9)$$

The memory content $\bar{h_t}$ which uses the reset gate to store the relevant information from the past, is computed as follows:

$$\bar{h_t} = \tanh(W_{hx}x_t + W_{hh}(r_t h_{t-1}) + b_h) \quad (10)$$

where W_{hx} is the weight between the input state and the hidden state, W_{hh} is the weight at the hidden state, and b_h is the bias.

The final memory at the current time step is computed as follows:

$$h_t = z_t h_{t-1} + (1 - z_t)\bar{h_t} \quad (11)$$

As we mentioned earlier, the forward and backward passes of the network are initiated with BPTT.

2. Objectives

The specific objectives of this research are to (i) design a word prediction model using state-of-the-art deep neural language models such as stacked LSTM, GRU and biLSTM, and (ii) implement the next word prediction techniques and compare them in terms of predicting the next word.

3. Methodology

In this research, the next word generation techniques comprise stacked LSTM, Bi-LSTM stacked over an LSTM layer and stacked GRU. The *Pride and Prejudice* corpus from Project Gutenberg library is available from https://www.gutenberg.org/browse/scores/top. It is preprocessed and tokenized using the *Tokenizer* package from Keras to train the proposed model. The Sequential Model API from Keras is employed to capture the meaning of a word through learn embedding along with other parameters such as memory cell, dense layer, and activation function, respectively, for each of the proposed deep learning techniques. Greedy Decoding Algorithm See [24] and Keneshloo *et al.* [25] is adopted in the inference to take the highest scoring item at each stage using the *argmax* mathematical function provided in *numpy*. The designed system is implemented using Python 3.7, Tensorflow 2.6.0 with Keras and a Nvidia Graphical Processing Unit (GPU).

4. Technology Description

The research documentation is well understood and enables both experts and novices to comprehend the research background based on the following:

- The research findings are explained clearly.
- The limitations of existing methods are well articulated through a literature review.
- The discovery is coherent as experiments are performed.
- The proposed model and existing methods are compared.
- The results are well explained.

5. Development

Three models are proposed and developed for next word generation (Figure 5).

