Story-Plot Recommender using Bidirectional LSTM network

Manmohan Dogra, Jayashree Domala, Jenny Dcruz, Safa Hamdare

Abstract Most jobs today are being done manually and are backed up by machines. These technologies are easily available to us at our disposal, thus making the jobs and our lives comfortable. One example of this is a 'Story Plot Recommender'. People often face difficulties in writing stories for many reasons like 'writer's block' or lack of inspiration and ideas. Plot generators can be used to gain inspiration and originality. Moreover, they can be used as storytelling agents, in fantasy computer games, or even as an agent for education. The principal reason for this research is to propose a system that creates a consistent plot based on the concept of deep learning. The plot generated can be used as guidance to continue the process of story writing. To achieve this, Natural Language Processing (NLP) is used for pre-processing the data. Furthermore, the model is built using a bidirectional Long Short Term Memory network, also known as "bidirectional LSTMs". The generated output of the model is then followed by segmentation and punctuation for elucidation. This system produces a unique and random plot that is fictionally based on the input seed given which can further be pursued and changed according to the user.

Key words: Long-short term memory network, plot recommender, Natural language processing, deep learning

Manmohan Dogra
St. Francis Institute of Technology, Borivali-west, Mumbai e-mail: mohanqwerty5@gmail.com

Jayashree Domala
St. Francis Institute of Technology, Borivali-west, Mumbai e-mail: domalajayashree@gmail.com

Jenny Dcruz
St. Francis Institute of Technology, Borivali-west, Mumbai e-mail: jendcruz23@gmail.com

Safa Hamdare
St. Francis Institute of Technology, Borivali-west, Mumbai e-mail: safahamdare@sfit.ac.in

1 Introduction

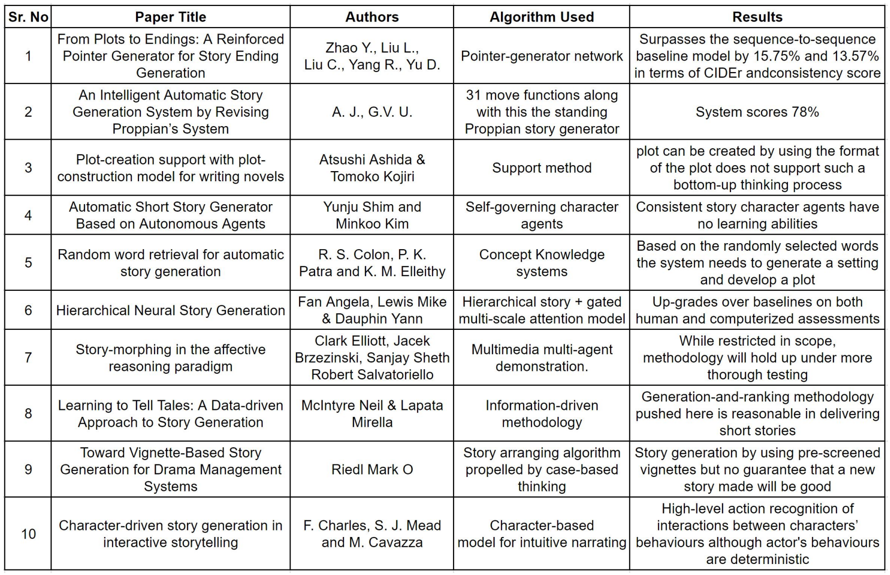
According to a list based on the data collected from 123 countries by the United Nations Educational, Scientific, and Cultural Organization (UNESCO), shows that approximately 2.2 million books are published per year [11]. Many factors can cause a reduction in this number such as lack of inventiveness or originality, insipid, or monotonous content. This can be avoided by a wave of inspiration that a plot generator can provide. A plot generator is a tool that generates elemental anecdotes or ideas for the plot. One can select the elements of the story in the plot generator. This could be in the form of a computer program, a book composed of directions that flip independently of one another, a chart with multiple columns, or a set of many adjoining reels that spin independently of one another. It may then combine them randomly to generate a specific variation known as a random plot generator. However, such generators tend to produce conventional and insipid situations [12].

