

Deep Learning-LSTM based Football Commentary Generation and PCFG based Event Generation Dependent on User Input

Priyadarshan Dhabe¹, Koushal Sadavarte², Neel Kulkarni³, Shivam Lagdive⁴, Eshan Mehendale⁵ and Saloni Nimgaonkar⁶

¹⁻⁶Department of Artificial Intelligence and Data Science, Vishwakarma Institute of Technology, Pune, 411037, Maharashtra, India

Email: priyadarshan.dhabe@vit.edu, sunil.koushal21, neel.kulkarni21, shivam.lagdive21, eshan.mehendale21, saloni.nimgaonkar21@vit.edu

Abstract— Live commentary is a pivotal element of sports broadcasting, furnishing real- time updates and analysis to enhance the viewing experience. This exploration paper presents a new approach for live commentary generation using intermittent Neural Networks (RNN). This exploration opens up new avenues for automated live commentary systems, offering real- time updates that allure cult and elevate the sports broadcasting experience The suggested RNN model is trained on a huge dataset of actual match data leveraging the successional character of sporting events. The RNN-LSTM armature is intended to detect temporal correlations in sporting events and generate contextually consistent commentary.

Index Terms— RNN, PCFG, Football, soccer, Deep learning.

I. INTRODUCTION

RNN and LSTM are deep learning techniques that revolutionized the field of neural networks. RNNs are neural networks designed to process linked data using linked feedback. Unlike traditional neural networks, RNNs can store information about past inputs, allowing them to correlate contents and store long-term expectations. This feature makes RNNs well suited for tasks where the current output depends not only on the current input but also on previous inputs in the system. The LSTM network is an RNN architecture that solves the fading problem by introducing memory and gating techniques. Brain memory enables LSTMs to store and access information in long strings.

The purpose of this research is to explore the use of RNN and LSTM models for competing feedback. The plan aims to capture the dynamics of the game using the continuity of live events and generate context-sensitive reviews in real time. The ability of RNNs and LSTMs to learn from the past and combine physical data makes them suitable for this task; where models are trained on historical sports information data, allowing them to learn the differences between different sports and adapt to dynamic situations. living conditions. A thorough review is done to assess the quality, accuracy and validity of the reviews produced.

II. LITERATURE REVIEW

Applying machine learning techniques to live game feedback creates the ability to automate processes, provide

real-time updates and enhance the viewer experience during events. Leveraging the power of supervised learning, RNNs, LSTMs, and datasets, machine learning models can generate accurate content and deploy analytics that provide insights to audiences.

[1] Graves, A. (2013). This article discusses the use of RNNs, including LSTMs, to generate sequences. It explores various techniques such as time-backpropagation and gradient clipping to train RNNs. The authors present applications of RNNs for text recognition, music and speech recognition. This article also discusses the problems of training RNNs such as overfitting and vanishing/exploding gradients. This useful paper discusses RNN-based Seq2Seq models, particularly LSTMs, for applications such as machine translation and speech recognition.

[2] Li Wen., Zhang, X. And Tang, Y. (2017). This research paper focuses on generating sports-related information using LSTM networks. The authors present a model that preserves the physical prospects of competing events and makes it integrated and context sensitive. They explore how to increase the performance of LSTM-based models by using preprocessing approaches like as segmentation and feature extraction. The article also addresses the issues of processing noise data, generating different analyzes and adapting models for different sports. Performance evaluation and evaluation of the planning process include metrics such as social media reviews, relationships with similar events, and user satisfaction. The results show that the foundation is able to create informative and entertaining reviews that increase the audience's understanding and enjoyment of the game.

LSTM is a type of Recurrent Neural Network RNN architecture that has received a lot of traction in the field of deep learning, especially for sequence modelling applications. Hochreiter and Schmidhuber proposed LSTM networks in 1997 as a fix to the vanishing gradient problem encountered by regular RNNs.

The vanishing gradient problem is caused by the gradient signal shrinking exponentially over time when errors are back propagated across numerous time steps. This issue inhibits typical RNNs' capacity to detect relationships that last in sequential data. Memory cells are used in LSTM networks to overcome this issue. and gating mechanisms that allow the network to selectively store and access information over long periods.

