Copy-cat Bot for Narendra Modi which generates plausible new speeches in Modhi's style using machine learning approaches

Roshani Abeysekera[1] and DDA Gamini [2*]

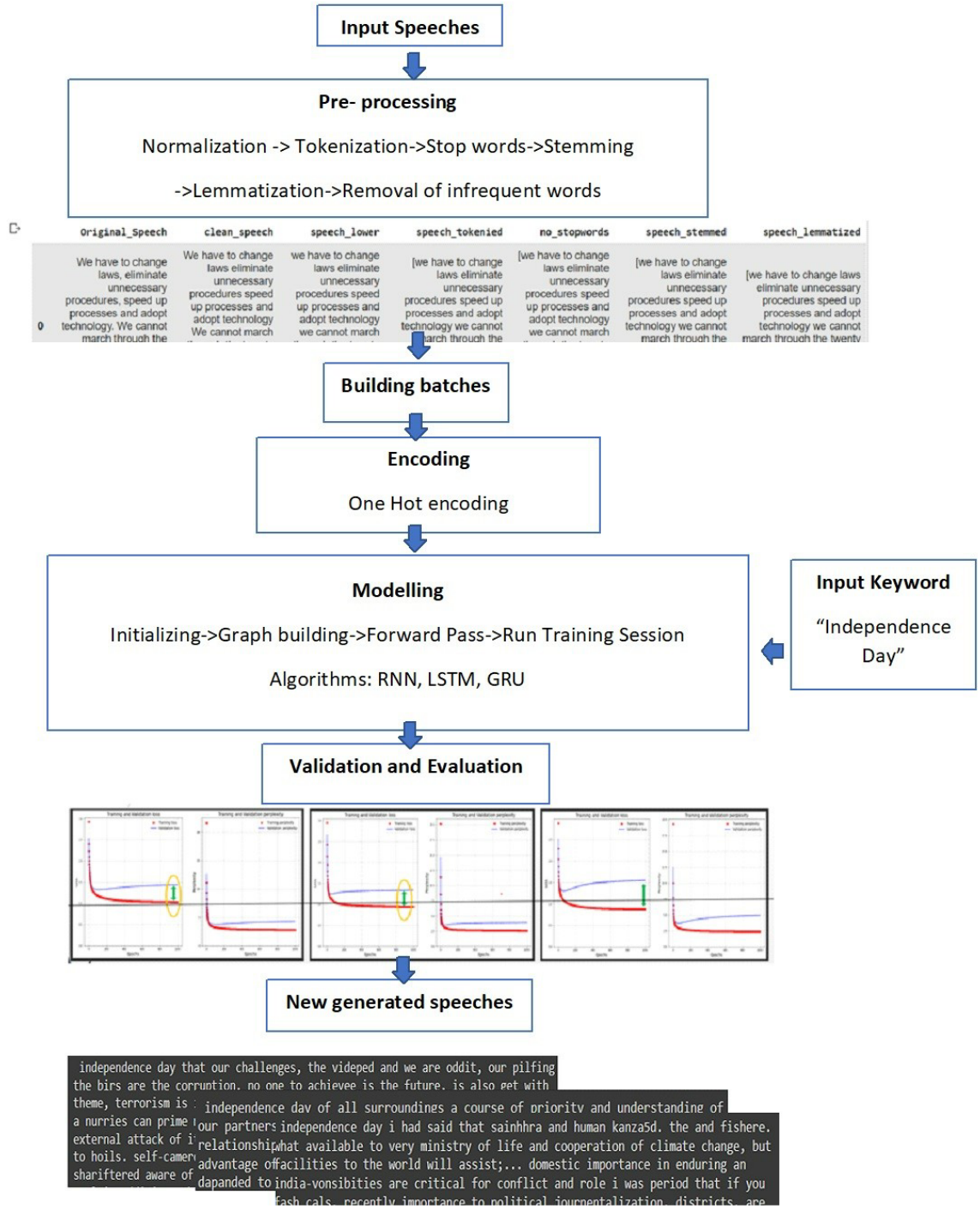
Department of Computer Science, Faculty of Applied Sciences, University of Sri Jayewardenepura, Nugegoda, Sri Lanka.

Date Received: 03-08-2022 Date Accepted: 20-10-2022

Abstract

Many consequences in the human past can be traced back to that one well-written, well-presented speech. Speeches grasp the power to move nations or touch hearts as long as they are well-thought-out. This is why gaining the expertise of speech giving and speech writing is something we should all intent to gain. A copy-cat bot is a model that can learn the writing and talking style of a certain person and replicate it. The main objective of this research study is to apply simple Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM) Recurrent Neural Networks and Gated Recurrent Unit (GRU) in developing a speech generation system that deep learns one text and then generates new text. This research looks into the generation of English transcripts of Narendra Modi's speeches. The generated text using LSTM and GRU models has great potential. The output resulted by RNN is less realistic and pragmatic, but its variants LSTM and GRU performed better. Though the grammatical correctness and the sentence transitions were absent in generated text of LSTM and GRU, but their output is somewhat logical as compared to RNN. LSTM and GRU performed better as it generated more realistic text and training loss is small, perplexity is small and mean probability is high compared to RNN.

Copy-cat Bot for Narendra Modi which generates plausible new speeches in Modhi's style using machine learning approaches



*Correspondence: gamini@sjp.ac.lk

Keywords: text generation, simple Recurrent Neural Network, Long Short-Term Memory Recurrent Neural Networks, Gated Recurrent, speeches, political speech generation.

1. Introduction

Political speeches are the pivotal event that connects the different parts of society as a united entity. The main intention of political speech is to get the corresponding end result through persuasion. Making speeches is a vital section of the politician's role in declaring policy and persuading people to agree with it.

Put in writing for the spoken word is a unique discipline. Congressional speechwriters' products should be written primarily although not exclusively to be heard not read. Political speeches are preferable formed in simple, direct, and often short sentences that can be easily understood by listeners. Speechwriters should use rhetorical devices such as repetition, variation, cadence and balance. It is important for speechwriters to study audiences and the occasion for and purpose of the speech. Most effective speeches do not exceed 20 minutes in length.

All literature is made up of words but the things that are said in plays expression of the ability to express thoughts and feelings differ one's speech from the other. Copying the speech style and habits of well-known political figures or other celebrities is a complicated task. Imitation provides the actor another window into creating a character. Imitation has a long and noble tradition in art. Characters are brought to life by their language and manner. However, creating one's speech that looks like his style is time-consuming and the speechwriter needs to observe the character well. If the speechwriter is familiar with the subject and the positions and style of the executive only small changes may be required. In other cases, the executive may notice that the speech does not have the right tone or flow and the entire speech may have to be re-drafted. In these situations, the copy-cat bot plays a major role in making the work effective. The aim of this research is to develop a copy-cat bot to generate plausible new speeches which look like some other speeches using the remarkable patterns in text format.

