CricSum: Cricket News Generation from Live Text Commentary using Abstractive Text Summarization Technique

MSc Research Project
Data Analytics

Sachin Muttappanavar
Student ID: 20144253

School of Computing
National College of Ireland

Supervisor: Dr. Rejwanul Haque

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Sachin Muttappanavar
Student ID:	20144253
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Dr. Rejwanul Haque
Submission Due Date:	31/01/2022
Project Title:	CricSum: Cricket News Generation from Live Text Commentary using Abstractive Text Summarization Technique
Word Count:	7500
Page Count:	20

Signature:	Sachin Muttappanavar
Date:	30th January 2022

CricSum: Cricket News Generation from Live Text Commentary using Abstractive Text Summarization Technique

Sachin Muttappanavar
20144253

Abstract

Cricket is one of the most played and popular games in the world. There are roughly 1 billion cricket enthusiasts worldwide. Most people rely on live text commentaries rather than watching live video streaming of the Cricket match. This resulted in the emergence of online websites providing live text commentaries. As a result, a massive amount of live text data relating to cricket was generated. There isn't a lot of work done on summarising a cricket game using this live text of Cricket game. Therefore, we implemented a system to generate news from the live text commentaries of Cricket games. Data is collected from the ESPNCricinfo website. Stakeholders may use our proposed model for the automatic creation of news articles, which eliminates human effort and time spent drafting the report. We have trained different models over commentary data and the best model is selected based on the ROUGE score.

1 Introduction

Robotic journalism strives to generate automatic news from the existing corpus as a completed output or as a template for the following post-editing. Robotic journalism, in particular, is concentrating on sectors such as sports, finance, and comparable statistics-based reporting. The automatic sports news generation has gained lots of attention in recent days. Many sports games like cricket, football, hockey, and others are being played every day. Manually writing the news article related to each game is tedious and time-consuming. Also, no research is carried out to summarize cricket games. Therefore, we are implementing a system that can generate automatic news from the live text commentaries of the Cricket games. Cricket is the most famous sport in the world. Due to its popularity, several websites emerged where one can find live text commentary as well as match-related news articles. The emergence of live text commentary resulted in the accumulation huge corpus of cricket commentaries. For a particular match, manually written news articles and live text commentary data consist of the same wordings. Therefore we are utilizing live commentaries data of the match to generate the news. We are using the ESPNCricinfo website data for this work[1]. Implementing a system to generate cricket news benefits stakeholders by reducing labor work and time. To implement our methods, Natural Language Processing (NLP) text summarization techniques are employed. The

[1]ESPNCricinfo: `https://www.espncricinfo.com/ci/content/site/company/terms_use.html`

goal of text summarization is to compress long source documents into shorter ones while keeping the important points. The cricket game summarization task is more challenging than traditional text summarization because there are distinct writing styles between the news and commentaries. Particularly, commentary texts are more informal than the news.

Due to its practical importance, sports game summarization has steadily gained the attention of scholars in recent days. Zhang et al. (2016) worked on constructing the news from live commentaries of a football game. He created the dataset consisting of 150 samples for training the model. Later there were many improvements in the model built by Huang et al. (2020) and Wang et al. (2021) contributed to the previously available dataset. Also, there are other works on sport summarization in various sports like Hockey, football, and many others using live commentaries. There haven't been many efforts done on cricket match summaries or news production from live text commentary. There is no off-the-shelf dataset available for analysis of Cricket game summarization.Therefore, Data collection was a crucial task in this work. We have built a separate scraper to crawl the data from the ESPNCricinfo website and created CricSum dataset. For this project, around 800 match instances are being gathered. We have collected all forms of Cricket match data. The data contains a lot of noise. As a result, extensive data cleaning activity is performed. Data processing plays a vital role in this research work as it needs domain knowledge, insight into the cricket game. In this study, abstractive text summarization approach is explored. Pre-trained abstractive summarization models like T5, BART are fine tuned with our custom dataset. These sequence2sequence models tend to produce high-frequency names in the summary than the actual right names. To fix this issue we have masked the player names in the data. We have trained model in template2template way. ROUGE score is used for an evaluation of model outputs. There are many other areas where summarization of documents containing named entities and written in different styles is required. One such application is event summarization using tweets.

2 Related Work

Text summarization in NLP involves reducing long documents into concise summaries. It is one of the application of the NLP. As manual text summarization work incurs lots of time and cost, automatic summarization of the text has drawn a lot of attention and therefore forms a substantial motivation for academic research. Various researches have been done on text summarization using NLP. Text summarization finds its application in areas like Newsletter, Media monitoring, Financial research books, sports, literature, automated content creation, Email threads and many other. As part of the literature Survey, we have reviewed researches that gave state-of-art results on extractive and abstractive text summarization, Sports news generation.