The cell state, input gate, forget gate, and output gate are the main elements of an LSTM cell. The network's memory is the cell state, and the gates control the flow of information. The input gate governs the flow of freshly acquired data into the cell state, the forget gate dictates which information is to be forgotten from the cell state, and the output gate determines which information is output from the cell state. These gating mechanisms enable LSTMs to retain relevant information over long sequences while suppressing irrelevant or noisy inputs.

LSTMs have been effectively used to a broad range of problems, such as natural language processing, speech recognition, machine translation, time series forecasting, and others. They have demonstrated superior performance compared to traditional RNNs in handling long-range dependencies and capturing complex patterns in sequential data. In the literature, several modifications and expansions to the fundamental LSTM architecture have been presented. The Gated Recurrent Unit (GRU) is one such version, which simplifies the LSTM design by merging the forget and input gates into a single update gate. GRUs achieve similar performance to LSTMs while having a smaller number of parameters.

Researchers have recently concentrated on enhancing the efficiency and interpretability of LSTM models. Techniques such as bidirectional LSTMs, attention mechanisms, and stacked or deep LSTM architectures have been explored to enhance the representation and modeling capabilities of LSTM networks.

III. METHODOLOGY / EXPERIMENTAL

In Below is the general methodology used in this project.

A. Data Collection

Gather the commentary data from reliable sources. This can be in the form of a text file, dataset, or web scraping.

B. Data Preprocessing

Preprocess the commentary data to make it suitable for training the LSTM model. Some of the preprocessing steps that are performed include:

- Removing unnecessary characters or symbols.
- Tokenizing the text into individual words or sub-words.
- Removing stop-words or performing other text cleaning techniques.
- Encoding the text data into numeric representations (e.g., word embeddings).

C. Data Preparation

Divide the preprocessed data into two sets: training and testing. Ascertain that the dataset is sufficiently representative and diversified for successful model training and assessment.

D. Model Architecture

This involves determining the number of LSTM layers, hidden units, and other hyper parameters. Consider using techniques such as dropout, batch normalization, or bidirectional LSTMs to improve model performance.

E. Model Training

Using the training dataset, train the LSTM model. Model parameters including learning rate and batch size can be tweaked to improve model convergence and performance. Track the model's performance by monitoring training performance metrics

F. Model Evaluation

The model will be evaluated based on its performance on testing dataset. Calculate metrics such as perplexity, BLEU score, or accuracy to assess the model's performance in generating text based on the given inputs.

G. Model Fine-tuning

If the model performance is not satisfactory, consider fine-tuning the model by adjusting hyperparameters or experimenting with different architectures. This iterative process helps optimize the model for better results.

H. Text Generation

Use the trained LSTM model to generate text based on given inputs. Provide relevant inputs, such as the event type or player names, to the model and observe the generated output. Fine-tune the text generation process as needed to achieve desired results.

I. Iterative Refinement

Continuously analyze the generated text and refine the model and data preprocessing steps based on feedback and insights. Iterate through steps 4 to 8 to improve the model's performance and text generation quality.

J. Documentation and Reporting

Document the entire process, including the data preprocessing steps, model architecture, training details, evaluation metrics, and any insights or observations made during the project. Summarize the findings and results in a clear and concise manner. The next step was to implement the system, which involves tokenization, part-of-speech tagging, parsing, and applying the CFG rules to identify the emotions in the input text. The system outputs the dominant emotion in the text as either happy, sad or neutral.

Data Collection: Gather a sufficient amount of commentary data related to the sports events you are interested in. The data should include descriptions of various events, such as goals scored, fouls conceded, cards shown, corners conceded, etc.

1. *Data Preprocessing:* Clean and preprocess the collected data to remove irrelevant information, correct spelling errors, and handle any inconsistencies in the data. This step may involve tokenizing the text into words, removing stop-words, and performing other text preprocessing techniques.
2. *Annotation:* Annotate the collected commentary data with relevant labels for different events, such as "Goal scored," "Foul conceded," "Card shown," "Corner conceded," etc. This step is crucial for creating a labeled dataset for training and evaluating the LSTM model.
3. *Dataset Split:* Divide the annotated dataset into three parts: training, testing, and validation. The training collection will be utilized to train the LSTM model, the validation set for hyper parameter tuning and model evaluation during training, and the testing set for evaluating the performance of the final trained model.
4. *Encoding:* Convert the annotated commentary data into a suitable numerical representation that can be processed by the LSTM model. This may involve techniques such as word embedding or one-hot encoding to represent words as vectors.
5. *LSTM Model Architecture:* Design and build an LSTM-based model for sequence generation. Specify the model's input and output structures. The input sequence will be the encoded commentary data, and the output sequence will be the generated sentences based on the input.
6. *Training:* Using the training set, train the LSTM model. Adjust hyperparameters such as the number of LSTM layers, hidden units, learning rate, and batch size to optimise the model. Iteratively train the model, assessing performance on the validation set and changing the model as needed.

