Sequence to Sequence: Real-time commentary generation for Cricket

Gautam Phadke, Krunaal Tavkar, Manish Jangid, Sudarshan Sunder

gphadke@usc.edu (2661440757)
tavkar@usc.edu (8105863007)
mjangid@usc.edu (4487371627)
ssunder@usc.edu (4286303747)

1 Introduction

Commentary has been an integral part of Cricket for a long time. It provides an extra dimension to viewers through perspective, context, stats and sometimes even humour. People who aren't able to watch the game on TV or online often tune into text based commentary on websites like ESPN and CricBuzz [3].

However, commentary involves a lot of manual labor, especially, for generating an online text based one. The commentator will usually watch the game ball by ball, and type out a brief summary about each ball. Although this medium has been fairly successful, there have been some issues with it; Manual inspection of the game, long delays between the ball being played and commentary being generated, and sometimes a very biased view of the game. Furthermore, in some countries like India, with a plethora of regional languages, there are very few sources of commentary for lesser known languages.

There is a relatively small amount of research and project work undertaken in context of commentary generation. Most implementations fail to capture the entire context of the current ball, and others fail to generate meaningful coherent sentences [5].

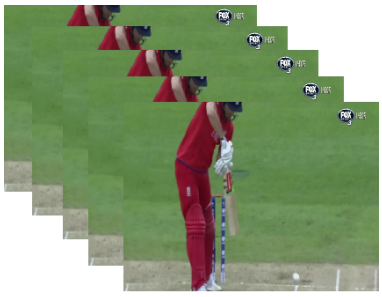
The aim of our project is to generate live commentary for cricket games through Natural Language based Deep Learning Techniques. This will remove the need for manual inspection and leads to real-time, unbiased text generation. We have designed Deep Learning models to generate commentary in both English and Hindi.

2 Method

Subsequent sections describe the methodology that includes data collection and preparation, description of an end to end Sequence to Sequence (Seq2Seq) model, and it's evaluation.

2.1 Materials

We prepared a novel data-set that comprises of Video frames extracted from ICC Champions Trophy 2013 finals between India and England, and it's corresponding ball-by-ball commentary. 247 balls (125 by India and 122 by England) were bowled in the match. We split the video into several clips, one for each ball, and then divided each clip into several frames. This resulted in a data-set of around 254,000 images (1000 image frames for every ball). We then select 40 frames for every ball, that focus explicitly on the action of bowling and hitting the ball.



1.0.1,good start to the over by the bowler. A good swinging delivery and the batsman defends the ball. No run.

1.0.1,good ball by the bowler. Just outside off-stump and the batsman just taps it down. No run.

1.0.1,a good out-swinging delivery by the bowler. The batsman plays a defensive shot. No run.

1.0.1,a good ball to start thr innings with. A hint of swing for the bowler and the batsman confidently defends the ball. No run.

1.0.1,No run fuller length ball and it veers away a touch just outside off Sharma knocks it to the off side

1.0.1,we under way with the 1st ball of the innings. A little hint of swing for the bowler, the batsman plays a good looking defensive shot.

Fig. 1. Sample Data point represents one ball. Every data point consists of 40 video frames, and 6 corresponding text commentaries.

We also scraped English commentary from websites like Cricinfo and CricBuzz [3]. To augment the data further, we manually typed out some commentary for each ball played in the game. This was done so as to increase the size of our training data. Hindi commentary was generated using the Google Translate API for every corresponding English commentary. Thus, we obtained a total of 3048 text sequences (1524 for English and 1524 for Hindi). The size of our English corpus was 24025 words, and size of our Hindi corpus was 25620 words.

2.2 Procedure

Concretely, given a sequence of video frames X $(x_1, x_2, ...x_n)$, we aim to generate a sequence of text Y $(y_1, y_2, ...y_m)$ that "best summarizes" the given video frames. This would be further elaborated in upcoming sections. Our procedure consists of two steps: Encoding the image features by generation of Object Masks, and Generation of text commentary using Sequence to Sequence (Seq2Seq) LSTM (Long-Short term Memory Network) modules. Our end to end model is diagrammatically represented in Fig. 2.

