Text Generation using Long Short-Term Memory to Generate a LinkedIn Post

Muhammad Rizqi Assabil [a,1], Novi Yusliani [b,2*], Annisa Darmawahyuni [b,3]

[a] Undergraduate Student of Informatics Engineering, Universitas Sriwijaya, Indonesia
[b] Computer Science Faculty, Universitas Sriwijaya, Indonesia
[1] 09021281924065@students.unsri.ac.id; [2*] novi_yusliani@unsri.ac.id; [3] riset.annisadarmawahyuni@gmail.com
* corresponding author

ARTICLE INFO

Article history
Received 2023-02-06
Revised 2023-05-31
Accepted 2023-08-23

Keywords
Natural Language Generation
Artificial Neural Network
Long Short-Term Memory

ABSTRACT

LinkedIn is one of the most popular sites out there to advertise oneself to potential employer. This study aims to create a good enough text generation model that it can generate a text as if it were made by someone who posts on LinkedIn. This study will use a Neural Network layer called Long Short-Term Memory (LSTM) as the main algorithm and the train data consists of actual posts made by users in LinkedIn. LSTM is an algorithm that is created to reduce vanishing and exploding gradient problem in Neural Network. From the result, final accuracy and loss varies. Increasing learning rate from its default value of 0.001, to 0.01, or even 0.1 creates worse model. Meanwhile, increasing dimensions of LSTM will sometimes increases training time or decreases it while not really increasing model performance. In the end, models chosen at the end are models with around 97% of accuracy. From this study, it can be concluded that it is possible to use LSTM to create a text generation model. However, the result might not be too satisfying. For future work, it is advised to instead use a newer model, such as the Transformer model.

1. Introduction

Text generation, also known as Natural Language Generation (NLG), is subfield of artificial intelligence and Natural Language Processing (NLP) that aims to create a computer system that is capable of constructing texts that are human readable [1]. It is not only used to just create a sentence, but it can also be used to make a weather report [2], patient report [3], and even to automate image captioning [4]. NLG has been around since 1960 with the creation of ELIZA, a system that simulates conversation between human, created by an MIT professor, Joseph Weizenbaum [5]. There are several methods to create an NLG system, such as Markov Chain [6], Recurrent Neural Network (RNN), and Long Short Term Memory (LSTM).

LSTM is a variation of RNN. LSTM is created to try and solve problems that exist in vanilla RNN, exploding and vanishing gradient problem [7,8,9]. These two problems are solved by performing extra calculations in the neural network that ensures values on each neuron won't go too high or too low. These calculations work by putting inputs through three gates, called the input gate, the forget gate, and the output gate [10].

NLG using LSTM has been done before but with different ways. In [11], the author did not only use sequence as a training data, but he also highlighted important words in each sequence. They called these highlighted words as context and every context are chosen with either word importance or word clustering techniques. In [12], they create an NLG that will try to combine two paragraphs into one. This is done by having that input paragraphs as the training data. This means that training has to be done every time an input is received.

LinkedIn is one of the most popular website today to advertise oneself. It's a professional social media platform. The website allows its users to create a post describing their daily life, but in a more professional setting, and also to show off their achievements. This will serve as a kind of portfolio for potential recruiters to review. Naturally, this encourages people to create exaggerated posts in an effort to look better than others in the platform [13]. This behavior obviously doesn't suit everybody. On this study, we attempt to create an NLG system using LSTM to create texts similar to a real LinkedIn post.

2. Literature Study

A. Long Short Term Memory (LSTM)

In vanilla RNN, there are problems called the exploding and the vanishing gradient problem. These problems happen when RNN has to unroll way too many times. When RNN trains, it has to do multiplication between two values together again and again. How many times it performs multiplication depends on the amount of input. The result is a value that either will be way too high (exploding) or too low (vanishing), making it impossible to do backpropagation [7,8,9]. Effectively, vanilla RNN is unusable for real world scenario. LSTM solves this by doing extra steps during the multiplication process. These extra steps are called "gates" and in LSTM there are three gates. They are called input gate, forget gate, and output gate [10]. Figure 1 shows how an LSTM unit looks like.