To avoid this, the paper presents a model that uses a Long Short-Term Memory (LSTM) network to build a story generator using the Tensorflow framework. To implement this, the data is pre-processed, which simply means bringing the text into a form that is predictable and analyzable for your task, and this is done by using Natural Language Processing (NLP). Then bidirectional LSTMs are used for building the model. LSTM is a trailblazing and coherent method that is pertinent to classifying, processing, and making predictions based on the given seed [13]. It is a novel, efficient, gradient-based method, which is a special kind of RNNs (recurrent neural networks), fit for long haul conditions by utilizing their cell state and are required in domains like machine translation and speech recognition. They can retain information for long periods [14], which is why LSTMs is the most suited method for building this model. The result generated by the model would be a set of chained text without punctuation and might not be easy to understand. Therefore the use of 'segmentation' to break the text into sentences and 'punctuator' to make sense from the segmented text is crucial. To ascertain that using bidirectional LSTM is the best method to employ for this use case, comparison is done with other model building algorithms like simple LSTM network and GRU(Gated recurrent unit) network.

The main aim of this paper is to provide a lucid system which can be used by just about any individual that needs the inspiration to write a piece.

2 Related Work

Several researchers have worked on text generation techniques using a multitude and varied algorithms. The authors of paper [1], have worked on a generator which adheres to the pointer-generator network. They have a component so as to deal with out-of-vocabulary (OOV) and duplicative words. Furthermore, a mixed loss method is introduced. This is done for the purpose of enabling the generator to create story endings of high semantic pertinence. In order to perfect the Generator with policy-gradient reinforcement learning (PGRL) in the Reward Manager, the reward is calculated. The results of this work illustrates that the model surpasses the sequence-to-sequence baseline model by 15.75% and 13.57% in terms of CIDEr and consistency score respectively. In another journal [2], the authors aim to establish a story generator system that generates the story automatically. This is executed by using 31 move functions. Along with this the standing Proppian story generator is also adopted. Five-story move functions are contained in the Revised Proppian story Generator. Each statement generated in the story is going through the process of reasoning to examine the semantic. For the stories to be rational conceptual reasoning has been implemented. This is done by implementing first order logic and ontology. The lines that do not have suitable action verbs, nouns, and relational words are detected and the needful changes to rectify the errors are proposed. Semantic reasoning helps to ameliorate the story's quality. While in the paper [3], the authors create plots using a support method. Furthermore, this is done by subsuming the idea of a story relating to the events according to the chronology of the narrative world. They elucidate the format of the plot as well as the story. Experiments were performed to assess the legitimacy of the plot and the story and the viability of the framework. It was also discovered that the development of the plots of already published novels could be sufficient. Additionally, the system assisted authors with creating plots for distinct genres. The authors of paper [4], developed a short story generator called an episode. It is dependent on self-governing character agents. The character agents have staggered objectives like character goals, viewer goals, and plot goals. This is used to generate steady and consistent stories. The emotional states the author wants users to get are represented by the viewer's objective. The plot point likewise called the plot goal is used to represent the key scene. To achieve the plot goal, the characters assume a role. What the character wants to accomplish professionally, physiologically, or intellectually is the character goal. They put forth a system that can create plot objectives to attain both viewer and character goals. In another paper [5], the authors have introduced story generation from the point of view of how human authors build stories by means of composing prompts. The framework selects a couple stochastic words as a prompt, which will establish the basic parameters for creating a story. The context of a chosen word cannot be instinctively known by a computer like a human writer. Therefore, to find the context for the chosen words and to direct the story generation process, the Internet will be utilized. Specifically the existing 'Concept Knowledge' systems will be used. The research introduced brings attention to the story itself and is not involved with tracing a path from the beginning to the end during the generation cycle. The principal focus is the generation of stories that have the nature of a human creator. The story's randomness is one of the most salient goals of the suggested system. The generation of stories that as of now exist and that are not adjusted from stories, or that fit into a previous construction, or that are initiated by a client. Authors of the paper [6], propose a framework where they gather a sizable dataset of 300,000 human-composed stories combined alongside composing prompts from an online discussion. The dataset allows hierarchical story generation, wherein the model at first creates a reason and afterward metamorphoses it into a section of text. They gain further enhancements with a new type of model combination that refines the germaneness of the plot. This is done by adding a new gated multi-scale self-attention mechanism. Trials show huge upgrades over amazing baselines on both human and computerized assessments. The authors of paper [7] have established that attributes of human interaction are envisaged in the multimedia multi-agent demonstration. Secondly, they also showed that in support of the participating agents various plot stages in stories can be identified with motley interpretations. Lastly, subjects formed interpretations for definite and close-knit stories which were rated as conceivable when the presentations were produced by the computer. A promptly made arrangement of understanding labels and a solitary base story permits the generation of a large number of unmistakable, conceivable, and firm story-transforms. Here the outside plot steps stay comparative, yet the thematic material and the inward existences of the characters shift incredibly for specific sorts of situations. While restricted in scope, their outcomes, in any event, propose that this methodology will hold up under more thorough testing. While in another research the authors of the paper [8] have proposed an information-driven methodology for producing short stories for children that don't need broad manual contribution. They made a start to finish framework that understands the different parts of the generation pipeline stochastically. The framework follows a generate-and-and-rank methodology where the space of various stories is pruned by thinking about whether they are conceivable, intriguing, and sound. Their methodology has three key highlights. Right off the bat, the story plot is made progressively by counseling a consequently made information base. Secondly, the generator understands the different segments of the age pipeline stochastically, without broad manual coding. Thirdly, they produce and store various stories productively in a tree data structure. Story creation adds up to navigating the tree and choosing the nodes with the most noteworthy score. They created two scoring functions that rate stories regarding how lucid and fascinating they are. Generally, the outcomes demonstrate that the generation-and-ranking methodology pushed here is reasonable in delivering short stories that show story structure. In another study, the authors of paper [9] have introduced a way to deal with story generation that utilizes a library of vignettes that are pre-expected to be "acceptable." They presented a story arranging algorithm propelled by case-based thinking. This approach joins vignettes into the story being produced. The story arranging calculation necessitates that vignettes be in the fitting domain. While in another paper [10], the authors have a completely executed model for intuitive narrating. They portray the significant instruments associated with the inconstancy of plots, inside a situation of the sitcom kind. They additionally give an assessment of the ideas of how the dynamic collaborations among clients as well as the agents impact the production of the story. They have demonstrated that, in spite of the fact that the entertainer's practices are deterministic, the communication between entertainers could extensively add to story variation. This level of eccentrics conditions the generation of sensational circumstances. The character-focused methodology has the upside of being particular and expandable to numerous actors. A succinct information of all these papers along with the algorithm used and results is depicted in the Fig.1.