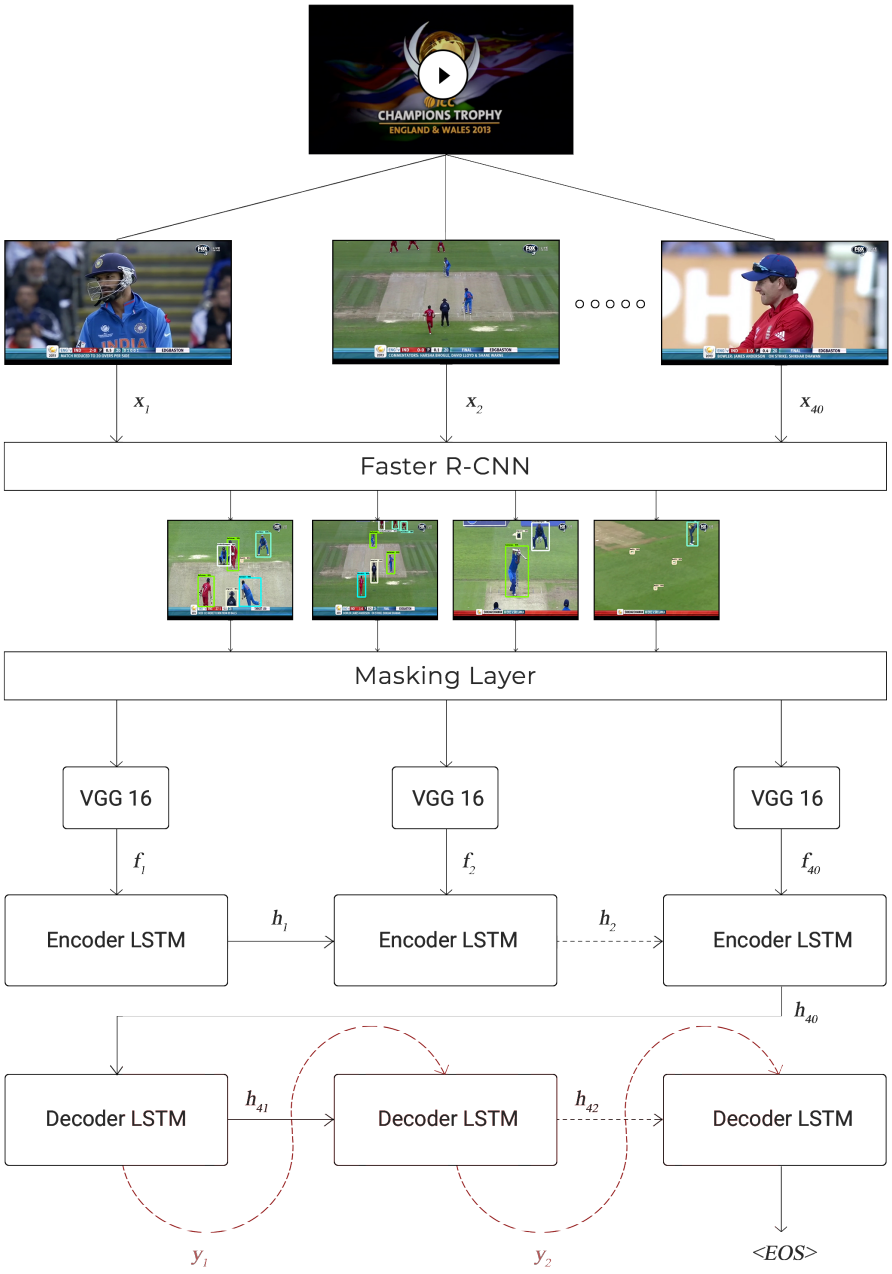


Fig. 2. End to Eng model for text commentary generation

2.2.1 Generation of object masks using Faster R-CNN

We manually tagged each frame using Labellmg [1], a graphical annotation tool built on Python. From each frame, we identified 6 different objects; Batsman, Bowler, Fielder, Wicket-keeper, Umpire, and the Ball itself. We manually marked bounding boxes for each of these objects. We then used Faster R-CNN [4] algorithm to perform object detection on these frames, so as to capture spatial information about important features. Objects obtained from Faster R-CNN were used to generate image masks [7], and these image masks were encoded using VGG16 [6] so as to obtain localized features.

Thus, for a given video frame at time t, i.e. (x_t), we extracted features from the last layer of fully trained Faster R-CNN. These features were concatenated with localized features obtained from last layer of VGG16. For every video frame X $(x_1, x_2, ...x_n)$, the features F $(f_1, f_2, ...f_n)$ were extracted. The size of resultant features for every ball was 400 X 8192. These features were stored in a numpy file, and later used as inputs to the encoder LSTM module.

2.2.2 Generation of Commentary using Seq2Seq LSTM model