With powerful artificial Intelligence/machine learning libraries becoming readily available as open source, it seems obvious to apply them to speech writing.

1.1. Copy-cat Bot Models

Opinion summarization is the piece of work of automatically creating summaries that reflect subjective information transferred in multiple documents such as blogs, reviews, social media, or internet forums. Arthur Brazinskas et al. (Brazinskas, et al., 2020) introduced a simple end-to-end approach to unsupervised abstractive summarization of opinions which does not use any summaries in training. Human evaluation of the generated summaries (by considering their alignment with the reviews) shows that those created by the model better reflect the content of the input.

The circulation of fake news has affected many interruptions in society and weakened the news ecosystem. Hence, fake news should be carefully examined and combated. Thai Le et al. (Le, et al., 2020) formulated a novel problem of adversarial comment generation to fool fake news detectors. They introduced an end-to-end malicious comments generation framework that can generate realistic and

relevant adversarial comments to fool five of the most popular neural fake news detectors to predict fake news as real news with attack success rates of 94% and 90% for a white box and black box settings.

1.2. Political Speech Generation

Valentin Kassarnig (Kassarnig, 2016) presented a novel approach to training a system on speech transcripts to generate new speeches. They showed that n-grams and Justeson Katz POS tag filter (JK POS) (Kassarnig, 2016) are very effective as language and topic model for this task. They used 6-grams, a simple statistical language model which helps to determine very quickly all words which can occur after the previous five ones and how likely each of them is. Justeson and Katz (JK) POS tag filter for two and three-word terms used as the topic model. This paper presents a manual and an automated approach to evaluate the quality of generated speeches. In an experimental assessment generated speeches have shown very high quality in terms of grammatical correctness and sentence transitions.

Joseph Bullock et al. (Bullock & Luengo-Oroz, 2019) presented a proof-of-concept experiment to understand the complexity and illustrate the possibilities, of automatic text generation in the international political sphere. English language transcripts of speeches given by political leaders at the United Nations General Assembly (UNGA) between 1970 and 2015 inclusive, used as training data with little restriction on the content. They trained the Average-Stochastic Gradient Descent Weight-Dropped Long Short-Term Memory network (AWD-LSTM) language model that can generate text in the style of these speeches covering a variety of topics. Text is generated by 'seeding' the models with the starting point of a sentence or paragraph, then letting it predict the following text.

A sequence generation framework, called Sequence Generative Adversarial Nets (SeqGAN), to effectively train generative adversarial nets was proposed by Lantao Yu et al. (Yu, et al., 2017). For text generation scenarios, they applied the proposed SeqGAN to generate Chinese poems and Barack Obama political speeches. They used an oracle evaluation mechanism to clearly demonstrate the superiority of SeqGAN over strong baselines. A bilingual evaluation understudy (BLEU) score was used as an evaluation metric to measure the similarity degree between the generated texts and the human-created texts. For three real-world scenarios, i.e., poems, speech-language, and music generation, SeqGAN showed excellent performance on generating the creative sequences comparable to real human data.

1.3. Recurrent Neural Networks

Political speeches using Recurrent Neural Networks (RNNs) as language models were generated by Valentin Kassarnig (Kassarnig, 2016). The RNN takes as input a sequence of words and outputs the next word. Words were represented by one-hot-encoded feature vectors. The RNN had 50 hidden layers and used tanh (hyperbolic tangent function) as an activation function. For assessing the error, they used the cross-entropy loss function. Furthermore, they used Stochastic Gradient Descent (SGD) to minimize the loss and Backpropagation Through Time (BPTT) to calculate the gradients. After training the network for 100-time epochs (14 h) the results were still pretty poor. The majority of the generated sentences were grammatically incorrect.

*Correspondence: gamini@sjp.ac.lk

Ilya Sutskever et al. (Sutskever, et al., 2011) demonstrated the power of Recurrent Neural Networks (RNNs) trained with the new Hessian-Free optimizer (HF) by applying them to the task of predicting the next character in a stream of text. They introduced a new RNN variant that uses multiplicative connections which allow the current input character to determine the transition matrix from one hidden state vector to the next. After training the multiplicative RNN with the HF optimizer for five days on 8 high-end Graphics Processing Units, they were capable to surpass the achievement of the best previous single method for character-level language modeling.

Partha Pratim Barman et al. (Barman & Boruah, 2018) presented a Long Short Term Memory network (LSTM) model for instant messaging. Their model goes through the data set of the transcripted Assamese words and predicts the next word using LSTM with an accuracy of 88.20% for Assamese text and 72.10% for phonetically transcripted Assamese language.

A novel training algorithm that results in improved text generation compared to standard models, such as bilingual evaluation understudy (BLEU) or Recall-Oriented Understudy for Gisting Evaluation (ROUGE) was proposed by Marc'Aurelio Ranzato et al. (Ranzato, et al., 2016). Their work was motivated by two major deficiencies in training the current generative models for text generation: exposure bias and a loss that does not operate at the sequence level. They proposed the MIXER algorithm which enables successful training of reinforcement learning models for text generation. Their results showed that MIXER outperforms three strong baselines for greedy generation, and it is very competitive with beam search.

Alex Graves (Graves, 2014) showed how Long Short-term Memory recurrent neural networks can be used to generate both discrete and real-valued sequences with complex, long-range structures using next-step prediction. The method is demonstrated for text (where the data are discrete) and online handwriting (where the data are real-valued). It had also introduced a novel convolutional mechanism that allows a recurrent network to condition its predictions on an auxiliary annotation sequence and used this approach to synthesize diverse and realistic samples of online handwriting. Comparison with real and generated text show that these regions are a fairly accurate reflection of the constitution of the real data although the generated versions tend to be somewhat shorter and more jumbled together.

Recurrent neural networks (RNN) with three commonly used recurrent units; a traditional tanh (hyperbolic tangent function) unit, a long short-term memory (LSTM) unit, and a recently proposed gated recurrent unit (GRU) were empirically evaluated by Junyoung Chung et al. (Chung, et al., 2014). Their evaluation focused on the task of sequence modeling on several datasets including polyphonic music data and raw speech signal data. The evaluation clearly illustrated the excellence of the gated units; both the LSTM unit and GRU, over the traditional tanh (hyperbolic tangent function) unit. However, they could not make a concrete conclusion on which of the two gating units was superior.