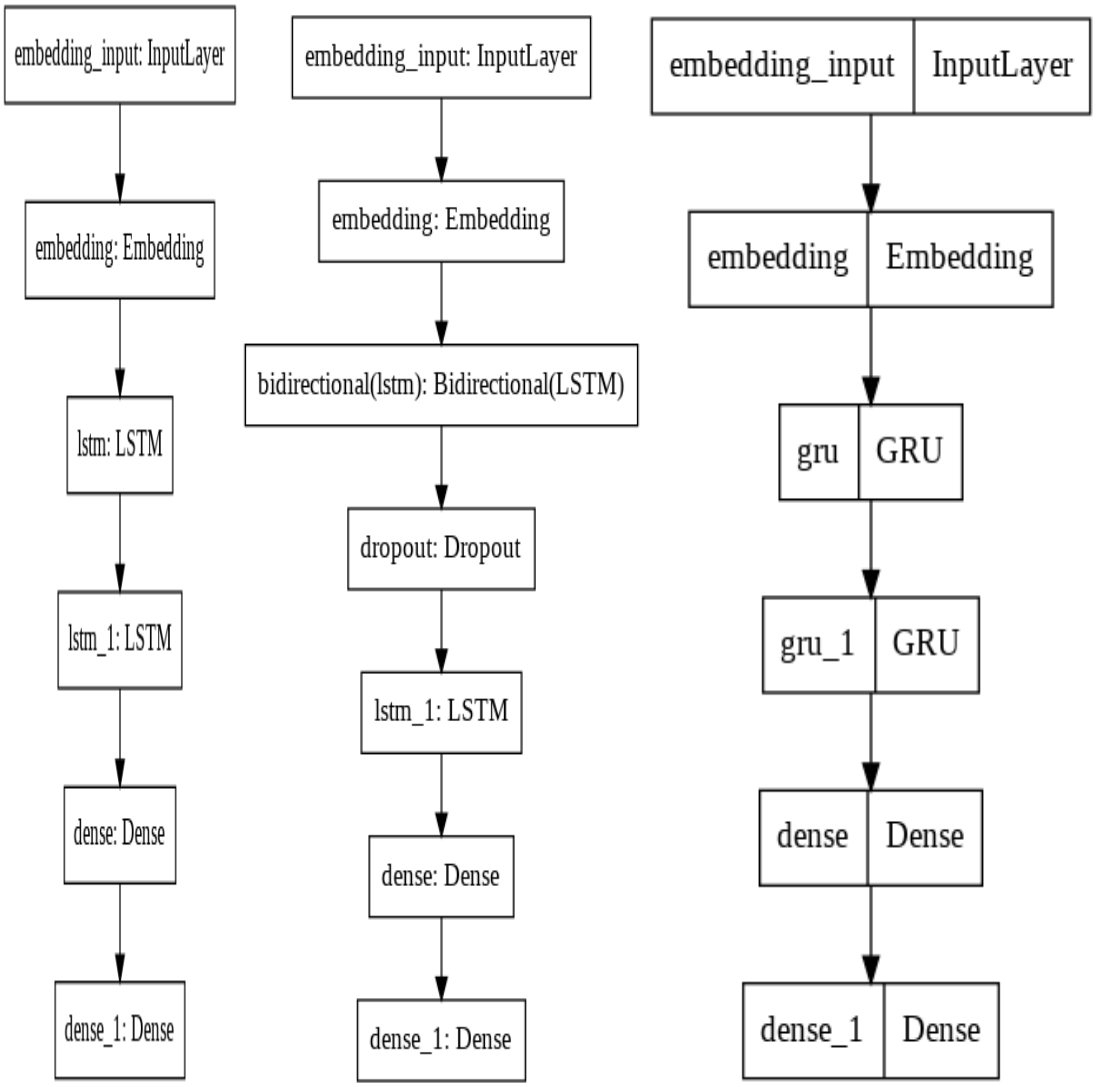

Figure 5. Proposed Stacked Next Word Generation Model

Model 1 is a stacked LSTM that captures the structure of the text sequences. For the text sequences to have meaning after tokenization, the best vector representation of each word is learned during the model training process (that is, learned using the LSTM model). To stack the LSTM layers, one needs to change the configuration of the previous LSTM layer by setting *"return_sequences"* on the layer to be true. Two dense layers are created. The first dense layer takes units (memory cells) and activation functions as arguments, then passes the activation functions as *"ReLU"* which converts negative values to 0. The other dense layer takes the number of units to be vocabulary and infers the most probable word to generate as an output within the vocabulary.

Model 2 is developed by stacking a Bi-LSTM layer over an LSTM layer. As mentioned in previous sections, Bi-LSTM can use both forward and backward dependencies. This may

help the network to capture some dependencies which otherwise would not have been captured by the standard (forward) LSTM.

Model 3 is a Stacked GRU that follows the same structure as Model 1, but with a reduced number of gates, hence simplifying the network architecture. "*ReLU"* activation function is used at the dense layer, which is also adopted for Model 1 and Model 2, to convert negative values to 0 and to ignore positive values.

5.1 Experiments and Results

The dataset contained 701932 words and a unique vocabulary of 7244 words. The split between the training dataset and testing dataset is 80 by 20. We performed preprocessing (text cleaning and tokenization) and converted the text to sequences. Here, four words were inserted into the list. Three words were the input and the fourth word the output.

Having encoded the input sequences, we had to separate dependent and independent variables using array slicing. The class vector was transformed into binary matrix representation using loss function as categorical entropy. This is necessary so that the model learns to predict the probability distribution for the next word. The ground truth from which to learn from is 0 for all words except the actual word that comes next.

All three deep neural language models were created using the Sequential Model APIfrom Keras. The learned embedding takes 3 arguments, namely input dimension (size of vocabulary), output dimension (vector space for each word chosen to be 10) and input length. Two layers LSTM with 1000 memory cells each were defined. A dense, fully connected layer with 1000 neurons and activation functions as "*ReLU*", which converts negative values to zero and ignores positive values. Another dense layer is created in which the number of units will be the size of the vocabulary because output is needed within this vocabulary. A *Softmax* activation function is used to scale numbers into probabilities.

For the deep learning training step, categorical cross entropy is used for multi-class classification tasks with Adaptive Moment Estimation (*Adam)* optimizer Kingma and Ba [26] (a combination of AdaGrad and Rmsprop). The parameters for the *Adam* optimizer to update the weights iteratively based on training data are as follows: learning rate 0.001, dropout 0.2 and L1 regularization to prevent overfitting on the training data. The model was fitted on the training data (n = 80%) and we set the maximum number of epochs at 100. We allowed the model to stop after 79 epochs if there was no improvement in validation loss.

6. Results

After training the models, we tested each model with the testing dataset (n= 20%). However, our goal with training was to minimize the loss during model training. The parameters "*min_delta"* and "*patience"* were checked at the end of each epoch. Once the loss reached an optimal level the best model was saved and the training terminated. The summary of performance of the best trained model, including the loss and accuracy evaluated from the training data at the end of each batch update, is presented in Table 1.