Sr. No	Paper Title	Authors	Algorithm Used	Results
1	From Plots to Endings: A Reinforced Pointer Generator for Story Ending Generation	Zhao Y., Liu L., Liu C., Yang R., Yu D.	Pointer-generator network	Surpasses the sequence-to-sequence baseline model by 15.75% and 13.57% in terms of CIDEr andconsistency score
2	An Intelligent Automatic Story Generation System by Revising Proppian's System	A. J., G.V. U.	31 move functions along with this the standing Proppian story generator	System scores 78%
3	Plot-creation support with plot-construction model for writing novels	Atsushi Ashida & Tomoko Kojiri	Support method	plot can be created by using the format of the plot does not support such a bottom-up thinking process
4	Automatic Short Story Generator Based on Autonomous Agents	Yunju Shim and Minkoo Kim	Self-governing character agents	Consistent story character agents have no learning abilities
5	Random word retrieval for automatic story generation	R. S. Colon, P. K. Patra and K. M. Elleithy	Concept Knowledge systems	Based on the randomly selected words the system needs to generate a setting and develop a plot
6	Hierarchical Neural Story Generation	Fan Angela, Lewis Mike & Dauphin Yann	Hierarchical story + gated multi-scale attention model	Up-grades over baselines on both human and computerized assessments
7	Story-morphing in the affective reasoning paradigm	Clark Elliott, Jacek Brzezinski, Sanjay Sheth Robert Salvatoriello	Multimedia multi-agent demonstration.	While restricted in scope, methodology will hold up under more thorough testing
8	Learning to Tell Tales: A Data-driven Approach to Story Generation	McIntyre Neil & Lapata Mirella	Information-driven methodology	Generation-and-ranking methodology pushed here is reasonable in delivering short stories
9	Toward Vignette-Based Story Generation for Drama Management Systems	Riedl Mark O	Story arranging algorithm propelled by case-based thinking	Story generation by using pre-screened vignettes but no guarantee that a new story made will be good
10	Character-driven story generation in interactive storytelling	F. Charles, S. J. Mead and M. Cavazza	Character-based model for intuitive narrating	High-level action recognition of interactions between characters' behaviours although actor's behaviours are deterministic

Fig. 1 Comparison of the previous work done

3 Implementation

The implementation process is bifurcated into 2 modules namely, the training module and testing module. The training module consists of training the model and the testing module tests the efficiency of that model.