Title	Author	Algorithms	Conclusion	Year
Generating Sequences with Recurrent Neural Networks (Graves, 2014)	Alex Graves	Long Short-term Memory recurrent neural networks	The resulting system is able to generate highly realistic cursive handwriting	2014

			in a wide variety of styles.	
Generating Text with Recurrent Neural Networks (Sutskever, et al., 2011)	Ilya Sutskever, James Martens, Geoffrey Hinton	RNNs trained with the new Hessian-Free optimizer (HF) by applying them to character-level language modeling tasks.	Able to surpass the performance of the best previous single method for character-level language modeling.	2011
Political Speech Generation (Kassarnig, 2016)	Valentin Kassarnig	n-grams and Justeson & Katz POS tag filter (J&K POS)	Generated speeches have shown very high quality in terms of grammatical correctness and sentence transitions.	2016
A RNN based Approach for next word prediction in Assamese Phonetic Transcription (Barman & Boruah, 2018)	Partha Pratim Barmana Abhijit Boruaha	Long Short-Term Memory network	Predicts the next word using LSTM with an accuracy of 88.20% for Assamese text and 72.10% for phonetically transcripted Assamese language	2018
Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. (Chung, et al., 2014)	Junyoung Chung Caglar Gulcehre KyungHyun Cho Yoshua Bengio	Long short-term memory Recurrent neural networks Gated recurrent unit	The evaluation clearly illustrated the excellence of the gated units; both the LSTM unit and GRU, over the traditional tanh (hyperbolic tangent function) unit.	2014
Unsupervised Opinion Summarization as Copycat-Review	Arthur Brazinskas Mirella Lapata	A simple end-to-end approach to unsupervised	Human evaluation of the generated summaries shows	2020

*Correspondence: gamini@sjp.ac.lk

Generation. (Brazinskas, et al., 2020)	Ivan Titov	abstractive summarization	that those created by the model better reflect the content of the input.	
MALCOM: Generating Malicious Comments to Attack Neural Fake News Detection Models. (Le, et al., 2020)	Thai Le Suhang Wang Dongwon Lee	Malcom, an end-to-end malicious comments generation framework	94% and 93.5% of the time on average Malcom can successfully mislead five of the latest neural detection models to always output targeted real and fake news labels.	2020
Automated Speech Generation from UN General Assembly Statements: Mapping Risks in AI Generated Texts. (Bullock & Luengo-Oroz, 2019)	Joseph Bullock, Miguel Luengo-Oroz	Average-Stochastic Gradient Descent Weight-Dropped Long Short-Term Memory network (AWD-LSTM) language model	Presented a proof-of-concept experiment to illustrate the ease with which a highly-accurate model that generates politically sensitive text can be created and highlighted the potential dangers of automated text generation.	2019
SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient (Yu, et al., 2017)	Lantao Yu, Weinan Zhang, Jun Wang, Yong Yu	SeqGAN used to generate Chinese poems and Barack Obama political speeches.	For three real-world scenarios, i.e., poems, speech-language, and music generation, SeqGAN showed excellent performance on generating the creative sequences comparable to real human data.	2017

In summary, these papers pointed out the wide range of different methodologies that had been employed successfully in the domain of text generation. Novelty of this research is the generation of speeches based on the given keyword using RNN, LSTM and GRU.

Training a computer to be able to think like a human being has been studied for decades. For any system to be able to learn, it must be programmed to learn to perform a task. Research work presented in this paper is focused on the generation of language models from the given keyword or headword by training different machine learning models.

1) *Recurrent Neural Network:* Recurrent neural networks are designed with associated data processing elements that are approximately designed to function like the human brain. They are constructed from layers of network nodes that have the potential to process input and forward output to other nodes in the network. In this way, it can acquire knowledge of the sequences of a problem and then generate absolutely new plausible sequences for the problem domain. RNN has a memory that takes care of information about early calculated results. Like feed-forward neural networks, RNNs can operate on data from initial input to final output. Unlike feedforward neural networks, RNNs use feedback loops, such as backpropagation throughout the time, through the computational process to loop information back into the network. This link inputs and is what enables RNNs to process sequential and temporal data. The most popular issues with RNNS are gradient vanishing and exploding problems. The gradients refer to the errors built as to the neural network trains. If the gradients start to explode, the neural network will come to unstable and unable to learn from training data (Laskowski, 2018).

2) *Long Short-Term Memory:* One drawback to standard RNNs is the vanishing gradient problem, in which the performance of the neural network suffers because it can't be trained properly and accurately. This occurs with deeply layered neural networks, which are applied to process complex data. Standard RNNs that use a gradient-based learning method demean as they grow bigger and more complex. Tuning the parameters productively at the earliest layers becomes too time-consuming, cumbersome and computationally expensive [15].

One solution to the issue is called long short-term memory (LSTM) networks, which computer scientists Sepp Hochreiter and Jurgen Schmidhuber invented in 1997. RNNs made with LSTM units categorize data into short-term and long-term memory cells. Doing so allow RNNs to figure out which data is dominant and should be remembered and looped back into the network. It also allows RNNs to figure out what data can be abandoned (Laskowski, 2018).

3) *Gated Recurrent Unit:* The Gated Recurrent Unit (GRU) is the younger sibling of the more ordinary Long Short-Term Memory (LSTM) network, and also a type of Recurrent Neural Network (RNN). Just like its sibling, GRUs are able to effectively keep long-term dependencies in sequential data. In addition, they can direct the "short-term memory" issue plaguing vanilla RNNs. The composition of the GRU permits it to adaptively take captive dependencies from large sequences of data without discarding information from earlier parts of the sequence. This is accomplished through its gating units, similar to the ones in LSTMs, which find a/the solution to the vanishing/exploding gradient problem of traditional RNNs.

*Correspondence: gamini@sjp.ac.lk

These gates are in control of synchronizing the information to be kept or discarded at each time step (Loye, 2022).

2. Methodology

Copy- cat bot for Narendra Modi which generates plausible new text which looks like some other text. In this research different machine learning algorithms are trained on preprocessed English transcript of Narendra Modi speeches. The proposed method is CRISP-DM methodology. First, the model data are fed. Models on the collected text are trained next. Then the model which automatically generates new speeches in the vein of Modhi's previous speeches is tested.

2.1. CRISP-DM Methodology

CRISP-DM stands for the cross-industry process for data mining that encourages best practices and allows projects to replicate. The CRISP-DM methodology provides an organized approach to planning a data mining project. This model represents the sequence of events. The tasks can be executed in a different order and it will often be required to backtrack to previous tasks and recurrent certain actions (Simmons, 2022). It is a model with six phases: business understanding, data understanding, data preparation, modeling, evaluation, deployment.