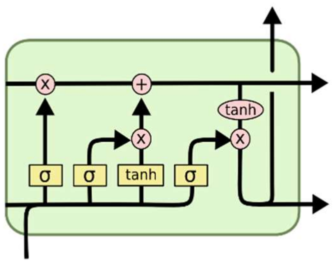


Fig. 1. A Singular LSTM Unit [14]

$$f_t = \sigma_g (W_f x_t + U_f h_{t-1} + b_f) \quad (1)$$

$$i_t = \sigma_g (W_i x_t + U_i h_{t-1} + b_i) \quad (2)$$

$$o_t = \sigma_g (W_o x_t + U_o h_{t-1} + b_o) \quad (3)$$

$$c_{tx} = \sigma_h (W_c x_t + U_c h_{t-1} + b_c) \quad (4)$$

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

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

Equation (1) is calculation for the forget gate. It will calculate weight times the input plus some bias like how neural network usually works. It will then go through a sigmoid activation function. Equation (2) and (4) are the calculation for the input gate. Equation (3) is the calculation for output gate. Equation (2), (4), and (3) are similar to (1), except (4) goes through a tanh activation function (noted by the σ_h symbol). And finally, (5) and (6) covers the calculation for cell state and hidden state. These two states will be sent to the next LSTM unit, but a hidden state will also act as an overall output for that LSTM unit.

B. Text Data Preprocessing

Before data can be processed by an LSTM unit, every data has to go through several preprocessing steps. This will help reduce the size of the overall corpus and also to help LSTM to detect patterns in texts better. Preprocessing steps vary for each NLP case, but for this study it will be divided into cleaning the data, removing stopwords, casefolding, tokenization, and then turning each data into a padded sequence.

1. Data cleaning involves the process of removing parts in the sentence that are considered as a disturbance, such as removing extra whitespaces, removing emojis, and removing characters that are not Latin alphabets.

2. Stopwords removal is the process of removing words that are deemed unnecessary in the corpus. Unnecessary could mean that the words are redundant in a sentence or it appears too many times. There isn't an actual agreement of every words that are considered as stopwords, thus the list of unnecessary words vary for each case [15].

3. Casefolding is the process of turning every character in the corpus into lowercase or uppercase. In this case, every character will be turned into lowercase [16].

4. Text tokenization is the process of separating each sentence into smaller chunks. Each sentence could be separated word by word, phrase by phrase, or into any other form of fragmentation [17]. These separated chunks are called tokens. These tokens will then also be used as a vocabulary.

5. Every sequence will then be turned into an n-gram sequence and then into a padded sequence. N-gram is a sequence with *n* amount of words [18], while padded sequence is the process of turning every sequence into a same sized sequence [19]. The final element of the padded sequence will then be one hot encoded. One hot encoding is the process of converting categorical data into numerical data [20].

3. Methodology

Since this study aims to create an NLG system to generate text similar to a LinkedIn posts, the training data would have to be real posts from that site. Data for this study is taken from Kaggle[1]. The data consists of around 34000 real posts made by LinkedIn users. For this study, there will be two NLG system created, one to generate text word per word, and one to generate text character per character. The reason this is done is that when each sentence is tokenized word by word, the vocabulary size is gigantic (50541 words). This will cause problems during training process where running out of memory will be common occurrence. For comparison, a character by character tokenization will result in a vocabulary in the size of only 57. Data will then be separated into 80% for train data and 20% for validation data.

After preprocessing steps has been done, the data will go through a neural network with several layers. The architecture can be seen in Figure 2. Embedding layer will turn each input into an n-dimensional vector representation. The idea is that words with similar meaning will have vectors that are close to each other. Data will then go into two bidirectional LSTM layer. Bidirectional LSTM will train the neural network with data going forward and backward. It will then go through a dense layer where each neuron will represent a token in the vocabulary. All of these will be done using the TensorFlow library.

The hyperparameters that will be tuned are the learning rate of the neural network optimizer (in this case, Adam), dimension size of the embedding and LSTM layers, and batch size. Table 1 represents every test case that is tried. W in Test Code represents that it's a model that generate text word by word. C represents a model to generate text character by character. To measure how well the model performs, metrics that will be considered are accuracy, loss, precision, recall, f1-score, and the model training duration. To make sure the training process doesn't go too long, training will be stopped if 97% accuracy is reached. The total epoch required to reach this accuracy will also be considered as a metric. Additionally, each model will also be tested by giving the same text as an input. A questionnaire will then be made where respondents could judge how good the generated texts are.