The Seq2Seq model consists of two stages; Encoding and Decoding. During encoding stage, given the input feature sequence F $(f_1, f_2, ...f_n)$ obtained from Object Detection Module, encoder LSTM computes a sequence of Hidden states $(h_1, h_2, ...h_n)$. Details regarding this computation is mentioned in [2].

The decoding stage always begins with a Beginning Of Sentence $< BOS >$ tag, and ends with an End OF Sentence $< EOS >$ tag. All words in the corpus are mapped to their respective ID's using a Label Encoder. During decoding stage, a distribution over the text sequence Y $(y_1, y_2, ...y_m)$ given the input feature sequence F $(f_1, f_2, ...f_n)$ is defined as:

$$\begin{aligned} p(Y|F) &= p(y_1, y_2, ...y_m|f_1, f_2, ...f_n) \\ &= p(y_1|h_n, y_0) \times p(y_2|h_{n+1}, y_1) \times ... \times p(y_m|h_{n+m-1}, y_{m-1}) \\ &= \prod_{t=1}^{m} p(y_t|h_{n+t-1}, y_{t-1}) \end{aligned}$$

As seen from the above equation, a word generated at time t (y_t) is conditioned only on the previous word (y_{t-1}) and previous hidden state (h_{n+t-1}).

The objective function maximizes the log probability of this distribution w.r.t parameters of the LSTM, using stochastic gradient descent. Thus, the parameters

generated after end-to-end training of encoder-decoder LSTM are given by:

$$\theta^* = argmax_\theta \sum_{t=1}^{m} log(p(y_t|h_{n+t-1}, y_{t-1}; \theta)$$

where θ^* represents optimal parameters.