2.2. Data Understanding

The data source of this research is the English transcript of Narendra Modi which is available from a free resource. Dataset focus on English transcript of Modhi speeches, date place and title text. Data is complete and relevant. There are 467 speeches available in text format and manually labeled to access the ground truth. The reason behind taking the text format speeches are: -

- Number of features that can be stored is high.
- Amount of processing time to extract features is fast.
- Amount of processing time for analysis is fast.
- Required storage capacity is low.
- Availability of existing sources for data extraction is normal.

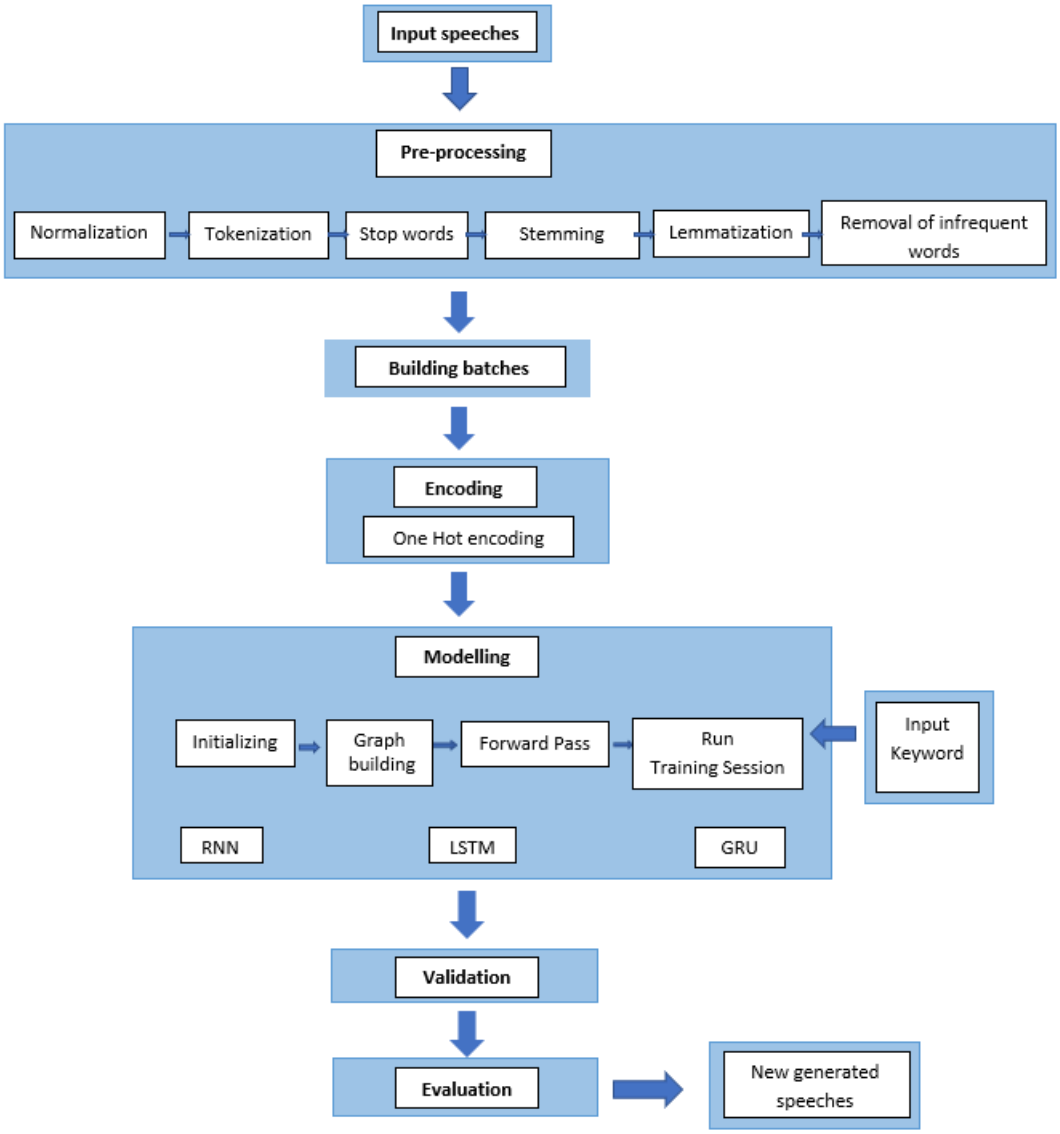


Figure1. Flow diagram of speech generation.

2.3. Data Preparation

Pre-processing the text is an action that turn the input document to a different output format. The main aim is to simplify, normalize the text so the ambiguities from the point of view of the model are removed. These kinds of operations are usually simple, rule-based operations, which are removing features. There is another type of pre-processing operation, which is adding features, so the texts are getting richer. From the implementation point of view, it is to insert or remove steps.

*Correspondence: gamini@sjp.ac.lk

	Original_Speech	clean_speech	speech_lower	speech_tokenied	no_stopwords	speech_stemmed	speech_lemmatized
0	We have to change laws, eliminate unnecessary procedures, speed up processes and adopt technology. We cannot march through the twenty first century with the administrative systems of the nineteenth century.	We have to change laws eliminate unnecessary procedures speed up processes and adopt technology We cannot march through the twenty first century with the administrative systems of the nineteenth century	we have to change laws eliminate unnecessary procedures speed up processes and adopt technology we cannot march through the twenty first century with the administrative systems of the nineteenth century	[we have to change laws eliminate unnecessary procedures speed up processes and adopt technology we cannot march through the twenty first century with the administrative systems of the nineteenth century]	[we have to change laws eliminate unnecessary procedures speed up processes and adopt technology we cannot march through the twenty first century with the administrative systems of the nineteenth century]	[we have to change laws eliminate unnecessary procedures speed up processes and adopt technology we cannot march through the twenty first century with the administrative systems of the nineteenth centuri]	[we have to change laws eliminate unnecessary procedures speed up processes and adopt technology we cannot march through the twenty first century with the administrative systems of the nineteenth centuri]

Figure 2. A sample of pre-process text.

2.4. Building Batches

To process the number of sentences in parallel in the network, batches are formed. The input in any feed-forward network is a matrix of shape [x*y] where x is batch size and y is feature size. The vocabulary is divided into sentences and from these sentences, the batch is formed. The first word of each sentence of the batch is processed in parallel then the second word of sentence of each batch and so on (Taneja, 2017).

2.5. Encodings

It is important to prepare the incoming data in a way that the computer can understand it. Two main encoding methods are one-hot encoding and word embedding. In this research, one-hot encoding is used.