[1] https://www.kaggle.com/datasets/shreyasajal/linkedin-influencers-data

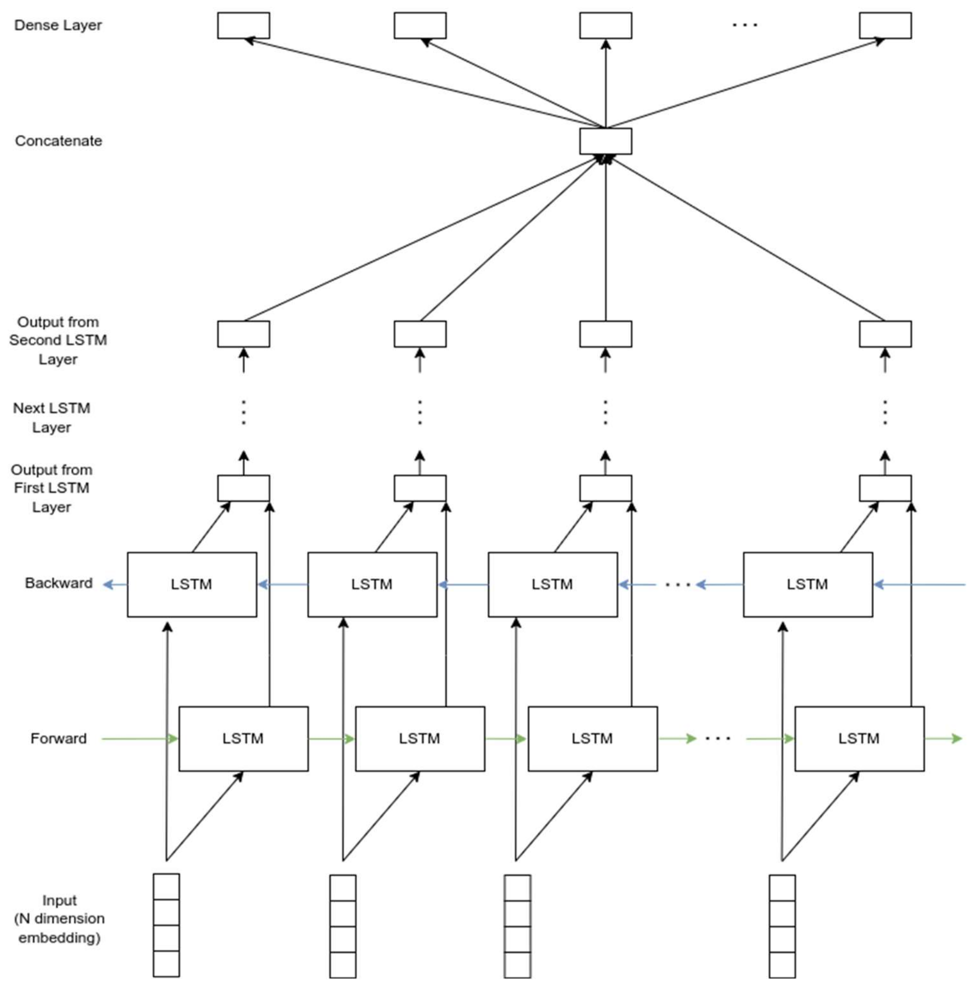

Dense Layer
Concatenate
Output from Second LSTM Layer
Next LSTM Layer
Output from First LSTM Layer
Backward
LSTM
Forward
Input (N dimension embedding)

Fig. 2. Full Architecture

Table 1. Hyperparameters for Each Test Cases

Test Code	Learning Rate	Dimension Size			Batch size
		Embedding	*LSTM*	*Second LSTM*	
C-1	0.001	50	100	100	16
C-2	0.001	100	250	250	16
C-3	0.001	200	450	450	16
C-4	0.01	50	100	100	16
C-5	0.01	100	250	250	16
C-6	0.01	200	450	450	16
C-7	0.1	50	100	100	16
C-8	0.1	100	250	250	16
C-9	0.1	200	450	450	16
C-10	0.001	50	100	100	32
C-11	0.001	100	250	250	32
C-12	0.001	200	450	450	32
C-13	0.01	50	100	100	32
C-14	0.01	100	250	250	32
C-15	0.01	200	450	450	32
C-16	0.1	50	100	100	32
C-17	0.1	100	250	250	32
C-18	0.1	200	450	450	32
W-1	0.001	50	100	100	16
W-2	0.001	100	250	250	16
W-3	0.001	200	450	450	16