2.1 Neural Abstractive Summarization:

The popularity of the neural abstractive summarization model has grown. They are capable of producing coherent and flexible summaries. The neural abstractive model feeds documents x containing n sentences to an encoder layer, which generates representations, and then to a decoder, which generates a summary y one word at a time. Much research is carried out to improvise model architecture. For instance, models that allow directly copying words from input to output and models with coverage attention layer to discourage the selection of the repetitive words by the model (Gu et al. (2016), See et al. (2017)). Also, there are new approaches are being followed in the summarization of text. One such is the usage of guidance signals along with input documents to make summaries more correct. For example,Gehrmann et al. (2018b)implemented neural attention model which takes the set of keywords along with input documents in the generation of abstractive summary(Li et al. (2020)). Jin, T. Wang, and Wan 2020 extract important relations like subject, relation, and object from source document and depict them in a graph neural network. They demonstrated that fluent and faithful summaries can be generated by decoding the extracted relations. Similar work is conducted by Saito et al. (2020), keywords and highlighted sentences are pulled from the source document and models are trained on these extracted sentences. Dou et al. (2021) built a GSum framework for the summarization task. This framework can take different external guidance signals as input along with the document. Guidance signals can be highlighted sentences, keywords, relations, and retrieved summaries. For extracting the guidance signal from the documents both manual and automatic systems are defined. In the manual process, the user finds the important sentences from the document and prepares the guidance signal. In the case of an automatic system, an oracle extractor is used to create a guidance signal. Both x and y are used to deduce the guidance signal in automatic prediction. They implemented various models such as BertExt, BertAbs, MatchSum, BART. The experiment is carried out on 6 datasets like Reddit, XSum, CNN/DM, WikiHow, New York Times, PubMed. The results of the experiment show that their model is effective and performed well on the four prominent summarization datasets when highlighted sentences were used as guidance.Overall, this research has implemented a remarkable method to generate fluent summaries, and the interpretation of the results was good.The standard approach in abstractive summarization involves supervised learning of models with document-summary pairs which are costly to acquire. To address this problem unsupervised learning model is built by Chu and Liu (2019) for summarization of the multiple documents(Liu and Lapata (2019)). MeanSum model can produce good summaries with the ROUGE score of 47.90. A drawback of this model is, it lacks attention or pointers. Their model does not give a solution for an unsupervised way of summarizing the single documents. Attentional encoder-decoder RNN models proposed by do well in summarizing the text document without additional tuning. The model was trained on the Gigaword and DUC datasets. Model generated summaries are not so coinciding with gold summaries but summaries are readable and fluent. As part of future work, they intend to research methods for successfully producing unique words in the summary. Marek et al. (2021) research work studies the summarization of the Czech news reports and the effect of named entities for this task. SumeCzech dataset was used for this task. Dataset consists of around one million news articles collected from the Czech news website. Named entities present in the dataset are annotated using Spacy's NER. They first implemented extractive summarization using the baselines Random, TextRank algorithm to select the single sentence headlines of the news articles. They proposed a Named Entity Density extractive model to select the headline sentence. Model select the sentence based on the highest ratio of the count of entities over the length of the sentence. The investigation shows that the proposed model produced good results as a baseline in selecting the headline sentences for the news article. To generate the novel summaries an abstractive summarization sequence-to-sequence with NER model was developed. The proposed model produced state-of-the-art results. They created ROGUE-NE, a novel evaluation metric that quantifies the overlap of named items between the actual and produced summaries. Evaluation results show that the model performs better than the state-of-the-art methods that struggle with the named entities in summarization.

2.2 Abstractive summarization with transformed based model

Various researchers worked on the abstractive summarization of documents. Zolotareva et al. (2020) explored Transfer Learning with unified Text-to-Text and recurrent neural networks. Sequence-to-Squence model is built with TensorFlow, Keras, machine learning and neural network libraries. This model is configured following values for hyperparameters Epochs = 25, Optimizer = 'rmsprops', batchSize = 64, latent dimensions = 256, Embedding dimension = 200, Loss function = "sparsecategoricalcrossentropy" . These parameters are selected through a manual process. The accuracy and loss values of the model are analyzed to check the performance of the model. The model was trained and tested over the BBC news dataset sourced from Kaggle. Similarly, pre-trained model T5 is fine tunned with BBC news custom data. 80 percent of data is used as a training set and the rest as a test set. Results of this model are found and analyzed. They showed that the T5 base model performed well on BBC news data compared to the recurrent neural network model. As future work, they suggested studying the transformer-based models for document summarization tasks. In this the research methods and model results are excellent. Gehrmann et al. (2018a)Neural Abstractive summarization models are good at generating fluent summaries but fail to select the contents from the document. In this research paper, this problem is addressed by applying the bottom-up attention step as a selector. The bottom-up attention step restrains the model to select the likely phrases from the document. The model is trained over CNN-DM and NYT corpora. The system combined with the bottom-up attention layer has shown significant improvement in ROUGE value. Bajaj et al. (2021) researched on summarizing the long documents using pre-trained language models. They studied the low resource setting of summarization of long source documents. They curated a dataset of 120 instances consisting of the source document and summary from Amicus Briefs. Salient features from the document are extracted using GPT-2 language model perplexity scores. Compressed documents are fed to the BART model. ROUGE evaluation is used to evaluate the BART model. BART model produced a good ROUGE value and the summary of the model was coherent as well as fluent. Improvising the performance of the existing summarization techniques is a challenging task. The research was done in this direction to improve the existing techniques and one such work is by . They have used CNN/DailyMail and NewsRooms datasets. Their proposed framework consists of the transformer to encode the source document and then the sequence-to-sequence model. It is discovered that the transformer and seq2seq models work well together, resulting in a more detailed encoded vector representation. This research outcome conveys that paying attention to the wording of target words during abstraction improves performance. The proposed hypothesis and framework experiment on the extractive and abstractive summarization task and CNN/DailyMail datasets are used for evaluation. The results of the models are outperform existing state-of-art models.The novel data-driven framework is developed for abstractive summarization using semantic representation. It is an Abstractive Meaning Representation driven statistical abstractive summarization framework. Here, MAR graphs are constructed by parsing the source text, and then graphs are transformed into a summary graph. Final summaries are generated from this summary graph. The graph-to-graph transformation, which translates source semantics into a summary graph, is their fundamental goal. The research approach was promising and experiment results are favorable.Email thread summarization is a challenging task. EmailSum is a framework developed by Zhang et al. (2021)which implements email thread summarization tasks. Human annotated short and long summaries of 2549 email threads data related to different topics are used for this work. In this study different summarization techniques like single-document and hierarchical models, transfer learning, semi-supervised learning, extractive, and abstractive methods are explored. Manual evaluation of the summaries is done to evaluate the generated summaries. The challenging part of this study was understanding the sender's intent and determining the roles of sender and receiver. They discovered that treating email chain data as one document and fine-tuning T5establishes a reasonable baseline. They emphasised more on the manual evaluation of the model outputs as automatic evaluation does not ensure the correctness of summaries.