2.6. Modelling

After preprocessing of input text RNN, LSTM and GRU are trained to build a language model. The language model deep learns the input text format speeches to automatically generate new speeches in the vein of Modhi's previous speeches on the given topic. First the parameters of network are set and the variables of TensorFlow are initialized for training. Then forward propagation is implemented for predicting word probabilities. After that loss is calculated. Training the algorithms with stochastic gradient descent, back-propagation through time and then to verify implementation gradient checking is done. Finally, text is generated. Set the success technical metrics and choose the best model(s) viable for solving the business question.

2.7. Evaluation

Loss is calculated to know how well the model behaves after each iteration or optimization. Minimizing the loss function with respect to parameters of the model using different optimization techniques like backpropagation is an objective in the learning model.

Perplexity $= \prod_{t=1}^{T} (1/P_{Lm} (x^{(t+1)}|x^{(t)},\ldots,x^{(1)}))^{(1/T)}$

Here T is the total words in the sentences.

Perplexity, an often-used metric for evaluating the effectiveness of generative models, is used as a measure of probability for a sentence to be produced by the model trained on a dataset. Lower the perplexity value, the better the model. It is normalized on the length of sentences. Generally, we want our probabilities to be high, which means the perplexity is low. When all the probabilities were 1, then the

perplexity would be 1 and the model would perfectly, faultlessly predict the text. Conversely, for poor language models, the perplexity will be higher (Charan, 2020).

2.8. Deployment

Deploying the model, monitoring the result and maintaining the model quality is done. An investigation is done to determine if the research needs more iteration or only needs frequent maintenance.

3. Results And Discussion

3.1. Results

RNN, LSTM and GRU were trained with TensorFlow to develop language model. The dataset with 467 speeches available in text format and manually labeled was divided into train, validation and test splits. 80% for training, 10% for validation and 10% for testing. Train set for training the model, validation set used to tune the hypo-parameters and test set to know how well the model is performed. Each algorithm is trained using same topic in order to compare each of them.

3.1.1. Training, Validation Loss And Perplexity Curves

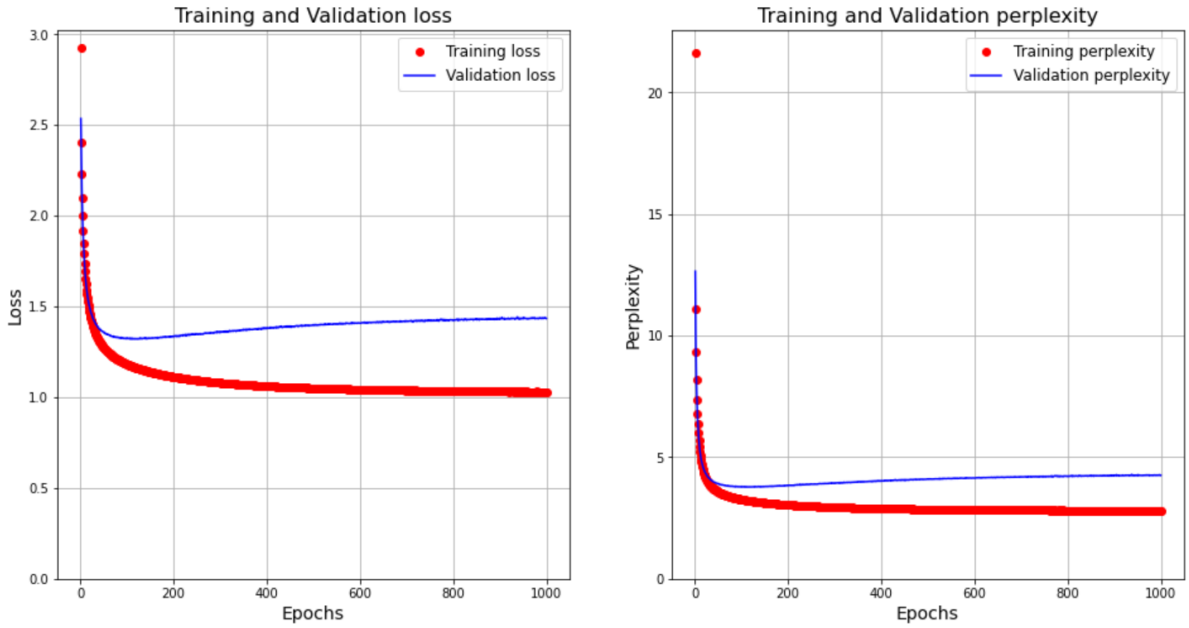


Figure 3. Training, validation loss and perplexity curves of RNN.

*Correspondence: gamini@sjp.ac.lk

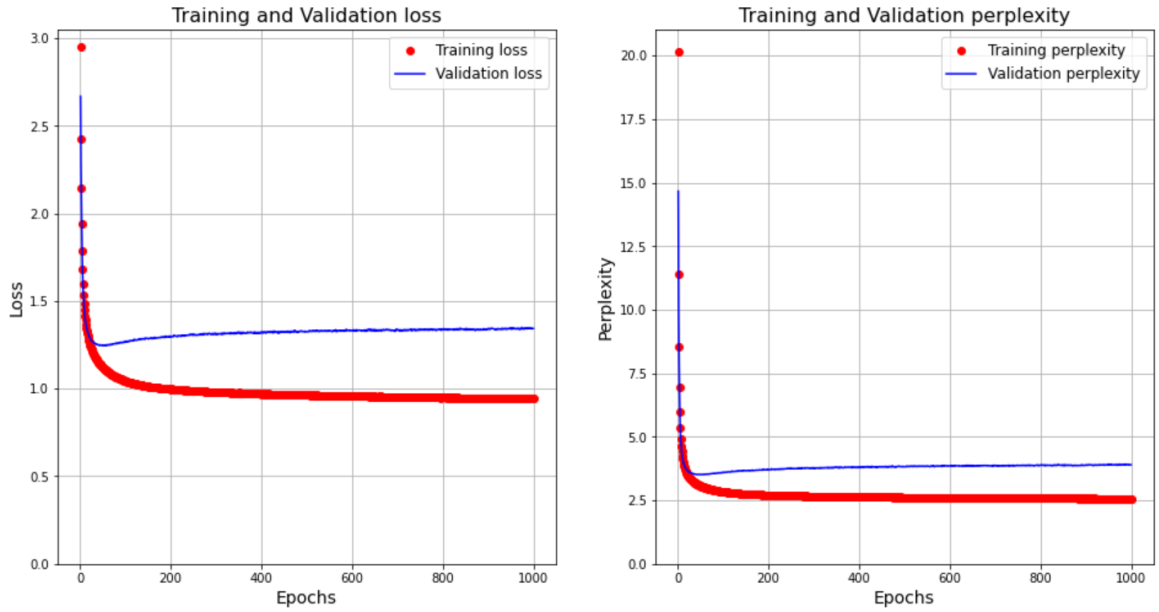


Figure 4. Training, validation loss and perplexity curves of LSTM.