W-4	0.01	50	100	100	16
W-5	0.01	100	250	250	16
W-6	0.01	200	450	450	16
W-7	0.1	50	100	100	16
W-8	0.1	100	250	250	16
W-9	0.1	200	450	450	16
W-10	0.001	50	100	100	32
W-11	0.001	100	250	250	32
W-12	0.001	200	450	450	32
W-13	0.01	50	100	100	32
W-14	0.01	100	250	250	32
W-15	0.01	200	450	450	32
W-16	0.1	50	100	100	32
W-17	0.1	100	250	250	32
W-18	0.1	200	450	450	32

4. Result and Discussion

Every time a training is done, accuracy, loss, epoch, and training duration are acquired. The result from each test case are shown in Table 2. The maximum epochs for each training is 150, meaning that every model that reach 150 epochs never reached the desired accuracy.

Table 2. Accuracy, Loss, Epoch, and Train Duration

Test Code	Accuracy	Validation Accuracy	Loss	Validation Loss	Epoch	Train Duration
C-1	0.9718	0.9322	0.1770	0.2694	37	2.78 minute
C-2	0.9792	0.9706	0.1374	0.1434	23	2.03 minute
C-3	0.9710	0.9588	0.1440	0.1617	17	1.83 minute
C-4	0.8090	0.8494	0.5773	0.4781	150	10.18 minute
C-5	0.5857	0.5853	1.4157	1.5580	150	10.95 minute
C-6	0.4012	0.3208	2.3160	2.9759	150	16.03 minute
C-7	0.2551	0.2669	2.5746	2.5237	150	9.78 minute
C-8	0.1714	0.1310	8.3463	10.4271	150	10.4 minute
C-9	0.1486	0.1661	17.7397	14.3418	150	15.73 minute
C-10	0.9759	0.9282	0.2338	0.3324	40	1.46 minute
C-11	0.9731	0.9722	0.1604	0.1445	27	1.18 minute
C-12	0.9702	0.9706	0.1504	0.1385	20	1.55 minute
C-13	0.9714	0.9776	0.1340	0.1068	56	1.78 minute
C-14	0.8988	0.9094	0.3395	0.3154	150	5.61 minute
C-15	0.5269	0.5049	1.6566	1.8359	150	10.5 minute
C-16	0.2800	0.2824	2.6872	2.6114	150	4.56 minute
C-17	0.1616	0.1653	3.9676	3.7478	150	5.53 minute
C-18	0.1465	0.1384	11.5181	9.4555	150	10.15 minute
W-1	0.9704	0.8841	0.2715	0.4628	143	11.31 minute
W-2	0.9719	0.9664	0.1153	0.1325	76	7.46 minute
W-3	0.9729	0.9857	0.2107	0.1669	58	8.71 minute
W-4	0.9709	0.9809	0.1066	0.0778	134	10.48 minute
W-5	0.9757	0.9857	0.1506	0.1228	102	9.56 minute
W-6	0.9743	0.9857	0.1629	0.1275	69	6.5 minute
W-7	0.9714	0.9871	0.1512	0.0968	47	2.01 minute
W-8	0.9757	0.9871	0.1050	0.0596	57	3.43 minute
W-9	0.9700	0.9857	0.1674	0.0916	63	6.16 minute
W-10	0.6743	0.7171	1.6548	1.5828	150	3.86 minute
W-11	0.9700	0.9814	0.2303	0.2045	114	3.66 minute
W-12	0.9714	0.9900	0.1555	0.1202	109	5.73 minute
W-13	0.9714	0.9871	0.1204	0.0933	48	1.25 minute
W-14	0.9729	0.9886	0.1732	0.1288	78	2.55 minute
W-15	0.9729	0.9814	0.1330	0.0739	59	3.26 minute
W-16	0.3886	0.4886	3.3625	2.6510	150	3.83 minute
W-17	0.0843	0.1414	24.8388	20.7708	150	4.96 minute
W-18	0.4571	0.4600	13.7626	12.7963	150	8.05 minute

When increasing learning rate, model will become more unstable. Accuracy and loss values would stay really high and would make train duration longer. Increasing batch size would cut down training time significantly without showing any big changes in other metrics. Increasing dimension size would also speed up training.

Below in Table 3 are the result of giving the same text to each model. To make sure there are significant amount of texts generated, C models will generate 50 characters, while W models will generate 15 words.