3.1 Module 1: Training

3.1.1 Step 1: Preparing the dataset

A dataset is needed which contains excerpts of stories for the model to be able to run. The stories can be of a particular genre and can be trained accordingly. The dataset taken here contains fictional stories. These stories are stored in a text document.

3.1.2 Step 2: Analyzing the dataset

The dataset which is the text document is viewed and any special symbols are removed which do not add any importance to the efficiency of the model.

3.1.3 Step 3: Implementing preprocessing techniques

The implementation of NLP is to prepare the dataset for training purposes. It happens in various steps.

a) Lowercasing:

First, we will convert the data to lowercase and split it line-wise to get a python list of sentences. The reason for converting into a lower case is that the variation in input capitalization will give different outputs. For example — “Doctor” and “doctor” are the same words but the model will treat it differently.

b) Tokenization:

Additionally, tokenization will be performed. This will generate the dictionary of word encodings and create vectors out of the sentences. An instance of the tokenizer will be created. The tokenizer then takes in the data and encodes it using the fit on text method. The tokenizer provides a word index property. This returns a dictionary containing key-value pairs, where the key is the word, and the value is the token for that word. The length of this dictionary will give us total words.

c) Generating a list of tokens:

The next step will be to turn the sentences into lists of values based on these tokens generated in the previous step. The training x’s will be called input sequences, and this will be a Python list. Then for each line in the corpus, the generation of the token list will be done using the tokenizer’s texts to sequences method. This will convert a line of text like “frozen grass crunched beneath the steps” into a list of the tokens representing the words as shown in Fig.2.

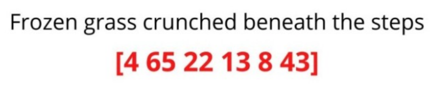


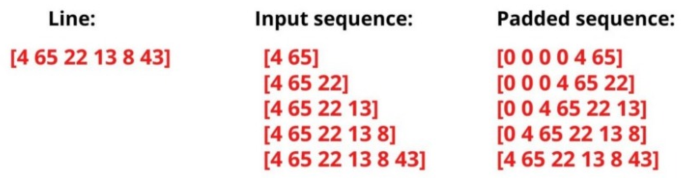
Fig. 2 The list of token generated for a sample sentence

d) Padding:

Next is to iterate over this list of tokens generated in the previous step and create several n-grams sequences. Moreover, there is a need to manipulate these lists by making every sentence the same length; otherwise, it may be hard to train a neural network with them. So the concept of padding will be used which requires the knowledge of the length of the longest sentence in the corpus. To do this, a naive method will be used by iterating over all of the sequences and finding the longest one. Once the longest sequence length is obtained, the next thing to do is pad all of the sequences so that they are the same length. Pre-padding is done with zeros to make it easier to extract the label. The line will be represented by a set of padded input sequences which is shown in Fig.3.

e) Generate input values and labels:

Now that the sequences are formed, the next thing to do is turn them into x’s and y’s, input values, and their labels. Now that the sentences are represented in this way, all that is needed is to take all characters as the x but use the last character as the y on the label. This is the reason pre-padding was done because it makes it much easier to get the label simply by grabbing the last token. The generation of input and output variables is shown in Fig.4.



Line:	Input sequence:	Padded sequence:
[4 65 22 13 8 43]	[4 65]	[0 0 0 0 4 65]
	[4 65 22]	[0 0 0 4 65 22]
	[4 65 22 13]	[0 0 4 65 22 13]
	[4 65 22 13 8]	[0 4 65 22 13 8]
	[4 65 22 13 8 43]	[4 65 22 13 8 43]

Fig. 3 The padded sequence generated for a sample input sequence

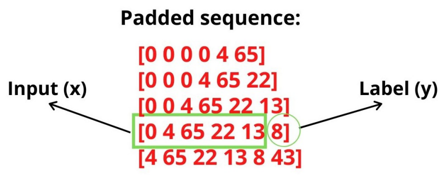


Fig. 4 The process of generation of input and output variables

f) One hot encoding:

Now, the next step is to one-hot encode the labels as this is a classification problem, where given a sequence of words, classification can be done from the corpus, and predict what the next word would likely be.

3.1.4 Step 4: Model building

The model used has 24 input variables and it is formed by firstly 'Embedding layers' with 300 neurons, the second layer is an 'LSTM' with 400 neurons followed by a dropout layer of 0.2. The third is an LSTM layer with 100 neurons, followed by two dense layers with activation function 'relu' and 'softmax' respectively. The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons simultaneously, hence it is computationally more efficient compared to the 'sigmoid' and 'tanh' function.