2.2.3 Evaluation

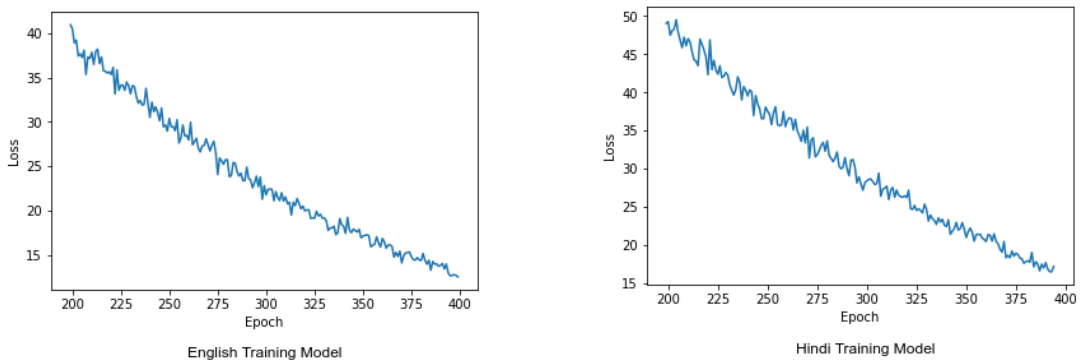


Fig. 3. Learning Curve

We use the soft-max cross-entropy loss function during our training phase. During the inference phase, once the parameters are learned, output of decoder LSTM at a given time t (z_t) is used to obtain the emitted word ($y_t^{'}$). We apply softmax function to obtain the probability distribution over words $y^{'}$ in vocabulary V. This probability distribution is given by:

$$p(y_t^{'}|z_t) = \frac{exp(W_y \times z_t)}{\sum_{y' \in V} exp(W_{y' \times z_t})}$$

where W_y is the parameter for decoder LSTM. Fig. 3. represents the learning curves for both the models. We observed that the both the models overfit after 400 epochs. Thus, we trained both the models only till 400 epochs each, using a batch size of 10.

3 Results

Cricket Commentary generation using Video sequence is a comparatively novel and unique research area. Also, our data-set being novel, there exists no baseline

for comparison. Hence, we focus on the qualitative analysis of our results, instead of a quantitative one. The essence of our exploratory qualitative analysis is to identify if our text generation model was able to capture any context from the video sequence features given to it for testing. We measure the quality of our results by manually comparing the generated text with events and components present in the given video sequence. Fig. 4. represents positive results for English and Hindi commentary, meaning that the context of video has been accurately identified by our model, and the generated commentary is comparable to human standards.