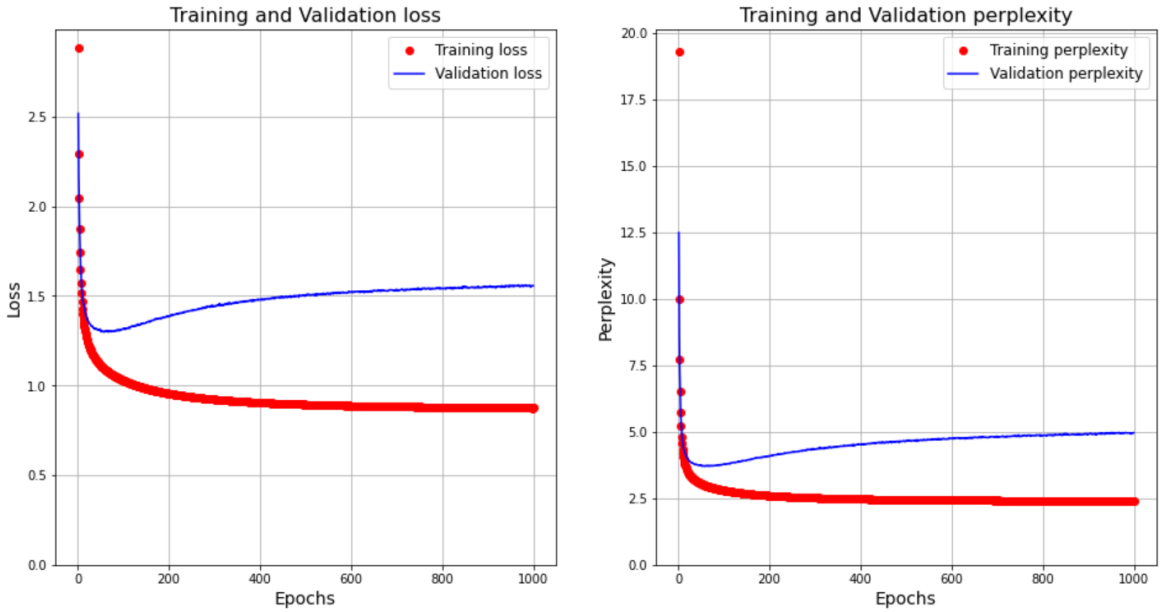


Figure 5. Training, validation loss and perplexity curves of GRU.

The plot between training, validation loss, training, validation perplexity of training the RNN, LSTM, GRU on dataset with 1000 epochs are shown in the above figures. It could be seen that training loss goes down, validation loss goes down and then it goes up. Training, validation perplexity curve is also monotonically decreasing and is algebraically equivalent to the inverse of the geometric mean per-word likelihood. A lower perplexity score specifies better generalization performance. The validation loss of LSTM is higher than the training loss, but it is not much high as in RNN. The highest variation between the validation loss and the training loss is found in GRU which means the model is not generalized to be

able to work on the real-life datasets. This is because GRU uses less training parameters and it executes faster and trains faster than LSTM. LSTM is more accurate on datasets with longer sequences.

3.1.2. Loss Curves

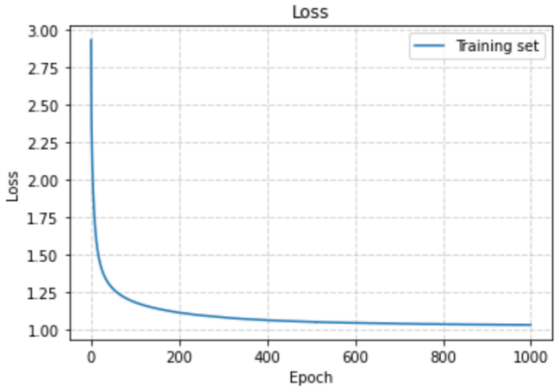


Figure 6. RNN model Loss vs Number of Epochs.

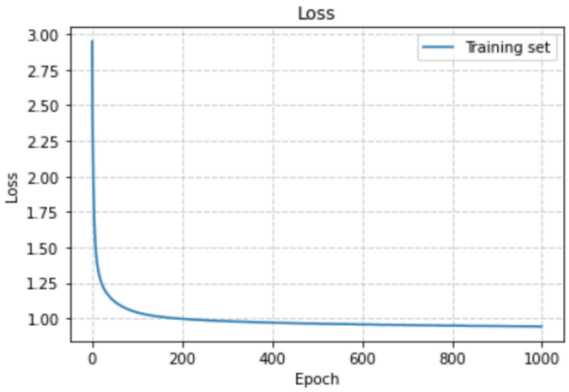


Figure 7. LSTM model Loss vs Number of Epochs.

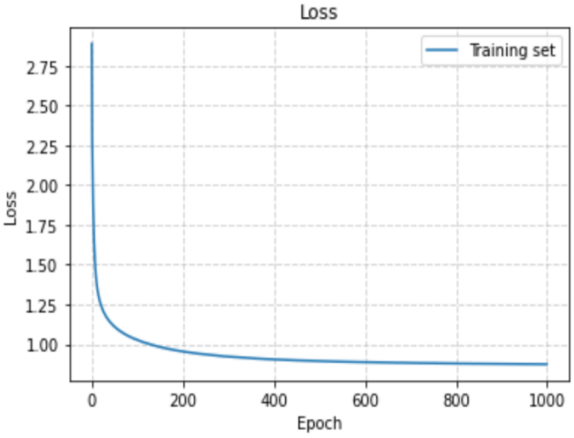


Figure 8. GRU model Loss vs Number of Epochs.

*Correspondence: gamini@sjp.ac.lk

Above figures give a summary on how each model loss varies with the number of epochs up to 1000.

3.1.3. Text Generated Outputs

Table 1: Mean probability, entropy and perplexity of generated outputs.

Algorithm	Mean probability	Entropy	Perplexity
RNN	0.5192999	1.456018	4.2888471663700685
LSTM	0.57575107	1.2644945	3.5413022909956346
GRU	0.5536837	1.4388195	4.1157163418303725

Mean Probability is high in LSTM and GRU. Entropy, perplexity are lower in LSTM and GRU.

Text Generated Outputs for The Keyword Independence Day.