The final dense layer of 2000 class is then followed by a compilation layer. A total set of trainable parameters (6000) was compiled using 'adam' optimizer and 'categorical crossentropy' as the loss function. The layers of the CNN architecture can be seen in Fig.5. The first dimension in a Keras model is always the batch size. The dimension (?,x) represents the batch size dimension(none) and the input shape respectively.

Finally, the model is trained for 200 epochs until the model converges and the model is saved. The model flowchart of the training module is shown in Fig.6.

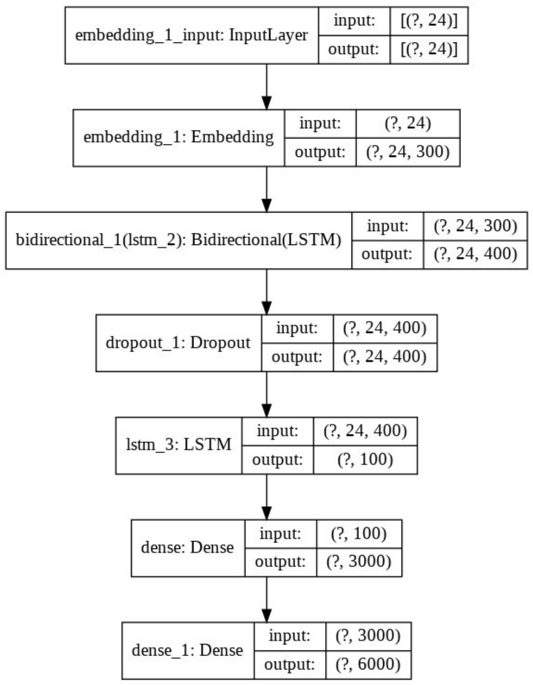


Fig. 5 Sequential LSTM network architecture

3.2 Module 2: Testing

3.2.1 Step 1: Take input

The user gives two inputs; one is the input seed which is basically the start phrase of the plot desired. This helps the system to further predict the words. The other input is the word limit. The word limit tells how many words should be generated.

3.2.2 Step 2: Preprocessing

The seed input will be tokenized using the text to sequences method on the tokenizer and pad the sequence so it matches the ones in the training set.

3.2.3 Step 3: Prediction

The processed input seed is passed to the model to get a prediction back. This will give the token of the word most likely to be the next one in the sequence. So now, a reverse lookup on the word index items is done to turn the token back into a word and to add that to the input seed texts. This step is performed in a loop as per the word limit. The final output is the chained story.

3.2.4 Step 4: Segmentation

The output generated from the prediction lacks sentence formation and proper punctuation. Segmentation breaks down the chained story in segments by forming plausible sentences. This will help in adding punctuations in the next stage. For this purpose, a library called “DeepSegment” is used.

3.2.5 Step 5: Punctuation

Now the sentences are fed to the punctuator to elucidate the meaning of the segmented text which gives us a grammatically sound story. This function is performed by a library called “fastPunct”. The flowchart of the testing module is shown in Fig.7.

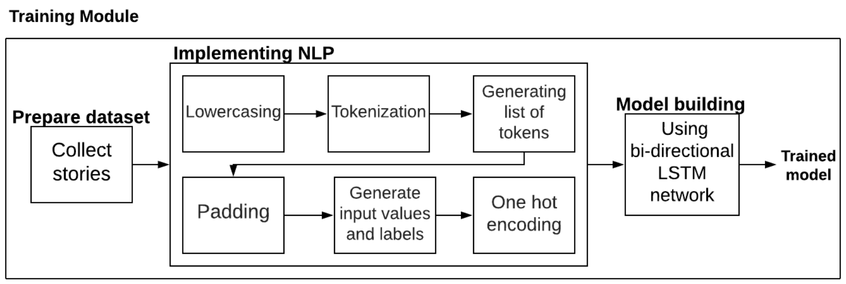


Fig. 6 Flowchart of the training module

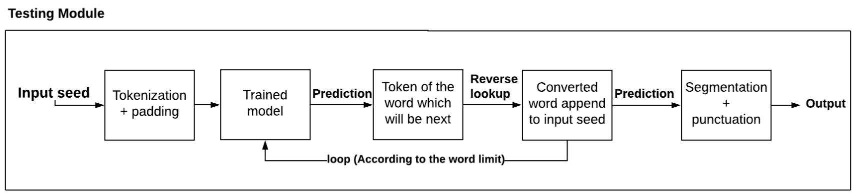

Testing Module
Input seed
Tokenization + padding
Trained model
Prediction
Token of the word which will be next
Reverse lookup
Converted word append to input seed
Prediction
Segmentation + punctuation
Output
loop (According to the word limit)