Table 1. Training Loss and Accuracy Results

Model	**Epochs**	**Loss**	**Accuracy**	**Validation Loss**	**Validation Accuracy**
Stacked LSTM	79	0.4048	0.8302	20.1037	0.0879
BiLSTM/LSTM	100	0.9282	0.7956	19.1344	0.0855
Stacked GRU	100	0.9347	0.8100	17.8198	0.0772

6.1 Comparison of the Research Results with Existing Technique

We considered Elmogy *et al.* [11] for the comparative study based on the following factors: topic theme, vocabulary (vocab) size, deep learning model, number of epochs, training loss and training accuracy, as depicted in Table 2.

Table 2. Current Study Versus Existing Work

	Topic Theme	**Vocab Size**	**Deep Learning Model**	**Number of Epochs**	**Training Loss**	**Training Accuracy**
Existing Work	Alice in Wonderland	2984	Bi-LSTM	120	0.0079	0.9989
	Twilight	15,626		80	0.0354	0.9812
Current Study	Pride and Prejudice	7244	Stacked LSTM	100	0.4048	0.8302
			biLSTM/LSTM		0.9282	0.7956
			Stacked GRU		0.9347	0.8100

As we can infer from Table 1, the number of epochs was set to 100. Once the parameter setup discovered the loss was no longer decreasing, the training terminated, which resulted in 79 epochs for stacked LSTM with highest accuracy of 83% with 0.4048 loss. Solawetz [27] stated the smaller the value of loss function, the better the model.

Table 2 shows that the current study takes advantage of different hyperparameter configurations (namely stacked LSTM, biLSTM and stacked GRU) which maximize the model's predictive accuracy in terms of how much correctness word our models predict.

The training validation loss and accuracy results are presented in Figure 6.

6.1 Next Word Prediction

As discussed in the previous section, the final layer in the deep learning model has one neuron for each word in the output vocabulary and a *softmax* activation function is used to output a likelihood of each word in the vocabulary being the next word in the sequence. In practice, heuristic search methods are used to return one or more decoded output sequences for a given prediction. Here, we adopted a Greedy Search decoding algorithm which takes the highest scoring item at each stage using the *argmax* mathematical function provided in *numpy*. That is then used as the next word and is fed as input to the next step. These steps are repeated with every item of vocabulary until one reaches some maximum length.

For the next step, namely to take input from the researchers, we opened an infinite loop where the execution was terminated once we entered a zero. If any word was typed, it would split that word, take the last three words and predict the next word (Table 3).