Table 3. Prediction Result for Each Case

Test Code	Input text	Prediction times	Output text
C-1	Hello guys, please congratulate me on	50	Hello guys, please congratulate me on the ustilink and the wek. nend. ., aunk on the s2
C-2	Hello guys, please congratulate me on	50	Hello guys, please congratulate me on the distinguished panel. my answer to the ppor to
C-3	Hello guys, please congratulate me on	50	Hello guys, please congratulate me on frister for onaline engaging and parent in tha uk
C-4	Hello guys, please congratulate me on	50	Hello guys, please congratulate me on that is is tanaunt. ppsenf ok auft.auants on auth
C-5	Hello guys, please congratulate me on	50	Hello guys, please congratulate me only and and pkonvpy tpapaev pany anaa and a a kaata
C-6	Hello guys, please congratulate me on	50	Hello guys, please congratulate me only anlellabrakaaoudeaeaneaf efotetia utandfenkagt
C-7	Hello guys, please congratulate me on	50	Hello guys, please congratulate me one oan an anenooonoater uk v iooto au nnandok oan k
C-8	Hello guys, please congratulate me on	50	Hello guys, please congratulate me onas.as.as.a.a. aoga a spae aae aaa.a.arooaa.a .aaaa
C-9	Hello guys, please congratulate me on	50	Hello guys, please congratulate me oner oooooooook a ok oeakkay akkko0araroavokk okk0k
C-10	Hello guys, please congratulate me on	50	Hello guys, please congratulate me on stor an entertary it is as on frem. boons ppromak
C-11	Hello guys, please congratulate me on	50	Hello guys, please congratulate me on a prest of healing trauma. and only 6 slots are a
C-12	Hello guys, please congratulate me on	50	Hello guys, please congratulate me on the us. link 27th a ther schoor can bus bus a pra
C-13	Hello guys, please congratulate me on	50	Hello guys, please congratulate me on an and the tradab an be an an anunot too fro th
C-14	Hello guys, please congratulate me on	50	Hello guys, please congratulate me on camas at 5:00 a panigister enuaunuauitauionank ry
C-15	Hello guys, please congratulate me on	50	Hello guys, please congratulate me ong witionow. t anok toeratione anatun utind y footu
C-16	Hello guys, please congratulate me on	50	Hello guys, please congratulate me on on on on onnf/ ann eokr kaefaeaaran'vaarkykokofk
C-17	Hello guys, please congratulate me on	50	Hello guys, please congratulate me oneaneaneaneaneyeany. Noyyenyyyyyyyeneftenyeaneyyyea
C-18	Hello guys, please congratulate me on	50	Hello guys, please congratulate me on oooooooooooneooon a a a okreearore0e0ro okafeao a
W-1	Hello guys, please congratulate me on	15	Hello guys, please congratulate me on t h r y m o n z o r s e m i n
W-2	Hello guys, please congratulate me on	15	Hello guys, please congratulate me on i d e : : : : : : : : : : : :
W-3	Hello guys, please congratulate me on	15	Hello guys, please congratulate me on a central part of holiday giving and the need and work but forward and spotify
W-4	Hello guys, please congratulate me on	15	Hello guys, please congratulate me on e g i n g e l e a d e c a n d
W-5	Hello guys, please congratulate me on	15	Hello guys, please congratulate me on a children's book author who can read with your children and work with them be
W-6	Hello guys, please congratulate me on	15	Hello guys, please congratulate me on the impact of trauma on our students and them indeed, below; or have will be
W-7	Hello guys, please congratulate me on	15	Hello guys, please congratulate me on and as it is the way who have from my college that is and nationally.
W-8	Hello guys, please congratulate me on	15	Hello guys, please congratulate me on dec. mean post mental issues and trauma. doesn't stop at school door w/ caeop stop