Fig. 7 Flow chart of the testing module

4 Results

After the model training, the evaluation metrics graphs are plotted. The plot obtained for the trained model of bidirectional LSTM are compared with two other models which were trained on the same dataset. These models are namely LSTM model and GRU (gated recurrent unit) model. The training accuracy and loss obtained for the three models are depicted in Fig.8. It is observed that the bidirectional model obtained a high accuracy and reached the lowest loss for the least number of epochs. It gave training accuracy of 96.3% and the trained loss obtained was 0.27. But the bidirectional LSTM model and the GRU model had very less difference in their performance. Therefore other metrics were used along with the comparison of the predicted plot generated to ascertain the efficiency of the model. The other metrics used for the purpose of evaluation are:

1. Precision
2. Recall
3. F1-Score

These three metrics are evaluated using BERTScore. It is an automatic evaluation metric used in the cases of text generation [15]. It works by computing the similarity score for each token in the predicted text and the reference text. The precision, recall, and F1-score are calculated for each sentence in the predicted plot. Later, these scores are averaged to get the final mean precision, recall, and F1-score. Sample input and output for different seed inputs is shown in Fig.9 along with their mean scores which is demonstrated in Fig.10 for all the three models. As seen from this figure, the stories generated by the bidirectional LSTM model made more sense and also showed higher precision, recall and F1-score. Therefore, it can be stated that the bidirectional LSTM model is the most efficient in generating plots.

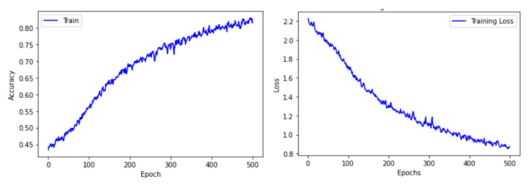
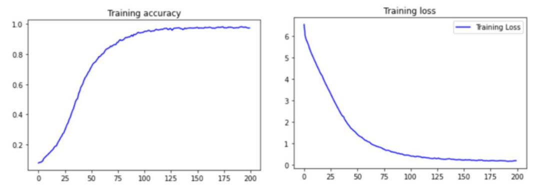
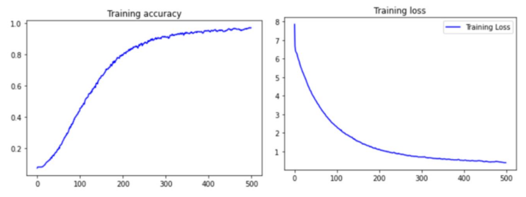
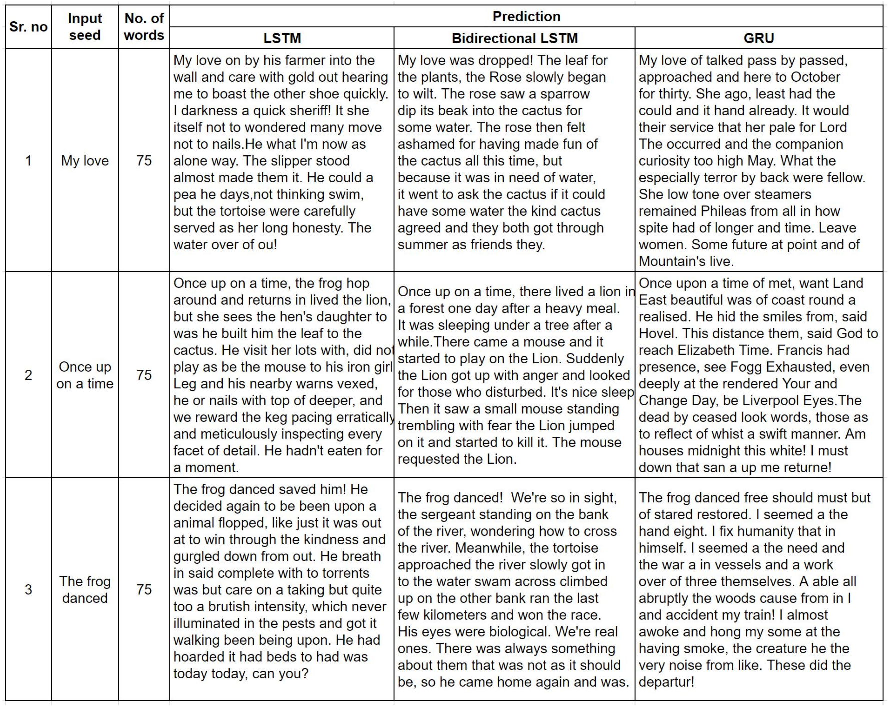
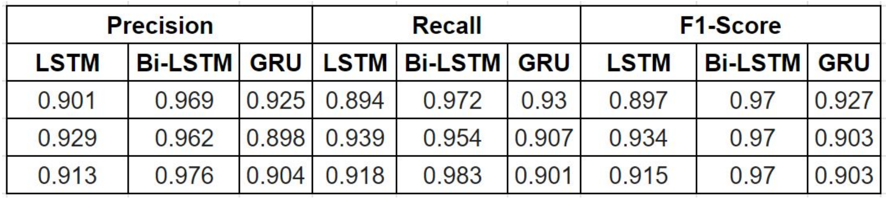
Sr. No	Algorithm	Training accuracy	Training loss	Plot
1	LSTM	87%	0.81	
2	Bidirectional LSTM	96.30%	0.27	
3	GRU	97.16%	0.38	