7. *Evaluation*: Examine the performance of the developed LSTM model on the test data set. Metrics like accuracy, precision, recall, and F1 score can be used to analyze how well the model generates relevant and precise sentences depending on the input.
8. *Generation*: Use the trained LSTM model to generate sentences based on the user's input. Preprocess the user's input to align it with the model's input requirements. To construct the desired output sentence, run the preprocessed input through the trained model.
9. *Iterative Refinement*: Iterate on the model and methodology based on the evaluation results and feedback. Fine-tune the model architecture, experiment with different hyperparameters, and explore additional methods for increasing the quality and variety of generated phrases.
10. *Deployment*: Once you are satisfied with the LSTM model's performance, deploy it in a suitable environment for practical usage. Develop an interface or application where users can input their desired events, and the model generates corresponding sentences based on the input.
11. input.

A. LSTM

The LSTM architecture is a type of RNN architecture that has been developed to solve the challenge of collecting distant dependencies in sequential data.

The vanishing gradient problem, which affects conventional RNNs, makes it challenging for them to remember details from prior distant time steps. LSTM overcomes this limitation by introducing a memory cell and several additional mechanisms that enable it to selectively hold or discard information over numerous time steps.

Here's a brief explanation of the key components of an LSTM:

Memory Cell: The memory cell is an LSTM's main building block. The internal state of the LSTM is continuously updated and maintained. The cells also act as a long-term memory that can store data from earlier time steps as well.

Gates: The information entering and leaving the memory cell is managed by gates in an LSTM. These gates are implemented as neural network layers that can learn to selectively let information through. The three main types of gates are:

Forget Gate: Decides which information to remove from the previous cell state.

Input Gate: Determines which new information to store in the current cell state.

Output Gate: Decides which data from the current cell state should be output.

Cell State Update: The forget gate, input gate, and output gate are used to update the cell state. The input gate selects the new information to be saved to the cell state, the output gate selects the information to be output from the current cell state, and the forget gate selects which portions of the prior cell state to forget.

Hidden State: The output that the LSTM produces at each time step constitutes the hidden state of the LSTM. It is derived from the current cell state but has additional transformations applied to it. The hidden state can be used to make predictions or feed into subsequent layers of the network.

By using these mechanisms, in sequential data, LSTM can detect long-term dependencies. It can retain important information over long periods, selectively forget irrelevant information, and update the internal state with new relevant information. LSTM has been widely utilized in numerous jobs involving natural language processing, including language modeling, machine translation, sentiment analysis, and text generation. Its ability to model and remember long-range dependencies makes it particularly effective in handling sequences of text data.