2.3 Sports new generation using natural Language Processing:

Constructing the news from the live text commentaries: In recent days, sports game summarization has drawn the attention of many researchers. Much research is being carried out to automatically generate the sports from the available sports-related data. Kanerva et al. (2019) built a system to generate the news of Finnish ice hockey. The system is trained on the news articles data collected from the Finnish News Agency STT. They manually prepared the data to get it to a state appropriate for training the end-to-end systems. Pointer generation network model having encoder-decoder with attention mechanism was used for model a probability distribution of the words over the known vocabulary. A distinct coverage attention vector is kept to tell the model about its previous attention decisions. Thus model prevents the repetition of texts in generated results. Their proposed model performed well with a BLEU score of 19.67 on test data. But their summary contains the high-frequency player names instead of the actual player name. Also, factual content was missing in the summary text. Efforts are made towards generating the news from the live text commentaries of the football game. Zhang et al. (2016) built a new dataset for their research. They have crawled live text commentaries of football games and manually written news articles from both Sina Sports and 163 football. They created a dataset consisting of 150 matches live text. Learning to rank model was developed with novel task-specific features. A probabilistic sentence selection method was implemented to prevent the local redundancy problem. Their experimental result shows that their proposed model was appropriate and the task is achievable. They stated that as live commentary and sports news are written in different styles, some post-editing rewritings will make the summary more natural. Also, they suggested concentrating on collecting datasets on a larger scale. Wang et al. (2021) extended this research by contributing the data as well as different abstractive summarization models. They addressed noise issues associated with the dataset and the neglect of lexical overlap between news and commentaries results in the low-quality pseudo-labeling algorithm. They proposed a two-step model consisting of Selector and Rewriter. Selector picks important sentences from live text commentaries and then seq2seq models are trained over it. Their system can summarise football games more appropriately. Gunasiri and Jayaratne (2019) worked on natural language generation(NLG) by using the scorecard as an input to the system. They tried to automate the Cricket game news generation process by using the NLP. Their system can be utilized by the journalist to automatically generate news thereby reducing time and cost incurred in the manual effort. A template-based system is developed to generate news from the structured text data in the Sri Lankan Style. They studied the usefulness of a template-based system for cricket sports news generation. They mentioned that their system is useful particularly in a language other than English as there are not many libraries, packages are available to build systems. Their proposed system typically consists of two levels of templates namely sentence template and phrase template. It was challenging to evaluate the NLG system because of the lack of evaluation metrics. They have used three evaluation scores which are calculated to compare the actual output of the system with the reference text from the two websites. Their system produced a good score in terms of similarity score, degree of closeness, and data count. But fluency of the summary was not good. Model was not able to follow the pattern in the news.

2.4 Summary:

A number of concerns and issues have been found as a result of the relevant literature study. Fewer studies have been conducted in the sports summarization task, and the majority of those that have been conducted are related to football and hockey games. To the best of my knowledge, a little study is done on summarising a cricket game utilizing live commentary and generating news. There are no off-the-shelf datasets available for generating the news from the Cricket game. Also, from the literature review, it was discovered that fewer studies linked to the use of pre-trained models in the sports news production task.

3 Methodology

This research project employs a text summarization method to summarise a cricket game leveraging live text commentary with great fluency and readability. Methodology followed in this research study is shown in Figure 1. This research methodology ensures that the investigation is conducted in a methodical manner.It consists of seven phases: Business Understanding, Data Collection, Data Understanding, Data Processing, Data Transformation, Modeling, Evaluation. The project cycle is described in these phases.

3.1 Business Understanding

At present, there is no existing system to automatically generate the news from the live text commentaries of the Cricket game. Expertise is necessary to hand compose the news article for the Cricket game, which is costly in terms of both money and time. To address this issue, we are implementing a system that automatically summarises the Cricket game post-completion. In terms of generating news stories, this approach is less labor-intensive, cost-effective, and time-consuming. It may be used by stakeholders in the development of cricket news. We have created a new dataset for this research work. To create the system, several texts summarising approaches such as extractive techniques, abstractive techniques, and pre-trained models are investigated and applied.

3.2 Data Collection

There are no off-the-shelf datasets for evaluating sports news creation, as far as we know. As a result, for this study, a fresh dataset must be created from scratch. Meanwhile, written commentary of cricket games is immensely popular all around the world. Most cricket fans do not have access to live video streaming of matches, thus they must rely on written commentary provided by online services. There are several internet sources that offer ball-to-ball live text commentary of Cricket matches. ESPNcricinfo is a popular website that offers live text commentary as well as news articles related to the Cricket game. Data for all cricket formats, including Twenty20 international, International one-day innings, and Test matches, is available on their website. Everyone has free access to this information. This information is being gathered for our research purposes. We have Cricket data of all the formats. We were required to collect both live text commentary and news articles from the same Cricket match for evaluation purposes. For each Cricket match, there exists a documented live text commentary after the matches and a manually written news article on the ESPNCricinfo website. News articles are written by professional editors manually and hence they are suitable for our evaluation of the summaries. We have scraped the 782 match instances data on the ESPNCricinfo. To scrape the data we have implemented a Java-based automation tool. Scraped data is parsed using a parser(Jsoup- Java library) to form the structured text file. Data includes information such as over details, run scored for each ball, player details, and text commentary for each ball. Sample cricket game report data is illustrated in the

3.3 Data Understanding

In this phase, we studied the scraped data. Data consists of live text commentaries and news articles of the same game. The collection contains all forms of Cricket game data. Test matches are often played over three to five days. They are made up of several innings. As a result, the text data for test matches is slightly lengthier. One-day internationals (ODIs) are played in a single day. When compared to Test matches, the size of ODI matches is tiny. Twenty20 matches are 20-over contests that often collect fewer text data than other types of matches. For our research study, we assume that all the formats of Cricket games follow the same writing style for news articles. Data is collected in the text files. The text consists of lots of noise. Some irrelevant data is captured in the data such as tweets, player's interviews, links, advertisements. These all data need to be erased from the data. We have conducted extensive statistical analyses of the data. Some cricket matches do not have commentary, indicating that the match was not played for various reasons. Excluding such instances from the data forms a dataset of size (489,2). Each Cricket match comprises an average of 143 sentences. The average number of words present in the commentaries is around 6599. Where as gold standard news articles contain an average of 5 sentences and around 232 words are present in each report. Dataset also consists of information about the player of the match and toss results. Data needs to be processed and transformed for training the models.

3.4 Data Pre-Processsing

This step focuses primarily on pre-processing to eliminate superfluous data from the given dataset, allowing the model training process to go more efficiently. Data consists of a lot of noise in it. Such noise will result in the prediction of inappropriate outputs by the model. Such data needs to be processed from the data. Dataset consists the online advertisements in the form of hyperlinks. These hyperlinks should be filtered out from the dataset. It is observed that some of the data is either commentary or news articles for some match instances. We have to delete such match instances from our dataset. Some of the commentary phrases for certain match instances are too tiny, and it is apparent that those words will not contribute to news production. So short sentences are should be taken out from the dataset.

3.5 Data Transformation

Data transformation involves converting data from one format into another format. As our data is text data, it needs to be transformed into vectors so that machine learning/ deep learning models can understand. Word embedding of the text should be done to understand the semantic, syntactic context of a word and also to understand the closeness between the words. There are various libraries available for tokenization of the sentences like NLTK, genism, Keras. We have to make use of these packages for tokenization of the text. For pre-trained models, respective tokenizers should be used to split the sentences into small chunks of words. This helps the model in understanding the meaning or context of the word in sequence.

3.6 Modelling

In this section, we have outlined architectures used for the BART, T5. The aforementioned models were employed to accomplish the work of summarising a Cricket game.