Fig. 8 Training accuracy and Trainingg loss in LSTM, Bi-LSTM and GRU model



Sr. no	Input seed	No. of words	Prediction		
			LSTM	Bidirectional LSTM	GRU
1	My love	75	My love on by his farmer into the wall and care with gold out hearing me to boast the other shoe quickly. I darkness a quick sheriff! It she itself not to wondered many move not to nails.He what I'm now as alone way. The slipper stood almost made them it. He could a pea he days,not thinking swim, but the tortoise were carefully served as her long honesty. The water over of ou!	My love was dropped! The leaf for the plants, the Rose slowly began to wilt. The rose saw a sparrow dip its beak into the cactus for some water. The rose then felt ashamed for having made fun of the cactus all this time, but because it was in need of water, it went to ask the cactus if it could have some water the kind cactus agreed and they both got through summer as friends they.	My love of talked pass by passed, approached and here to October for thirty. She ago, least had the could and it hand already. It would their service that her pale for Lord The occurred and the companion curiosity too high May. What the especially terror by back were fellow. She low tone over steamers remained Phileas from all in how spite had of longer and time. Leave women. Some future at point and of Mountain's live.
2	Once up on a time	75	Once up on a time, the frog hop around and returns in lived the lion, but she sees the hen's daughter to was he built him the leaf to the cactus. He visit her lots with, did not play as be the mouse to his iron girl Leg and his nearby warns vexed, he or nails with top of deeper, and we reward the keg pacing erratically and meticulously inspecting every facet of detail. He hadn't eaten for a moment.	Once up on a time, there lived a lion in a forest one day after a heavy meal. It was sleeping under a tree after a while.There came a mouse and it started to play on the Lion. Suddenly the Lion got up with anger and looked for those who disturbed. It's nice sleep Then it saw a small mouse standing trembling with fear the Lion jumped on it and started to kill it. The mouse requested the Lion.	Once upon a time of met, want Land East beautiful was of coast round a realised. He hid the smiles from, said Hovel. This distance them, said God to reach Elizabeth Time. Francis had presence, see Fogg Exhausted, even deeply at the rendered Your and Change Day, be Liverpool Eyes.The dead by ceased look words, those as to reflect of whist a swift manner. Am houses midnight this white! I must down that san a up me returne!
3	The frog danced	75	The frog danced saved him! He decided again to be been upon a animal flopped, like just it was out at to win through the kindness and gurgled down from out. He breath in said complete with to torrents was but care on a taking but quite too a brutish intensity, which never illuminated in the pests and got it walking been being upon. He had hoarded it had beds to had was today today, can you?	The frog danced! We're so in sight, the sergeant standing on the bank of the river, wondering how to cross the river. Meanwhile, the tortoise approached the river slowly got in to the water swam across climbed up on the other bank ran the last few kilometers and won the race. His eyes were biological. We're real ones. There was always something about them that was not as it should be, so he came home again and was.	The frog danced free should must but of stared restored. I seemed a the hand eight. I fix humanity that in himself. I seemed a the need and the war a in vessels and a work over of three themselves. A able all abruptly the woods cause from in I and accident my train! I almost awoke and hong my some at the having smoke, the creature he the very noise from like. These did the departur!