W-9	Hello guys, please congratulate me on	15	Hello guys, please congratulate me on the to be below. the event. on christmas or soundcloud. soundcloud. soundcloud. soundcloud. soundcloud. soundcloud.
W-10	Hello guys, please congratulate me on	15	Hello guys, please congratulate me on webinar on our students on whether schools can be but and trauma. but it is
W-11	Hello guys, please congratulate me on	15	Hello guys, please congratulate me on the project. here's it is often subtle but constant on and educators. free of it
W-12	Hello guys, please congratulate me on	15	Hello guys, please congratulate me on a solution to the pandemic are only partially true we look and seeing pre-k have
W-13	Hello guys, please congratulate me on	15	Hello guys, please congratulate me on a mean post mental health issues and trauma. for trauma. on the podcast and educators.
W-14	Hello guys, please congratulate me on	15	Hello guys, please congratulate me on stop at rutgers door w/ rutgers grad school conference on soundcloud. soundcloud. soundcloud. soundcloud. soundcloud.
W-15	Hello guys, please congratulate me on	15	Hello guys, please congratulate me on the vaccine will help obviously but it is will not a time will will current
W-16	Hello guys, please congratulate me on	15	Hello guys, please congratulate me on giving and and and and educational the planning marybeth professor marybeth but annual stop event
W-17	Hello guys, please congratulate me on	15	Hello guys, please congratulate me on with new with sessions sessions with sessions event. will be for the for current gift
W-18	Hello guys, please congratulate me on	15	Hello guys, please congratulate me on a the the the the home. the the and for for the for what up.

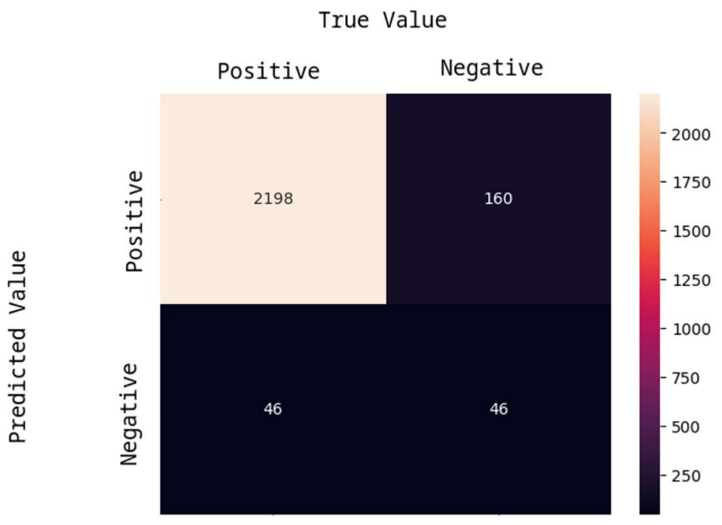
Table 4 shows results of questionnaire where each respondent judges the quality of the texts above. A total of 4 person has responded. Respondent judges by how similar (abbreviated in Table 4 as S) the texts are to texts made by humans and how good the grammar (abbreviated in Table 4 as G) in the texts are. Each metric is a scale from 1 to 5 where 1 represents worse and 5 represents best.

Table 4. Questionnaire Result

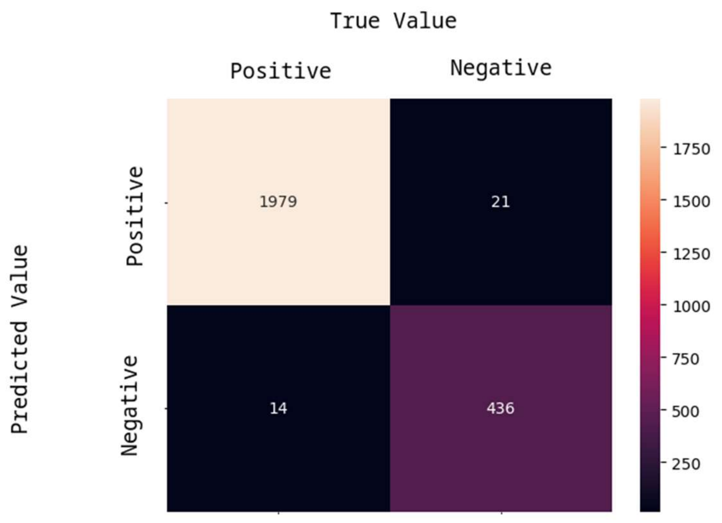
Code	**Answers**								**Total**	**Average**
	Respondent 1		*Respondent 2*		*Respondent 3*		*Respondent 4*			
	S	*G*	*S*	*G*	*S*	*G*	*S*	*G*		
C-1	3	3	2	2	3	3	2	2	20	5
C-2	4	3	3	4	3	3	3	2	25	6.25
C-3	4	4	4	4	3	3	3	2	27	6.75
C-4	3	3	1	1	3	3	1	1	16	4
C-5	3	3	2	1	3	3	1	1	17	4.25
C-6	3	4	2	1	3	3	1	1	18	4.5
C-7	3	4	2	1	3	3	1	1	18	4.5
C-8	3	3	1	1	3	3	1	1	16	4
C-9	3	3	1	1	3	3	2	2	18	4.5
C-10	4	3	2	3	3	3	2	2	22	5.5
C-11	4	4	3	3	3	3	4	2	26	6.5
C-12	4	4	3	3	3	3	4	2	26	6.5
C-13	3	3	2	2	3	3	2	2	20	5
C-14	4	3	2	2	3	3	2	2	21	5.25
C-15	3	3	2	2	3	3	2	2	20	5
C-16	3	3	1	1	3	3	2	2	18	4.5
C-17	3	3	1	1	3	3	2	2	18	4.5
C-18	3	3	1	1	3	3	2	2	18	4.5
W-1	3	3	3	3	3	3	2	2	22	5.5
W-2	3	3	2	3	3	3	2	4	23	5.75
W-3	4	3	3	3	3	3	5	3	27	6.75
W-4	4	5	3	3	3	3	2	3	26	6.5
W-5	5	5	3	4	3	3	5	3	31	7.75
W-6	5	5	3	3	3	3	4	3	29	7.25
W-7	5	5	3	3	3	3	5	3	30	7.5
W-8	4	4	4	4	3	3	3	2	27	6.75
W-9	3	3	2	3	3	3	4	4	25	6.25