3.6.1 Pre-trained model - T5

In transfer learning the model is first trained on the data-rich tasks and knowledge gained in solving this task is stored. Model apply this gained knowledge on different but similar problems. These pre-trained models can also be fine-tuned with custom datasets for the downstream task. T5 is one of the pre-trained models developed by google. It is a unified framework that converts text-based language problems into a text-to-text format. The model was trained on "Colossal Clean Crawled Corpus"(C4), and they were able to achieve state-of-the-art results on different tasks like summarization, text classification, question answering, and more. To aid future work the T5 model implementation is publicly available along with the C4 dataset.

3.6.2 Pre-trained model - BART

BART is a pre-trained denoising autoencoder for sequence-to-sequence models. It is trained on Corrupted text with a random noising function and a learning model to regenerate the original texts. It is a denoising autoencoder that means optimizes the reconstruction loss between the source document and the decoder's output. Transformer-based neural machine translation architecture is used by the BART. This BART model is pre-trained for various language problems. It can also be fine-tuned for downstream tasks like Sequence Classification, Token Classification tasks, Sequence Generation Tasks, Machine Translation. The model has shown state-of-the-art results in various text generation tasks.

3.7 Evaluation

In this step, we evaluate the degree to which the model meets our research objectives and try to identify if there is any reason for the model deficiency. As our research task is related to text summarization we will be using Recall-Oriented Understudy for Gisting Evaluation(ROUGE) metrics to assess the model performance. ROUGE is a software package(Lin (2004)) available for evaluating automatic text summarization. ROUGE is more interpretable than any other text summary evaluation metrics. ROUGE score basically a score of overlapping words. ROUGE score interpretation is as below:

- ROUGE n - recall: It tells about the what percent of the n grams present in the reference summary are also present in the generated summary. Recall calculation formula is shown in the Figure 2
- ROUGE n - precision: This score explains the what percent of n-grams in the generated summary are also present in the reference summary. Precision calculation formula is shown in the Figure 3
- ROUGE l - It denotes the number of longest sequences common between the produced summary and the reference summary.
  where, n is number of grams

After assessing the models with respect to research success standards, implemented model that meets the expected results become a final model.

4 Design Specification

This section explains about the implementation approach followed in this research. This research study is to generate news from the Live commentary texts of Cricket games. The goal of this project is to create a system that provides an abstractive summary of a cricket game using its text commentary. For generating the summary, news articles pertaining to the given match are used as a reference. To implement the system we proposed a method. The proposed method flowchart is shown in the Figure 4. We ended up with this framework through multiple experiments and observations. We first trained sequence2sequence models such as BART, T5 with live text commentary. It is observed that the transformer models are following the pattern in the reference but tend to generate the news with high-frequency names instead of the right player names even though player names are not present in the text commentaries. To overcome this name mismatch issue, we train the transformer sequence2sequence model in a template2template way. In template2template appraoch, we mask the player names with a special kind of token 'Player#' both in the input and target summary. To achieve masking of the player names we have implemented the script using the Standford NER tool. The Figure 7 depicts masked sample commentary. After replacing the player names with special tokens sentences look like a template. Post converting commentaries into the template, we train the transformer sequence2sequence model over templates. Through this approach, the model concentrates more on the relations among the players and their actions and is least impacted by the high-frequency names. Summary produced by the transformer models contains special tokens Player. To generate news, the report editor might manually replace these tokens.