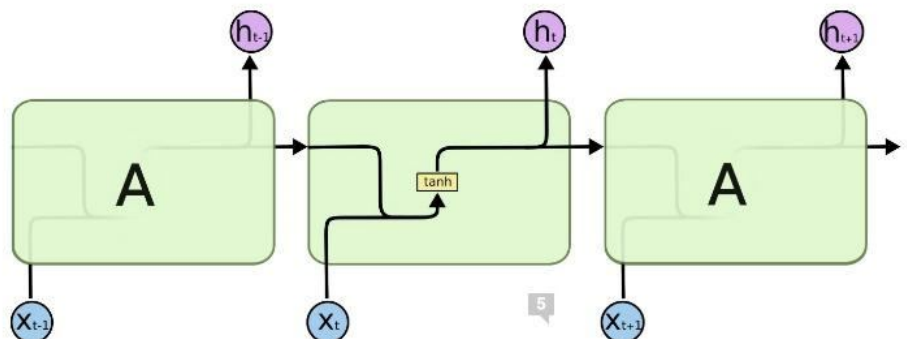


Figure 1: RNN

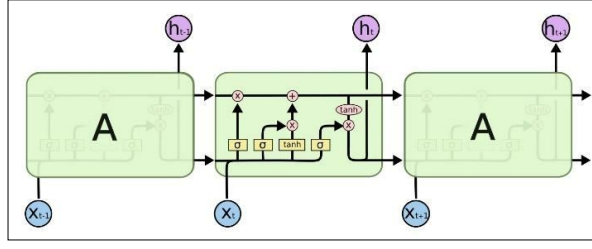


Figure 2: LSTM

B. Probabilistic Context Free Grammar (PCFG)

Probabilistic Context-Free Grammar (PCFG) can be used to model the generation of different types of soccer commentary sentences. In a PCFG for soccer commentary, you can define non-terminal symbols to represent different syntactic categories or constituents related to the commentary, such as "goal_scored," "corner_conceded," "foul_conceded," "delay," and "card." These non-terminal symbols represent different events or situations that can occur during a soccer match. Each non-terminal symbol can have associated production rules that specify how it can be expanded into a sentence. For example, for the non-terminal symbol "goal_scored," you can define production rules like "{scorer} scored a stunning goal!" or "What a goal by {scorer}!" Each production rule is associated with a probability indicating the likelihood of that rule being used. The PCFG allows you to generate soccer commentary sentences by starting with an initial non-terminal symbol, such as "goal_scored," and recursively applying production rules to expand the non-terminal symbols into a sequence of terminal symbols (actual words or phrases). The choice of production rules at each step is determined probabilistically based on their associated probabilities. For example, if the input data indicates that a goal has been scored by a player named "Thomas," the PCFG can generate a sentence like "Thomas scored a stunning goal!" or "What a goal by Thomas!" The choice between these two options is determined by the probabilities associated with the respective production rules.

By using a PCFG, you can generate a variety of soccer commentary sentences with different structures and phrases, while still maintaining the overall coherence and relevance to the events in the match. The probabilities associated with the production rules allow you to control the likelihood of different types of sentences being generated, adding diversity and naturalness to the commentary.

IV. MECHANISM

In the soccer commentary generation project, there are two key components where mathematics plays a role: the LSTM model and the PCFG.

LSTM Model

There are various equations that control how information flows across the network in the LSTM model. Here are the key equations involved:

Forget Gate: It chooses which data from the previous hidden state should be discarded. Refer (1):

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \quad (1)$$

Input Gate: It decides which new information to add in the current hidden state. The equation for the input gate is written as (2):

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \quad (2)$$

Candidate Hidden State: It computes the new candidate values to be added to the hidden state. The equation for the candidate hidden state is given as (3):

$$\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \quad (3)$$

Updated Cell State: It updates the cell state by combining information from the forget gate, input gate, and candidate hidden state. See (4):

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \quad (4)$$

Output Gate: It governs the LSTM cell's output. Refer (5):

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \quad (5)$$

Hidden State: It applies the output gate to the updated cell state to obtain the hidden state. The equation for the hidden state is (6):

$$h_t = o_t * \tanh(c_t) \quad (4)$$

These equations involve matrix multiplications (W_f , W_i , W_c , W_o) between the weight matrices and the concatenated input vectors ($[h_{t-1}, x_t]$), as well as element-wise operations (additions, multiplications) and activation functions (sigmoid and hyperbolic tangent).

Here,

$$\begin{aligned} i_t &\rightarrow \text{represents input gate.} \\ f_t &\rightarrow \text{represents forget gate.} \\ o_t &\rightarrow \text{represents output gate.} \\ \sigma &\rightarrow \text{represents sigmoid function.} \\ W_x &\rightarrow \text{weight for the respective gate(x) neurons.} \\ h_{t-1} &\rightarrow \text{output of the previous lstm block} \\ &\quad \text{(at timestamp t-1)} \\ x_t &\rightarrow \text{input at current timestamp.} \\ b_x &\rightarrow \text{biases for the respective gates} \\ \tilde{c}_t &\rightarrow \text{candidate cell state at timestamp t} \\ c_t &\rightarrow \text{cell state at timestamp t} \end{aligned}$$

B. Probabilistic Context Free Grammar (PCFG)

PCFG is a formal grammar framework that combines context-free grammar rules with probabilities. It is often utilized in natural language processing instances such as language generation.

Probabilities and production rules are used in the mathematics of PCFG. Each production rule is made up of a non-terminal symbol on the left and a series of symbols (terminal or non-terminal) on the right, along with a probability. The probabilities associated with the production rules define the likelihood of generating specific sequences of symbols. During the generation process, the PCFG selects production rules based on their probabilities to expand non-terminal symbols and generate sentences. Probabilities are frequently computed using statistical approaches such as maximum likelihood estimation or Bayesian estimation, with the frequency of occurrence of the production rules in the training data taken into account. The PCFG framework allows for the generation of sentences with varying probabilities based on the rules and their associated probabilities. By choosing production rules probabilistically, it enables the generation of diverse and contextually appropriate sentences.