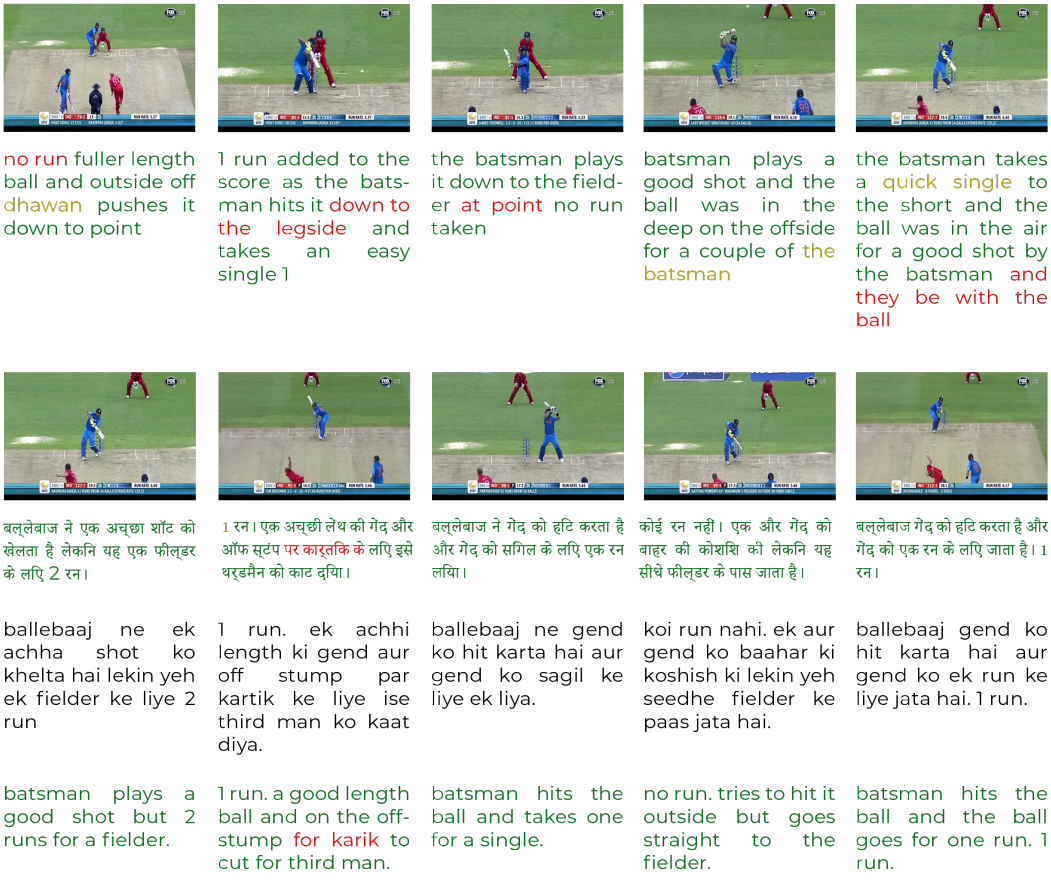


Fig. 4. Positive/Good commentary for English and Hindi.

Green color represents part of the caption where the context has been accurately identified, and human-like commentary has been generated. Yellow repre-

sents partially correct text generation, whereas Red represents part of the caption where context of the game has been wrongly identified.

Fig. 5. represents negative results for English and Hindi commentary, meaning that our model has not been able to identify the context of video sequence. Thus, it generates the sequence of text that has the maximum probability of occurrence in the training data-set. It essentially means that in this case, our model suffers from the problem of local minima.

Fig. 5. Negative/Bad commentary fro English and Hindi.

Both Fig. 4. and Fig. 5. show 10 examples each. Fig. 4. shows 5 good examples of English and 5 good examples of Hindi. Fig. 5. shows 5 bad examples of English and 5 bad examples of Hindi.

4 Discussion and Future Work

We notice that our model is successful in generating meaningful commentary for events that occur often in the dataset (For example, the event of a batsman defending the ball). However, for events that don't as often (For example, Batsman being stumped by the wicket-keeper), the model is not able to generalize well. In such cases, the commentary generated does not capture entire context of the event. This can be overcome by providing the model with more training data, spanning several matches, to ensure that it sees more type of events. Moreover, our model is able to identify the batsman or bowler in the frame, but not who he or she is. If we were to train our model with enough examples of various players, we would be able to identify them in the frame and use that information in generation of commentary.

Another possible addition to our model could be the use of a rule-based template. In some cases where the model isn't able to generate meaningful commentary, we can instead use generic templates like:

"_____ to _____. _____ plays the ball on the _____ for a single".

Our model would then detect the events that occurred in the given ball and the players involved, and uses the above template to provide commentary. For example:

"Watson to Dhoni. Dhoni plays the ball on the offside for a single."

Currently, our model produces commentary that just describes the ball being played, in a way similar to video captioning. In actual commentary however, there is an element of passion and excitement that appeals to the viewers who are either listening or reading. For example, instead of generating:

"Watson to Dhoni. Dhoni hits the ball over mid-wicket for a six."

we could potentially extend our model to generate a more realistic commentary like:

"Watson to Dhoni. What a great shot by Dhoni!! The ball has gone out of the park for a Six!"

References

1. (January 13, 2019), https://github.com/tzutalin/labelImg
2. Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., Schmidhuber, J.: Lstm: A search space odyssey. IEEE transactions on neural networks and learning systems **28**(10), 2222–2232 (2016)
3. GROUP, S., et al.: Cricbuzz Cricket Scores & News (2015)
4. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)

5. Sharma, A., Arora, J., Khan, P., Satapathy, S., Agarwal, S., Sengupta, S., Mridha, S., Ganguly, N.: Commbox: Utilizing sensors for real-time cricket shot identification and commentary generation. In: 2017 9th International Conference on Communication Systems and Networks (COMSNETS). pp. 427–428. IEEE (2017)
6. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
7. Yang, Z., Zhang, Y.J., ur Rehman, S., Huang, Y.: Image captioning with object detection and localization. In: International Conference on Image and Graphics. pp. 109–118. Springer (2017)