W-10	4	4	3	3	3	3	5	3	28	7
W-11	4	4	5	4	3	3	5	3	31	7.75
W-12	4	4	3	3	3	3	5	3	28	7
W-13	4	4	3	3	3	3	5	3	28	7
W-14	4	4	2	3	3	3	4	3	26	6.5
W-15	4	4	4	3	3	3	4	3	28	7
W-16	4	4	3	3	3	3	4	3	27	6.75
W-17	4	4	3	3	3	3	4	3	27	6.75
W-18	5	5	2	2	3	3	3	3	26	6.5

Lastly, the precision, recall, and f1-score for each model will be calculated. True Positives, True Negatives, False Positives, and False Negatives for every model will need to be calculated beforehand. To save spaces and study time, only models considered the best will be evaluated. Considering every metric that has been gained previously, model C-3 and W-11 are chosen. Data for testing is 10% of corpus that has been shuffled.



(a)



(b)

Fig. 2 Confusion Matrix. (a) shows matrix for W-11 model, and (b) shows C-3 model. Top left of matrix shows True Positive value, top right shows False Positive, bottom left shows False Negative, and bottom right shows True Negative.

Table 5. Precision, Recall, and F1-Score

Model / Metric	Word Based (W-11)	Character Based (C-3)
Precision	0.22272074229985844	0.9409993670156542
Recall	0.2677798918970857	0.870515163500567
F1-Score	0.21480836546197055	0.8932478722083184

When looking at results from Table 5, it can be seen that word-based model performs terribly compared to character-based model. The reason this happen could be traced back to the amount of vocabulary in word-based model. Whereas character-based model only has 57 different classes to predict, word-based model has 50541 different classes to predict. There's a big chance class imbalance is happening in word-based model where certain class didn't have enough representation in the corpus. This problem could be solved by oversampling the minority classes, undersampling the majority classes, or by simply adding more data to the corpus.

5. Conclusion

An NLG model made using LSTM algorithm performs pretty poorly for word per word prediction. Overall, this study can still be improved in a lot of ways. The most obvious way is to do the training process with more data. Another way is to add or decrease neural network layers. Model could also be made with different algorithm. Most popular NLG system these days uses the Transformer model [21]. Lastly, a different neural network framework could be tried, namely PyTorch.

References

[1] E. Reiter and R. Dale, “Building applied natural language generation systems,” *Nat. Lang. Eng.*, vol. 3, no. 1, pp. 57–87, Mar. 1997, doi: 10.1017/S1351324997001502.

[2] E. Goldberg, N. Driedger, and R. I. Kittredge, “Using natural-language processing to produce weather forecasts,” *IEEE Expert*, vol. 9, no. 2, pp. 45–53, Apr. 1994, doi: 10.1109/64.294135.

[3] F. Portet *et al.*, “Automatic generation of textual summaries from neonatal intensive care data,” *Artif. Intell.*, vol. 173, no. 7–8, pp. 789–816, May 2009, doi: 10.1016/j.artint.2008.12.002.