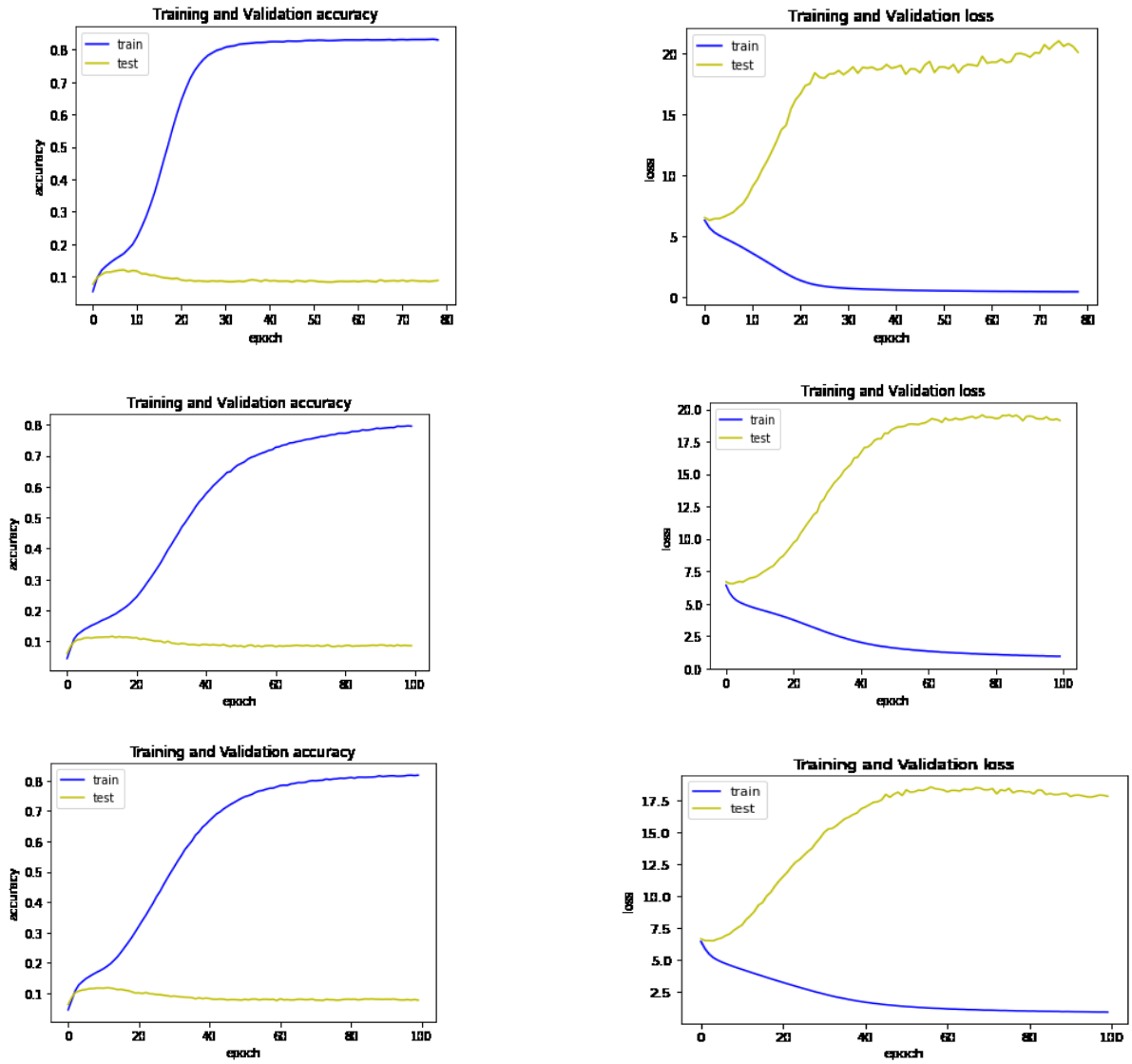

Figure 6. Accuracy and Loss for the Proposed Models

Table 3. Prediction Result

User Input	Sequences	Correct Next Word	Stacked LSTM	BiLSTM/LSTM	Stacked GRU
The Project Gutenberg	['The', 'Project', 'Gutenberg']	ebook	ebook	tm	licence
This eBook is	['This', 'eBook', 'is']	for	for	for	for
circulation within five	['circulation', 'within', 'five']	minutes	minutes	minutes	minutes
Her performance was	['Her', 'performance', 'was']	pleasing	pleasing	pleasing	pleasing
she was only	['she', 'was', 'only']	fifteen	fifteen	exceedingly	eight
to acknowledge her	['to', 'acknowledge', 'her']	figure	figure	figure	figure
Elizabeth most thankfully	['elizabeth', 'most', 'thankfully']	consented	consented	consented	consented
she is very	['she', 'is', 'very']	ill	ill	ill	welcome
Objection to my	['objection', 'to', 'my']	marrying	marrying	marrying	marrying
Exclusion of all	['exclusion', 'of', 'all']	conversation	conversation	conversation	conversation

From Table 3, it can be deduced that out of 10 sequences entered, the two layer LSTM predicted all the next words correctly. Bi-LSTM stacking on unidirectional LSTM predicted 8 correctly while the two layer GRU predicted 7 correctly. From these results, it is suggested that a two layer LSTM model is the most accurate in predicting the next word.

7. Business Benefits

The paper presented a model for next word prediction, and clearly identified the advantages and disadvantages of current RNN-based architectures. It can also be used for discovering theme topics in short texts with coherence.

8. Conclusion

A next word prediction task is an application of a language model in natural language generation that deals with generating words by repeatedly sampling the next word conditioned on the previous choices. A deep stacked LSTM, GRU and Bi-LSTM stacking on unidirectional LSTM are proposed in this paper for next word prediction. We have shown that stacked LSTM is the best structure for next word predictions. However, the models cannot generalize well on unseen data. We intend to apply text augmentation to further handle the overfitting of the models.

Further improvements and extensions, such as using the Transformer model for more dynamic text generation in real-time, as well as adopting more decoding strategies such as Temperature Sampling, Top-K Sampling, Top-P Sampling and Beam Search Algorithm, can be adopted for optimal next word predictions. Finally, we intend to use these language models for low-resource languages to predict next word generation.

References

[1] M.Tamizharasan, R. Shahana, and P. Subathra. 2020. "Topic Modeling-Based Approach for Word Prediction Using Automata." *Journal of Critical Reviews* 7(7): 744–749.