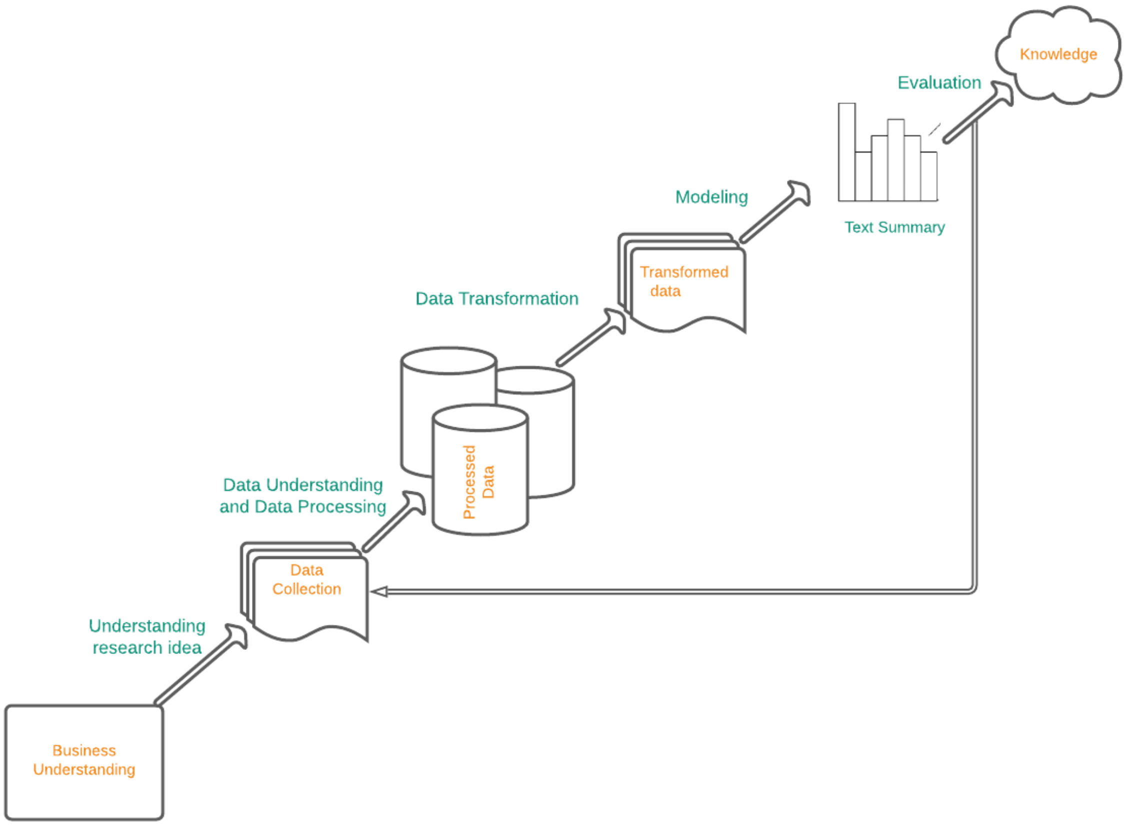


Figure 1: Methodology

$$\frac{number\_of\_overlapping\_words}{total\_words\_in\_reference\_summary}$$

Figure 2: ROUGE Recall Formula

$$\frac{number\_of\_overlapping\_words}{total\_words\_in\_system\_summary}$$

Figure 3: ROUGE Precision formula

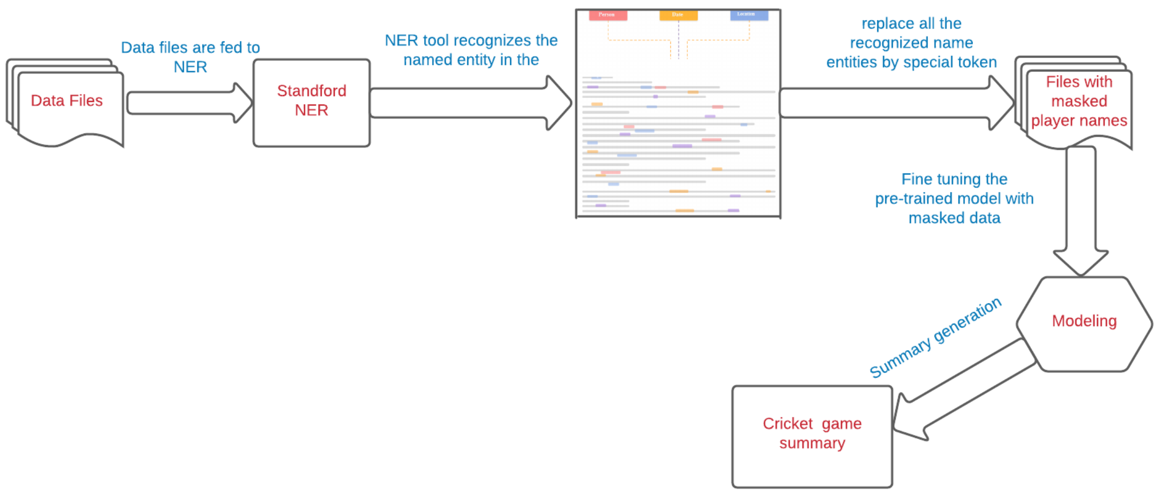

Data Files
Data files are fed to NER
Standford NER
NER tool recognizes the named entity in the
replace all the recognized name entities by special token
Files with masked player names
Fine tuning the pre-trained model with masked data
Modeling
Summary generation
Cricket game summary

Figure 4: Flow chart of proposed method

5 Implementation

This section describes the implementation of the different text summarization models and scraper for data collection. As mentioned in section 3, the data was not readily available for the research study. We have gathered data with help of a Java-based selenium automation script. For summarization tasks, TextRank, BART, T5 models are implemented. A detailed explanation of the implementation of scraper and models are explained below.

5.1 Implementation of Scraper

We have collected data from the ESPNCricinfo web application. As we planned to collect data from the web application, we have chosen the Selenium automation tool for writing the automation script to fetch data using Java. Selenium is the best open source automation tool available for the web browser application, it is easy to implement and use. To gather the web pages of the commentary and report sections of the website, two automation scripts are built. Like this, every match is visited on the website and web pages of commentary and news articles are scraped. All the collected HTML files need to be parsed. For parsing the web pages we have used the Jsoup, a java library for parsing, extracting, and manipulating data available in the HTML files. HTML files contain many irrelevant data along with text commentary and the news article. We managed to extract the required data by writing the complex xpaths. Extracted data contains information like current over, ball status, Player details, text commentary about the respective ball, player of the match, Teams name, score of each team, the status of the match. All the extracted data are saved into text files. Separate files are created for live text commentary and for corresponding news articles. Like this, all the collected HTML files are processed and required data are collected in the text files. Around 782 match instance data is collected. Sample data collected is shown in the Figure 6. Flow chart of the Scraper implementation is shown in the Figure 5

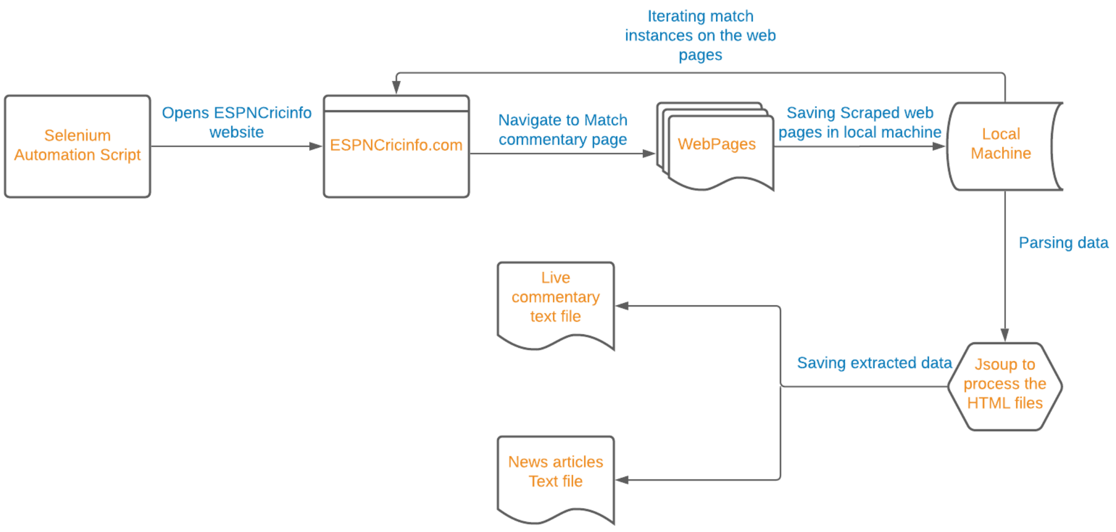
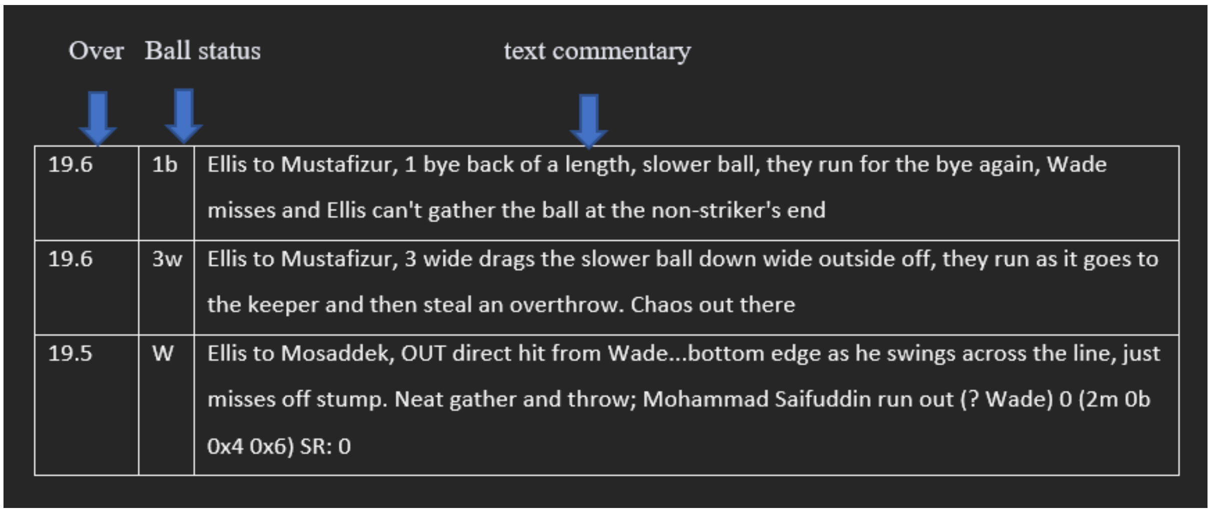