[4] D. Hutchison *et al.*, “Every Picture Tells a Story: Generating Sentences from Images,” in *Computer Vision – ECCV 2010*, vol. 6314, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 15–29. doi: 10.1007/978-3-642-15561-1_2.

[5] C. Ireland, “Alan Turing at 100,” *Harvard Gazette*, Sep. 13, 2022. https://news.harvard.edu/gazette/story/2012/09/alan-turing-at-100/ (accessed Oct. 04, 2022).

[6] P. A. Gagniuc, *Markov Chains: From Theory to Implementation and Experimentation*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2017. doi: 10.1002/9781119387596.

[7] B. Hanin, “Which Neural Net Architectures Give Rise To Exploding and Vanishing Gradients?,” 2018, doi: 10.48550/ARXIV.1801.03744.

[8] A. Graves, “Long Short-Term Memory,” in *Supervised Sequence Labelling with Recurrent Neural Networks*, vol. 385, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 37–45. doi: 10.1007/978-3-642-24797-2_4.

[9] P. J. Werbos, “Backpropagation through time: what it does and how to do it,” *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990, doi: 10.1109/5.58337.

[10] G. Van Houdt, C. Mosquera, and G. Nápoles, “A review on the long short-term memory model,” *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5929–5955, Dec. 2020, doi: 10.1007/s10462-020-09838-1.

[11] S. Santhanam, “Context based Text-generation using LSTM networks,” 2020, doi: 10.48550/ARXIV.2005.00048.

[12] D. Pawade, A. Sakhapara, M. Jain, N. Jain, and K. Gada, “Story Scrambler - Automatic Text Generation Using Word Level RNN-LSTM,” *Int. J. Inf. Technol. Comput. Sci.*, vol. 10, no. 6, pp. 44–53, Jun. 2018, doi: 10.5815/ijitcs.2018.06.05.

[13] J. Guillory and J. T. Hancock, “The Effect of Linkedin on Deception in Resumes,” *Cyberpsychology Behav. Soc. Netw.*, vol. 15, no. 3, pp. 135–140, Mar. 2012, doi: 10.1089/cyber.2011.0389.

[14] M. Peixeiro, “Introduction to LSTM Units While Playing Jazz,” *Introduction to LSTM Units While Playing Jazz*, Apr. 17, 2019. https://towardsdatascience.com/introduction-to-lstm-units-while-playing-jazz-fa0175b59012 (accessed Jan. 24, 2023).

[15] D. Rosenberg, “Stop, Words,” *Representations*, vol. 127, no. 1, pp. 83–92, Aug. 2014, doi: 10.1525/rep.2014.127.1.83.

[16] Y. A. Gerhana, A. R. Atmadja, W. B. Zulfikar, and N. Ashanti, “The implementation of K-nearest neighbor algorithm in case-based reasoning model for forming automatic answer identity and searching answer similarity of algorithm case,” in *2017 5th International Conference on Cyber and IT Service Management (CITSM)*, Denpasar, Bali, Indonesia, Aug. 2017, pp. 1–5. doi: 10.1109/CITSM.2017.8089233.

[17] G. Grefenstette, “Tokenization,” in *Syntactic Wordclass Tagging*, vol. 9, H. van Halteren, Ed. Dordrecht: Springer Netherlands, 1999, pp. 117–133. doi: 10.1007/978-94-015-9273-4_9.

[18] Zhong Su, Qiang Yang, Ye Lu, and Hongjiang Zhang, “WhatNext: a prediction system for Web requests using n-gram sequence models,” in *Proceedings of the First International Conference on Web Information Systems Engineering*, Hong Kong, China, 2000, vol. 1, pp. 214–221. doi: 10.1109/WISE.2000.882395.

[19] M. Dwarampudi and N. V. S. Reddy, “Effects of padding on LSTMs and CNNs,” 2019, doi: 10.48550/ARXIV.1903.07288.

[20] A. C. H. Choong and N. K. Lee, “Evaluation of convolutionary neural networks modeling of DNA sequences using ordinal versus one-hot encoding method,” in *2017 International Conference on Computer and Drone Applications (IConDA)*, Kuching, Nov. 2017, pp. 60–65. doi: 10.1109/ICONDA.2017.8270400.

[21] A. Vaswani *et al.*, “Attention Is All You Need,” 2017, doi: 10.48550/ARXIV.1706.03762.