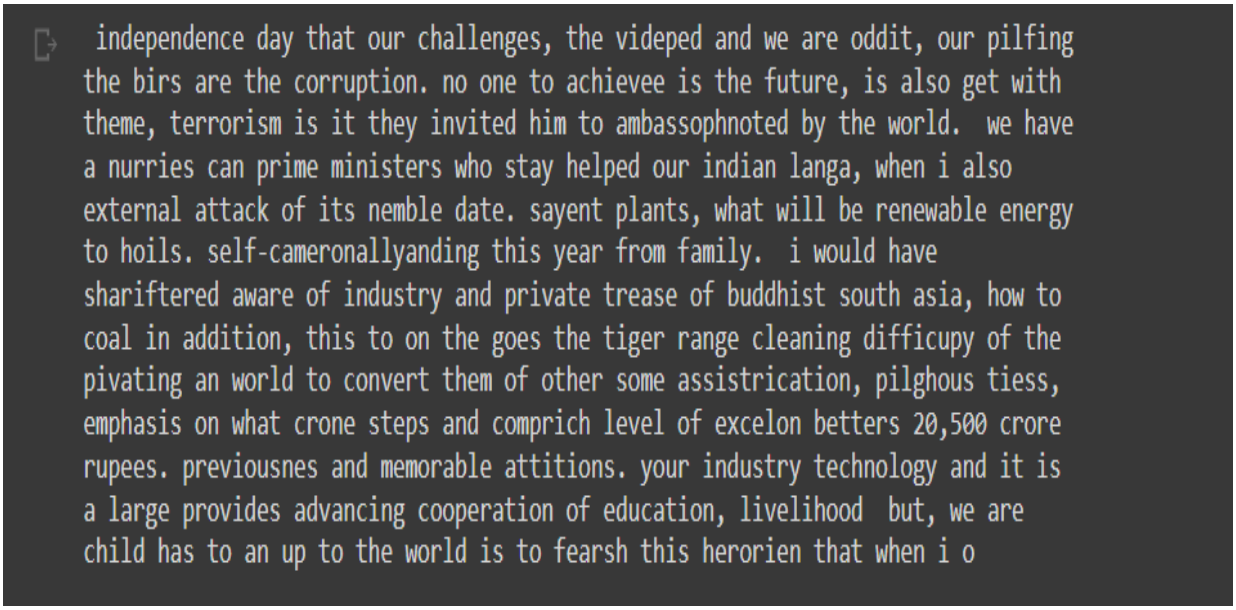


Figure 9. Generated outputs of Recurrent Neural Network.

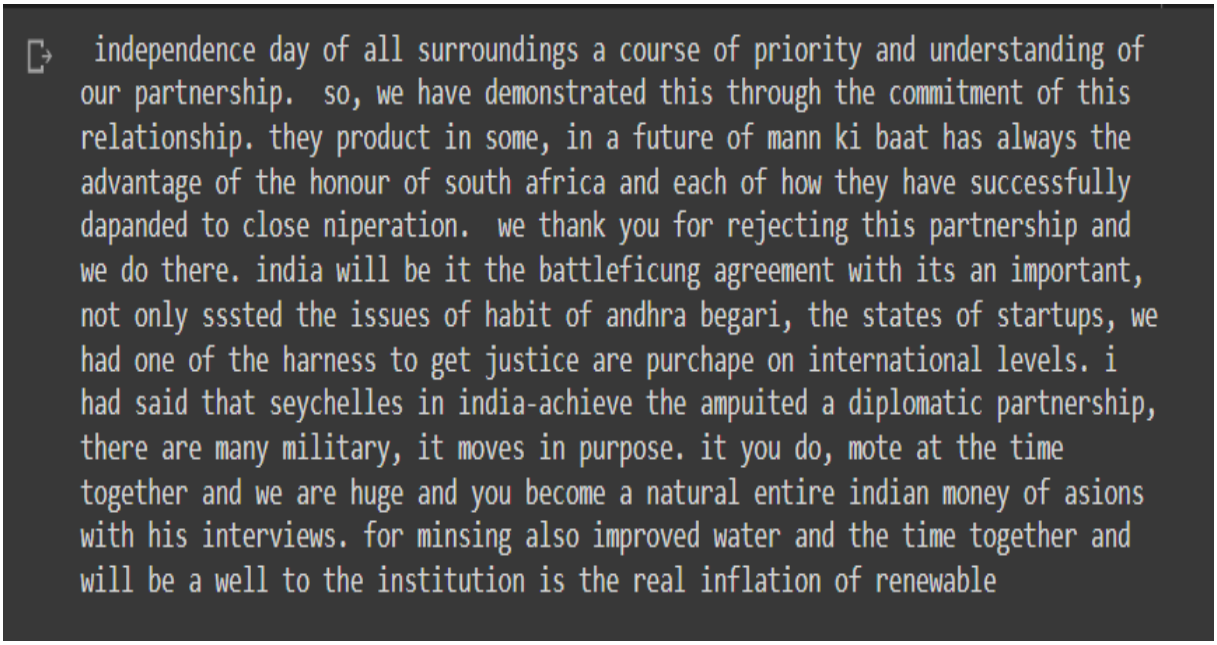


Figure 10. Generated outputs of Long Short-Term Memory.

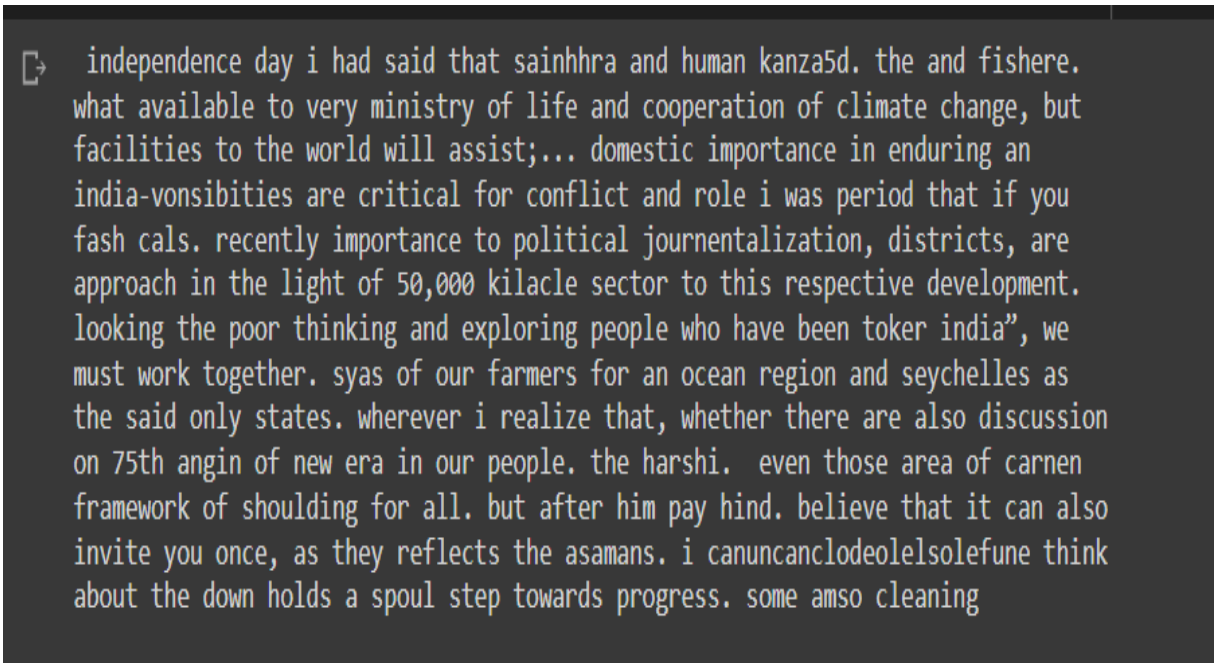


Fig. 11. Generated outputs of Gated Recurrent Unit.