Iterating match instances on the web pages
Selenium Automation Script
Opens ESPNCricinfo website
ESPNCricinfo.com
Navigate to Match commentary page
WebPages
Saving Scraped web pages in local machine
Local Machine
Parsing data
Live commentary text file
Saving extracted data
Jsoup to process the HTML files
News articles Text file

Figure 5: Flowchart of Scraper implementation


Over
Ball status
text commentary

19.6	1b	Ellis to Mustafizur, 1 bye back of a length, slower ball, they run for the bye again, Wade misses and Ellis can't gather the ball at the non-striker's end
19.6	3w	Ellis to Mustafizur, 3 wide drags the slower ball down wide outside off, they run as it goes to the keeper and then steal an overthrow. Chaos out there
19.5	W	Ellis to Mosaddek, OUT direct hit from Wade...bottom edge as he swings across the line, just misses off stump. Neat gather and throw; Mohammad Saifuddin run out (? Wade) 0 (2m 0b 0x4 0x6) SR: 0

Figure 6: Text commentary

5.2 Data Aggregation

We have used python language with latest version for code implementation. For data aggregation and processing we have leveraged the pandas library of python. For model training we have utilized the "google colab" a google cloud platform. They provide the free service for limited hours on daily basis. Because of this time constraint building the model took more time.

5.3 Dataset Description

As explained in section 5.1, data is scraped from the ESPNCricinfo website. Dataset consists of two columns and 465 rows. Description about is illustrated in Table 1. input_text contains live commentaries of Cricket matches and the target_text column contains news articles related to the match.

Table 1: Data Description

Column names	input_text,target_text
Number of columns	2
Number of rows	465

5.4 Dataset Processing

As mentioned in the section 5.1 dataset is store in the text file. Text files are imported into pandas dataframe. The dataset contains a lot of noise and was very unstructured. We've done a huge amount of data processing work. Follow below for data-processing steps:

- Some of the matches are not played due to rain or other reasons does not contain any commentary. Such a match does not contain commentary data though news article exists. This data is no longer useful for training the model and has been deleted from the dataset.
- Data includes advertisements for social media platforms such as Facebook, Messenger, Pinterest, and Twitter. These text records should be removed from the dataset.
- Live commentary text dataset consists of the tweets related to the event that happened in the game. Such text data is kept out of the dataset because our study focuses on producing news from live text commentary.
- Along with text commentaries, players' interviews text is also present in the data. They are taken out from the data as the study concentrates only on commentary texts.
- Some of the matches commentaries are too small(<15 characters). In the news generation, this text will not communicate any information. They've been removed from the data.

- All the commentary text was in decreasing order of the over. We should feed the model with increasing order of the commentary texts. Due to this, we transformed them into increasing order using pandas inverse function. Reverting the document is handled carefully so that there should not be mismatch in the first innings data and second innings data.
- News articles collected are too big. As per our research, we are trying to predict brief news articles. Due to this, it is required to filter the news articles. We manually observed the 10 news articles and tried to understand the pattern. It is understood that the first 5 to 8 sentences brief the complete match summary. It includes information like the winner of the match, player of the match, key players in the respective match, and their scores. We decided to filter the first 6 sentences from the news article and use them as reference summaries.

Processing of the commentary data and processing of the reports are conducted in different files. Because of that both are saved into different file formats. Later both the files are loaded into pandas dataframe and exported into excel files. Post processing there were 465 match instances of data is left in the dataset. These datasets are divided into three parts namely training set, test set, evaluation set in the ratio of 80%:10%:10%.

5.5 Implementation of Pre-Trained model - T5 base

T5 model was developed by Google(Raffel et al. (2019)) and it is pre-trained on data-rich tasks. The knowledge gained during the training phase is retained with the model and it is applied while solving the other different but similar problems. We are leveraging the HuggingFace transformers library which is an extremely famous python library for providing the pre-trained models that are significantly helpful in various natural processing tasks. It supports PyTorch, TensorFlow 2. In this research work, the HuggingFace library is being used for the implementation of the T5 model using PyTorch. PyTorch is an open-source machine learning library developed by Facebook AI researchers. We will be using the PyTorch software package as well for the implementation. We first implement the PyTorch Dataset class by inheriting the Dataset class. This class basically houses the functions required for fetching our dataset and loading data into the data loader and then feeding it to the network for model fine-tuning. LightningDataModule class inherits PyTorch Lightning data module. This class implementation consists of instance methods like train_dataloader, test_dataloader, val_dataloader for loading respective data into DataLoader. A LightningModule is implemented which inherits the PyTorch lightning module. Lightning modules do not define the model but define the system. It houses methods for the train loop, the validation loop, the prediction loop, the optimizer, the inference. We have used 'adam' optimizer for optimizing the weights of neurons and and learning rate was set to 0.0001. T5Sum is a custom class and it defines methods like from_trained, train, load_model, predict. from_trained method loads the pre-trained t5-base model. First, we create an instance of the T5Sum class. From this instance, we call the from_pretrained method to set the T5tokenizer and model. Tokenizers are used to create chunks of words from the text. We commence the model's training phase after loading the tokenizer and T5 model. To train the t5-base model we defined the train method which takes important arguments like training data frame, evaluation data frame, source maximum tokens, target maximum tokens, number of epochs. Internally these arguments are passed to the LightningDataModule and LightningModel class constructor to create their instance. Also, a trainer class instance is created and using this, a model is fit on the training dataset. Once the model is trained on the training data it is can be loaded using the load_model method. Once the model is loaded it can be used for the prediction of output summary. Dataset is split into training, test, and evaluation datasets. Model is trained using train dataset and for the testing, the test dataset is employed.