In essence, the LSTM model employs mathematical equations to interpret sequential input and capture long-term dependencies, while PCFG employs probabilities and production rules to generate sentences probabilistically. These mathematical foundations form the basis for the soccer commentary generation project and enable the generation of coherent and contextually relevant commentary sentences.

IV. RESULTS AND DISCUSSION

Current results generated are satisfactory for the given hardware and dataset limitations.

PCFG model is returning events with no errors.

The LSTM model's performance heavily relies on computational resources and model complexity. With upgraded hardware, such as more powerful GPUs or TPUs, A more complex LSTM model that can capture the complex trends and connections in the football commentary data can be trained and deployed.

Despite its initial limitations, the initiative has established a solid platform for future research and development. The application of LSTM revealed the power of sequence modelling in producing cohesive and contextually appropriate football commentary. By training on a large dataset of annotated soccer commentary, the LSTM model can learn the underlying patterns and generate realistic and engaging commentary sentences.

To improve the results in future iterations, it is recommended to consider the following:

Larger and more diverse dataset: Collecting a comprehensive dataset of soccer commentary from various matches, leagues, and languages can provide the LSTM model with a broader context and enable it to generate more diverse and accurate sentences.

Fine-tuning and hyperparameter optimization: Experimenting with different LSTM architectures, layer configurations, activation functions, learning rates, and regularization techniques can lead to improved performance. Fine-tuning the model on specific subsets of data or incorporating pre-trained embeddings can also enhance the model's ability to generate high-quality commentary.

Incorporating contextual information: Augmenting the input data with additional information such as match statistics, player profiles, and team histories can enhance the model's understanding of the game context. This can enable the LSTM model to generate more contextually relevant and insightful commentary sentences.

Post-processing and refinement: Implementing post-processing techniques, such as language modeling or rule-based filtering, can help refine the generated commentary sentences. This step can ensure grammatical correctness, coherence, and adherence to broadcasting style guidelines.

User feedback and evaluation: Engaging with users, professional commentators, or soccer enthusiasts to obtain feedback on the generated commentary can provide valuable insights for model refinement. Conducting systematic evaluations, such as human evaluations or automated metrics, can also help assess the quality and effectiveness of the generated commentary.

V. CONCLUSIONS

In conclusion, the initial LSTM implementation met our expectations considering the given computational capabilities. However, upgrading the hardware and considering the aforementioned improvements can lead to more promising results in generating realistic and engaging soccer commentary. The project serves as a starting point for further advancements in soccer commentary generation and showcases the potential of using advanced techniques in natural language processing for sports-related applications

ACKNOWLEDGMENT

We would like to thank Prof. (Dr.) Priyadarshan Dhabe for his invaluable guidance to our project.

REFERENCES

- [1] Graves, A., Mohamed, A. R., & Hinton, G. "Speech recognition with deep recurrent neural networks." in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 6645-6649), 2013
- [2] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078, 2014
- [3] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. "LSTM: A search space odyssey." in IEEE Transactions on Neural Networks and Learning Systems, 28(10), 2222-2232, 2017
- [4] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. "Empirical evaluation of gated recurrent neural networks on sequence modeling". arXiv preprint arXiv:1412.3555, 2014
- [5] Bai, S., Kolter, J. Z., & Koltun, V. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling." arXiv preprint arXiv:1803.01271, 2018
- [6] Lipton, Z. C., Berkowitz, J., & Elkan, C. "A critical review of recurrent neural networks for sequence learning". arXiv preprint arXiv:1506.00019, 2015
- [7] Karpathy, A., & Li, F. "Visualizing and understanding recurrent networks." arXiv preprint arXiv:1506.02078, 2015
- [8] Gers, F. A., Schmidhuber, J., & Cummins, F. "Learning to forget: Continual prediction with LSTM." Neural computation, 12(10), 2451-2471, 2000
- [9] Jozefowicz, R., Zaremba, W., & Sutskever, I. "An empirical exploration of recurrent network architectures." in Proceedings of the 32nd International Conference on Machine Learning (ICML) (Vol. 37, pp. 2342-2350), 2015