*Correspondence: gamini@sjp.ac.lk

3.2. Discussion

Figs 3 and 5 suggest that the generated text using LSTM and GRU models has great potential. The output resulted by RNN is less realistic and pragmatic. But its variants LSTM and GRU performed better. Though the grammatical correctness and the sentence transitions were absent in generated text of LSTM and GRU, but their output is somewhat logical as compared to RNN.

Input dataset was trained with 1000 epochs, the plots between training and validation loss, training and validation perplexity curves are given in above. It could be seen that loss is smallest in case of GRU and largest in RNN. GRU uses less training parameters and it executes faster and trains faster than LSTM. LSTM is more accurate on dataset with longer sequences. LSTM is more generalized to be able to work on real life dataset.

Google coLab was used in this research which is a product from Google Research. It allows to write arbitrary Python code through the browser and is well suited to machine learning. All most all modules needed for data science technologies are already installed in it. It is hosted entirely on the Google cloud and provides two hardware accelerators such as GPU (Graphical Processing Unit), TPU (Tensor Processing Unit) (Charan, 2020).

One hot encoding was used for vector representation. It was used since the categorical feature is not ordinal and the number of categorical features is less so one hot encoding can be effectively applied (Purgato, 2021). It creates a new binary feature for each possible category and assigns a value of 1 to the feature of each sample that corresponds to its original category. Generated text does not show a very high quality in terms of grammatical correctness and sentence transitions since the dataset contains some text in Hindi alphabet.

Conclusions

Legitimately call that the research objectives have been successfully met. In summary, the following research objectives were accomplished with success:

A speech generation system that deep learns one text and then generates new text using machine learning approaches was developed. Different RNNs like simple RNN, LSTM and GRU were trained on English transcript of Narendra Modi which is available from free resources. Based on experimental results, LSTM and GRU performed better as it generated more realistic text and training loss is small, perplexity is small and mean probability is high compared to RNN.

The validation loss of LSTM is higher than the training loss, but it is not much high as in RNN. The highest variation between the validation loss and training loss is found in GRU which means the model is not generalized to able to work on real life dataset. This is because GRU uses less training parameters and it executes faster and trains faster than LSTM. LSTM is more accurate on dataset with longer sequences.

From the generated speeches LSTM and GRU models has great potential. The output resulted by RNN is less realistic and pragmatic. But its variants LSTM and GRU performed better. Though the grammatical correctness and the sentence transition were absent in generated text in LSTM and GRU, but their output is somewhat logical as compared to RNN.

References

Barman, P. P. & Boruah, A., 2018. A RNN based Approach for next word prediction in Assamese Phonetic Transcription, 8th International Conference on Advances in Computing and Communication (ICACC-2018),Dept of CSE, DUIET, Dibrugarh University, Dibrugarh-786004, Assam, India, pp 117-123.

Brazinskas, A., Lapata, M. & Titov, I., 2020. Unsupervised Opinion Summarization as Copycat-Review Generation, In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, pp 5151-5169

Bullock, J. & Luengo-Oroz, M., 2019. Automated Speech Generation from UN General Assembly Statements:Mapping Risks in AI Generated Texts, International Conference on Machine Learning AI for Social Good Workshop, Long Beach, United States

Charan, R., 2020. The Relationship Between Perplexity And Entropy In NLP. [Online] TOPBOTS Web.
https://www.topbots.com/perplexity-and-entropy-in-nlp/

Chung, J., Gulcehre, C., Cho, K. & Bengio, Y., 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. NIPS 2014 Workshop on Deep Learning.

Graves, A., 2014. Generating Sequences With Recurrent Neural Networks. Neural and Evolutionary Computing (cs.NE); Computation and Language (cs.CL)

Kassarnig, V., 2016. Political Speech Generation. Computation and Language (cs.CL).

Laskowski, N., 2018. recurrent neural networks. TechTarget Web. https://www.techtarget.com/searchenterpriseai/definition/recurrent-neural-networks

Le, T., Wang, S. & Lee, D., 2020. MALCOM: Generating Malicious Comments to Attack Neural Fake News Detection Models, Accepted at the 20th IEEE International Conference on Data Mining (ICDM 2020), The Pennsylvania State University, USA

Loye, G., 2022. Gated Recurrent Unit (GRU) With PyTorch. FloydHub Blog https://blog.floydhub.com/gru-with-pytorch/

Purgato, V. P., 2021. Google Colab and Why You Should Use It. Medium Web https://medium.com/mlearning-ai/google-colab-and-why-you-should-use-it-28bf64a04717

Ranzato, M. A., Chopra, S., Auli, M. & Zaremba, W., 2016. Sequence level training with recurrent neural networks, conference paper at ICLR

Simmons, B. (2022). What is the CRISP-DM methodology? Smart Vision Europe Web.
https://www.sv-europe.com/crisp-dm-methodology/.

Sethi, A., 2020. One-Hot Encoding vs. Label Encoding using Scikit-Learn. Analytics Vidhya Web
https://www.analyticsvidhya.com/blog/2020/03/one-hot-encoding-vs-label-encoding-using-scikit-learn/

Sutskever, I., Martens, J. & Hinto, G., 2011. Generating Text with Recurrent Neural Networks, Proceedings of the 28th International Conference on International Conference on Machine Learning, University of Toronto, 6 King's Rd pp , Toronto, ON M5S 3G4 Canada, pp 1017–1024

Taneja, P., 2017. Text Generation Using Different Recurrent Neural Networks. Computer Science and Engineering Department, Thapar University.

**Correspondence: gamini@sjp.ac.lk*

Yu, L., Zhang, W., Wang, J. & Yu, Y., 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient Adversarial Nets with Policy Gradient, Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17), Shanghai Jiao Tong University, University College London.