5.6 Implementation of Pre-Trained model - BART

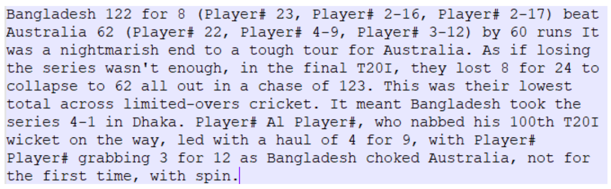
BART is a model developed by Facebook(Lewis et al. (2019)) which is a denoising autoencoder. It is trained by corrupting the text documents and optimizing the reconstruction loss in regenerating the source document. In this research study, we have used the publicly available simple transformer package of python for loading the Pre-Trained BART model. The simple transformer provides a simple way of utilizing the pre-trained models. We have used the Seq2SeqModel class of simpletransformers.seq2seq package for our text summarization task. We have also created the instance of Seq2SeqArgs class for defining the important arguments for the model. Using this instance we set the parameters like the number of epochs, the maximum length, optimizer, early stopping, learning rate. Seq2SeqModel class constructor method takes important arguments like encoder_decoder_type, encoder_decoder_name, Seq2SeqArgs instance. Internally the Seq2SeqModel connects with the HugginFace transformers library to load the requested model. Tokenizer, optimizer initialization happens internally. Dataset is divided into a training set, test set, evaluation set. On Seq2SeqModel instance train_model method is called to which training, as well as evaluation dataset, are passed. This initializes the training phase of the model. After the training phase, the model is evaluated using an evaluation dataset. Then evaluation loss is calculated. Both evaluation and training loss values are illustrated on the graph using the matplotlib library. We have experimented by running the model for different epochs and tried to find the optimal epoch value for which a model showed good results. Also, the training and evaluation loss graphs are illustrated to understand the model training process.

5.7 Implementation to mask the Player names in the dataset using Named Entity Recognizer(Standford NER)

Standford NER is a Java based Named Entity Recognizer. NER recognizes the person name, company name in the sequences of words in the text. NER jar files are downloaded into the local machine. NLTK library provides a way of reading these java files into python. We have implemented code to identify the player names in each match data. Some of the names identified are not proper. We have done extensive processing of these names manually. Then all the player names present in each match data are replaced by the special token 'Player'. Figure 7 shows the text with masked player names.

6 Evaluation

In this section, we discuss about the evaluation process and interpretation of the model results. As we know, this research work goal is to generate the news or summary from the



```
Bangladesh 122 for 8 (Player# 23, Player# 2-16, Player# 2-17) beat
Australia 62 (Player# 22, Player# 4-9, Player# 3-12) by 60 runs It
was a nightmarish end to a tough tour for Australia. As if losing
the series wasn't enough, in the final T20I, they lost 8 for 24 to
collapse to 62 all out in a chase of 123. This was their lowest
total across limited-overs cricket. It meant Bangladesh took the
series 4-1 in Dhaka. Player# Al Player#, who nabbed his 100th T20I
wicket on the way, led with a haul of 4 for 9, with Player#
Player# grabbing 3 for 12 as Bangladesh choked Australia, not for
the first time, with spin.
```

Figure 7: Sample masked dataset

live text commentary, we trained T5 and BART transformer models for summarization work. Given that our target variable data type is text summary, we require metrics that compare the generated summary to the reference summary and indicate how close and fluent the generated summary is to the reference summary. Therefore, we have used the popular text summary evaluation metric Recall-Oriented Understudy for Gisting Evaluation(ROUGE). ROUGE score near to zero indicates low similarity between the generated summary and reference summary, whereas ROUGE score close to 1 shows that the created summary and reference summary are quite comparable. We experimented with the models with different parameter values and selected the models parameters which showed high ROUGE values as final models. We have selected the following three models and their ROUGE score results are showcased in the Table 2.

Table 2: ROUGE Score of all models

Models	**ROUGE-1**	**ROUGE 2**	**ROUGE-l**
T5 base	32.8125	6.9444	29.6875
BART	37.5147	10.4743	32.1678
BART with mask data	**38.2057**	**5.1557**	**34.8633**

6.1 Experiment 1

First we fine-tuned the pretrained T5 model with our custom dataset. Dataset was split into training, test, evaluation set. T5 model configuration is given in the table Table 3.

Table 3: Model 1 configuration

Model	T5 base
source maximum token length	512
target maximum token length	128
batch size	4
Epochs	5
Early stopping	True
Learning Rate	0.0001
Optimizer	AdamW

After training the T5 model with our own dataset, we imported it from its stored location and used it to forecast the output summary. On the test dataset, we discovered the model's ROUGE score. ROUGE score of T5 base model is showed in the Table 2. The

ROUGE 1 accuracy score is 32%, which implies that 32% of the uni-grams in the produced summary are also present in the target summary. The precision score of ROUGE 1 is acceptable, however the ROUGE 2 value of 6% is less than the permissible margin. The ROUGE-l value is 29.68 %, which is a good result.

6.2 Experiment 2

Second model we trained is BART model. It is fine-tuned with the our custom dataset. Model configuration shown in the Table 4.

Table 4: Model 2 configuration

Model	BART
source maximum token length	1024
target maximum token length	1024
batch size	4
Epochs	5
Early stopping	True
Learning Rate	0.0001
Optimizer	AdamW

We trained the model on training dataset and trained model is used for prediction of summarise for the test dataset. It is discovered that the model summaries are excellent and provide high ROUGE scores. ROUGE score of the BART model is mentioned in the Table 2 . It generates summaries rather well, with a ROUGE 1 precision value of 37.5 percent, a ROUGE 2 precision value of 10.47 percent, ROUGE-l precision value of 29.6%. Results are slightly higher compared to the model 1 scores. This indicates that the high number of overlapping words present in BART generated summary are also present in the reference summary.

6.3 Experiment 3

Third experiment done using the masked data. BART model is trained on the masked data with parameter configuration showed in the Table 5 . Results of the model are

Table 5: Model 3 configuration

Model	BART
source maximum token length	1024
target maximum token length	1024
batch size	4
Epochs	10
Early stopping	True
Learning Rate	0.0001
Optimizer	AdamW

showcased in the Table 2.ROUGE 1, ROUGE 2, ROUGE L values are 38.2057, 5.1557, 34.8633 respectively. We can clearly see that the model is producing the significant

ROUGE 1, ROUGE L score. But fail to improve the ROUGE 2 values. In terms of generating common 1 grams and longest subsequent model 3 is performing well.