[2] Y. HaCohen-Kerner, A. Applebaum, and J. Bitterman. 2017. "Improved Language Models for Word Prediction and Completion with Application to Hebrew." *Applied Artificial Intelligence* 31(3): 232–250.

[3] Z-H. Zhou. 2003. "Three Perspectives of Data Mining." *Artificial Intelligence* 143(1): 139–146.

[4] G. G. Chowdhury. 2005. "Natural Language Processing." *Annual review of information science and technology* 37(1): 51–89.

[5] O. Iosifova, I. Iosifov, and O. Rolik. 2020. "Techniques and Components for Natural Language Processing." *Адаптивні системи автоматичного управління* 1(36): 93–113.

[6] A. Stolcke, N. Coccaro, R. Bates, P. Taylor, C. V. Ess-Dykema, K. Ries, E. Shriberg, D. Jurafsky, R. Martin and M. Meteer. 2000. "Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech." *Computational linguistics* 26(3): 339–373.

[7] F. Chaubard, R. Mundra and R. Socher. 2016. "CS 224D: Deep Learning for NLP1."

[8] L. A. Tuan, D. J. Shah and R. Barzilay. 2020. "Capturing Greater Context for Question Generation." In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[9] J. Devlin, M-W. Chang, K. Lee and K. Toutanova. 2019. "Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding." *arXiv preprint arXiv:1810.04805*.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin. 2017. "Attention Is All You Need." In *Advances in Neural Information Processing Systems*, , 5998–6008.

[11] A. Elmogy, B. Mahmoud and M. Saleh. 2019. "A Deep Learning Approach for Text Generation." In *2019 29th International Conference on Computer Theory and Applications (ICCTA)*, IEEE, 102–106.

[12] A. Graves, A. Mohamed and G. Hinton. 2013. "Speech Recognition with Deep Recurrent Neural Networks." *IEEE International Conference on Acoustics, Speech and Signal Processing*. *arXiv:1303.5775v1*.

[13] M. Wang, Z. Lu, H. Li, W. Jiang and Q. Liu. 2015. "genCNN: A Convolutional Architecture for Word Sequence Prediction." *arXiv:1503.05034.*

[14] P. P. Barman and A. Boruah. 2018. "A RNN Based Approach for next Word Prediction in Assamese Phonetic Transcription." *Procedia computer science* Vol. 143: 117–123.

[15] K. Shakhovska, I. Dumyn, N. Kryvinska and M. K. Kagita. 2021. "An Approach for a Next-Word

Prediction for Ukrainian Language." *Wireless Communications and Mobile Computing*.
[16] T. Mikolov, K. Chen, G. Corrado and J. Dean. 2013. "Efficient Estimation of Word Representations in Vector Space." In *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*
[17] S. Ambulgekar, S. Malewadikar, R. Garande and B. Joshi. 2021. "Next Words Prediction Using Recurrent NeuralNetworks." In *ITM Web of Conferences*, EDP Sciences.
[18] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, 2016. "TensorFlow: A System for Large-Scale Machine Learning." Retrieved from tensorflow.org
[19] F. Chollet. 2021. "Keras." Retrieved from https://keras.io/
[20] T-H. Wen 2018. "Recurrent Neural Network Language Generation for Dialogue Systems." (Doctoral thesis). https://doi.org/10.17863/CAM.22900
[21] Z. Cui, R. Ke, Z. Pu and Y. Wang. 2018. "Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-Wide Traffic Speed Prediction." *arXiv preprint arXiv:1801.02143*.
[22] M. Battula. 2019. "Time Series Forecasting with Deep Stacked Unidirectional and Bidirectional LSTMs."
[23] L. Xu and J. Hu. 2021. "A Method of Defect Depth Recognition in Active Infrared Thermography Based on GRU Networks." *Applied Sciences* 11(14): 6387, https://doi.org/10.3390/app11146387
[24] A. See. 2018. "Natural Language processing with Deep Learning." Retrieved from https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture15-nlg.pdf
[25] Y. Keneshloo, T. Shi, N. Ramakrishnan and C. K. Reddy. 2019. "Deep Reinforcement Learning for Sequence-to-Sequence Models." https://arxiv.org/pdf/1805.09461.pdf
[26] D. P. Kingma and J. Ba. 2014. "Adam: A Method for Stochastic Optimization." *arXiv prepr. arXiv1412.6980*
[27] J. Solawetz. 2020. "The Train, Validation, Test Split and Why you Need it" Available: https://blog.roboflow.com/train-test-split/