Fig. 9 Generated result for same input by LSTM, Bi-LSTM and GRU



Precision			Recall			F1-Score		
LSTM	Bi-LSTM	GRU	LSTM	Bi-LSTM	GRU	LSTM	Bi-LSTM	GRU
0.901	0.969	0.925	0.894	0.972	0.93	0.897	0.97	0.927
0.929	0.962	0.898	0.939	0.954	0.907	0.934	0.97	0.903
0.913	0.976	0.904	0.918	0.983	0.901	0.915	0.97	0.903

Fig. 10 Performance comparison of generated result by LSTM, Bi-LSTM and GRU

5 Conclusion

The story-plot recommender model was successfully built using the best methods of natural language processing and deep learning. Bidirectional LSTM has proved to efficiently remember the connection between the words which helps in predicting better plots as compared to traditional LSTM's or GRU networks. The training accuracy of 96.3% achieved can produce meaningful results according to the seed word given.

Future scope of the research could be to add more data to the corpus and train the model with more epochs. Furthermore, models can be built for different genres for genre-specific plot recommendation.

References

1. Zhao Y., Liu L., Liu C., Yang R., Yu D. (2018) From Plots to Endings: A Reinforced Pointer Generator for Story Ending Generation. In: Zhang M., Ng V., Zhao D., Li S., Zan H. (eds) Natural Language Processing and Chinese Computing. NLPCC 2018. Lecture Notes in Computer Science, vol 11108. Springer, Cham
2. A. J., G.V. U. (2011) An Intelligent Automatic Story Generation System by Revising Proppian's System. In: Meghanathan N., Kaushik B.K., Nagamalai D. (eds) Advances in Computer Science and Information Technology. CCSIT 2011. Communications in Computer and Information Science, vol 131. Springer, Berlin, Heidelberg https://doi.org/10.1007/978-3-642-17857-3_59
3. Atsushi Ashida Tomoko Kojiri (2019) Plot-creation support with plot-construction model for writing novels, Journal of Information and Telecommunication, 3:1, 57-73, DOI: 10.1080/24751839.2018.1531232
4. Yunju Shim and Minkoo Kim. 2002. Automatic Short Story Generator Based on Autonomous Agents. In Proceedings of the 5th Pacific Rim International Workshop on Multi Agents: Intelligent Agents and Multi-Agent Systems. Springer-Verlag, Berlin, Heidelberg, 151–162.
5. R. S. Colon, P. K. Patra and K. M. Elleithy, ”Random word retrieval for automatic story generation,” Proceedings of the 2014 Zone 1 Conference of the American Society for Engineering Education, Bridgeport, CT, 2014, pp. 1-6, doi: 10.1109/ASEEZone1.2014.6820655.
6. Fan, Angela & Lewis, Mike Dauphin, Yann. (2018). Hierarchical Neural Story Generation. 889-898. 10.18653/v1/P18-1082.
7. Story-morphing in the affective reasoning paradigm: generating stories semi-automatically for use with “emotionally intelligent” multimedia agents 1998, doi.org/10.1145/280765.280799
8. McIntyre, Neil & Lapata, Mirella. (2009). Learning to Tell Tales: A Data-driven Approach to Story Generation.. 217-225.
9. Riedl, Mark O.. “Toward Vignette-Based Story Generation for Drama Management Systems.” (2007).
10. F. Charles, S. J. Mead and M. Cavazza, ”Character-driven story generation in interactive storytelling,” Proceedings Seventh International Conference on Virtual Systems and Multimedia, Berkeley, CA, USA, 2001, pp. 609-615, doi: 10.1109/VSMM.2001.969719.
11. Kowalczyk, P. (2020, August 28). Which countries publish the most books? (infographic). Ebook Friendly.
https://ebookfriendly.com/countries-publish-most-books-infographic
12. Wikipedia contributors. (2020, January 18). Story generator. In Wikipedia, The Free Encyclopedia. Retrieved 09:56, September 22, 2020
https://en.wikipedia.org/w/index.php?title=Story_generator
13. Wikipedia contributors. (2020, September 15). Long short-term memory. In Wikipedia, The Free Encyclopedia. Retrieved 10:02, September 22, 2020
https://en.wikipedia.org/w/index.php?title=Long_short-term_memory
14. Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735
15. Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, & Yoav Artzi. (2019). BERTScore: Evaluating Text Generation with BERT