6.4 Discussion

The research project on Cricket live text commentary was a challenging work right from the data collection to the model evaluation. We have extraordinary work on the data creation part. We have scraped data of 782 matches. As the dataset gathered was unstructured and consists of a lot of noise in it we have conducted rigorous data processing tasks to prepare data for model training. We implemented the pre-trained models T5 base, BART. Models are fine-tuned with our dataset. It was observed that the sequence2sequence models tend to generate summaries with high-frequency player names instead of the actual player names present in the commentary. To address this issue, we employed a template2template model where input and output templates are with masked player names. This approach produced good results in terms of the ROUGE 1 and ROUGE L but scored less ROUGE 2 value. Dataset consists of a lot of noise. Names of the player are mentioned differently at different positions in the data. For example, AB de Villiers was written completely at some places, and in other places, it was just de or AB or Villiers. Due to this Named entity recognizer failed to find the names properly. We should have conducted manual cleaning of player names. Maybe due to this reason the model was not generating fair results. In fine-tuning the hyperparameters, we should have done more trial and error. Furthermore, since the sample size was small, the model was unable to understand the pattern in the data adequately. Due to time restrictions, we gathered around 800 matches of live text commentary. Overall, we were able to produce state-of-the-art results with fair and fluent summaries using the BART abstractive summarizer in template2template way.

7 Conclusion and Future Work

In this research work, we implemented challenging task Cricket game news generation using live text commentary. We have done excellent work in creating the dataset for our research study. A system is developed based on the pre-trained model BART for summarising the Cricket game using live text commentary. We trained the model in a template2template way. Our proposed method is appropriate. Model is able to produce fair and fluent summaries with high ROUGE 1 and ROUGE L scores. ROUGE 2 score is bit low.We propose that future effort focus on enhancing the model's ROUGE 2 score and contributing data to CricSum dataset. Source code and dataset is available on GitHub.[2]

References

Bajaj, A., Dangati, P., Krishna, K., Ashok Kumar, P., Uppaal, R., Windsor, B., Brenner, E., Dotterrer, D., Das, R. and McCallum, A. (2021). Long document summarization in a low resource setting using pretrained language models, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International*

[2]For source code: `https://github.com/sachinm226/CricSum`

Joint Conference on Natural Language Processing: Student Research Workshop, Association for Computational Linguistics, Online, pp. 71–80.
URL: *https://aclanthology.org/2021.acl-srw.7*

Chu, E. and Liu, P. (2019). MeanSum: A neural model for unsupervised multi-document abstractive summarization, *in* K. Chaudhuri and R. Salakhutdinov (eds), *Proceedings of the 36th International Conference on Machine Learning*, Vol. 97 of *Proceedings of Machine Learning Research*, PMLR, pp. 1223–1232.
URL: *https://proceedings.mlr.press/v97/chu19b.html*

Dou, Z.-Y., Liu, P., Hayashi, H., Jiang, Z. and Neubig, G. (2021). GSum: A general framework for guided neural abstractive summarization, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, Online, pp. 4830–4842.
URL: *https://aclanthology.org/2021.naacl-main.384*

Gehrmann, S., Deng, Y. and Rush, A. (2018a). Bottom-up abstractive summarization, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Brussels, Belgium, pp. 4098–4109.
URL: *https://aclanthology.org/D18-1443*

Gehrmann, S., Deng, Y. and Rush, A. M. (2018b). Bottom-up abstractive summarization, *EMNLP*.

Gu, J., Lu, Z., Li, H. and Li, V. O. (2016). Incorporating copying mechanism in sequence-to-sequence learning, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Berlin, Germany, pp. 1631–1640.
URL: *https://aclanthology.org/P16-1154*

Gunasiri, D. and Jayaratne, L. (2019). Automated cricket news generation in sri lankan style using natural language generation.

Huang, K.-H., Li, C. and Chang, K.-W. (2020). Generating sports news from live commentary: A chinese dataset for sports game summarization, *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pp. 609–615.

Kanerva, J., Rönnqvist, S., Kekki, R., Salakoski, T. and Ginter, F. (2019). Template-free data-to-text generation of Finnish sports news, *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, Linköping University Electronic Press, Turku, Finland, pp. 242–252.
URL: *https://aclanthology.org/W19-6125*

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V. and Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, *arXiv preprint arXiv:1910.13461* .

Li, H., Zhu, J., Zhang, J., Zong, C. and He, X. (2020). Keywords-guided abstractive sentence summarization, *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34, pp. 8196–8203.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries, *Text summarization branches out*, pp. 74–81.

Liu, Y. and Lapata, M. (2019). Hierarchical transformers for multi-document summarization, *arXiv preprint arXiv:1905.13164* .

Marek, P., Müller, Š., Konrád, J., Lorenc, P., Pichl, J. and Šedivỳ, J. (2021). Text summarization of czech news articles using named entities, *arXiv preprint arXiv:2104.10454* .

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer, *arXiv preprint arXiv:1910.10683* .

Saito, I., Nishida, K., Nishida, K. and Tomita, J. (2020). Abstractive summarization with combination of pre-trained sequence-to-sequence and saliency models, *arXiv preprint arXiv:2003.13028* .

See, A., Liu, P. J. and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Vancouver, Canada, pp. 1073–1083.
URL: *https://aclanthology.org/P17-1099*

Wang, J., Li, Z., Yang, Q., Qu, J., Chen, Z., Liu, Q. and Hu, G. (2021). Sportssum2.0: Generating high-quality sports news from live text commentary, *Proceedings of the 30th ACM International Conference on Information amp; Knowledge Management*, CIKM '21, Association for Computing Machinery, New York, NY, USA, p. 3463–3467.
URL: *https://doi.org/10.1145/3459637.3482188*

Zhang, J., Yao, J.-g. and Wan, X. (2016). Towards constructing sports news from live text commentary, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany.

Zhang, S., Celikyilmaz, A., Gao, J. and Bansal, M. (2021). EmailSum: Abstractive email thread summarization, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Association for Computational Linguistics, Online, pp. 6895–6909.
URL: *https://aclanthology.org/2021.acl-long.537*

Zolotareva, E., Misikir Tashu, T. and Horvath, T. (2020). Abstractive text summarization using transfer learning.