Automated Cricket News Generation in Sri Lankan Style using Natural Language Generation

M. H. D. Y. Gunasiri

2019

Automated Cricket News Generation in Sri Lankan Style using Natural Language Generation

A dissertation submitted for the Degree of Master of Science in Computer Science

M. H. D. Y. Gunasiri

University of Colombo School of Computing

2019

Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name:

Registration Number:

Index Number:

_____

Signature: Date:

This is to certify that this thesis is based on the work of

Mr./Ms.

under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:

Supervisor Name:

_____

Signature: Date:

Abstract

Automated sports journalism is a relatively a new technologically innovative field where actual journalistic content is created through Natural Language Generation (NLG). NLG is a sub task of natural language processing where the human readable form of text is composed from a non-linguistic representation of information. This dissertation provides a bibliographic review on applications of NLG related to journalism and other related NLG researches, as well as it provides a review on the tasks which are associated with the NLG process. Techniques and tools which support to perform those tasks and evaluate them, are also discussed.

Cricket is one of the most followed sport in South Asia. There is a wide requirement for news to get produced within a short period after the cricket match. Within Sri Lankan publishing this news article is a manual process and needs journalists with domain and language competencies. Most of the time it is not possible for sport journalists to focus on in-depth reporting due to time and cost constraints. Therefore, automated process would be efficient and cost effective.

As a solution, an Automatic Cricket News Generation System is presented through this research and how a template based natural language generation is utilized in implementing such system and its suitability are demonstrated. A system which could generate a journalistic summary of a cricket match using a score card in Sri Lankan style is implemented using pipeline approach in NLG. A methodology based on pipeline architecture is proposed for the system and it states how the data is transformed in each level in the architecture. Furthermore, it also focuses on the variation of the output generated via the system which is not much typically be in used in template-based NLG systems. The templates are created using the actual corpus written by journalists for the Sri Lankan newspapers.

This system is evaluated under manual and automatic generation. At the same time each module is unit tested. The generated text is evaluated by comparing with a reference text under 3 parameters which are Similarity Score, Degree of closeness, Data count. The results show that the generation system is capable of producing a grammatically correct and easy to read news piece for a given cricket match. The summary generated was also compared with a summary written manually by an expert and it shows that although creativity is not in a satisfactory level, accurate information could be gained through the summary generated.

Acknowledgements

I take this opportunity to express my gratitude towards those who helped me to make this project success.

I would like to show my gratitude to my supervisor Dr. Lakshman Jayaratne for the great support lent to me by showing the shortcomings in the research and for sending me in the right track with proper guidance throughout this research.

Next, I would like to express thanks to course coordinator Dr. L. N. C. De Silva who were being a helping hand throughout the research. I also thank all the lecture panel in UCSC for strengthening our knowledge during the Masters program.

Last but not least I remind my dear parents who are my strength and who are there with me through thick and thin.

Finally, I wish to thank all my batch mates and friends who helped me even through a word to make the research successful.

Table of Contents

Table of Figures

List of Tables

List of Abbreviations

API – Application Programming Interface

ICC – International Cricket Council

JSON – Javascript Object Notation

LGM – Language Generation Module

NLG – Natural Language Generation

NLTCT – Natural Language Test Case Template

NLU – Natural Language Understanding

REG – Referring Expression Generation

TTS – Text to Speech

XML – Extensible Markup Language

Chapter 1 : Introduction

1.1 Context and Motivation

With the vast development of technology, there is a significant increase in data collection. But some expertise knowledge is needed to manipulate and interpret those data in a human understandable form. Natural Language Generation (NLG) comes into play in such scenarios. NLG is a sub task of Natural Language Processing (NLP) where it uses techniques from Artificial Intelligence (AI) and Computational Linguistics (CL) in order to generate human understandable text automatically [1]. NLG is a fascinating area of research and many real-world applications are implemented based on it. NLG fulfills the need for systems to present data of most important aspects, in an understandable form for non-expert users.

Automating document generation is important because most people spend substantial portion of time in producing documents manually. Weather reports, discharge summaries of patients, code documentation by programmers, journalistic pieces are few such examples which would consume considerable amount of time and effort of users when they are prepared manually. Tools which would support those people quickly to produce or generate such documents may considerably enhance both productivity and morale.

NLG is getting popular in the context of journalism since the increase in availability of and access to data on a scale that prevents journalists from handling and using it for news reporting. The “robot journalism” or automated journalism [2], has reduced the variable cost for the journalism to the zero level. It is being researched that robot journalist can produce thousands of articles within shorter period of time. Cost and speed are two factors where automated recaps are clearly better than human sportswriters. In the domain of sports journalism, due to the abundant availability of information on databases about cricket matches, journalists might find it difficult to organize those data to produce news. If this editorial process is automated, journalists can spend time on in-depth reporting thus reducing time spent on large amount of data including statistics and numbers.

Cricket is one of the most popular sport in South Asian countries and it is getting popular among the other countries in the world too. Therefore, Cricket was chosen as the domain for sports journalism in this research. The main objective of this research is to generate journalistic piece on cricket matches. It would investigate different NLG techniques and the suitability of template-

based approach for generating news on cricket matches. The key aspect of this research is to have different template sentences which are changing according the situation of the match and thereby increasing the domain dependency. This system would produce a natural language output of text from structured data in a non linguistical format. Therefore, cricket news piece for a given match will be generated using the match score card.

Additionally, it was decided to use cricket in the context of sports journalism because input to the system is more standardized and widely accepted. Cricket match score cards which are available in several sports related online web sites are in a similar standard so that it can be used to create the journalistic summary and compare that with human uttered final commentaries or news in the similar websites such as cricinfo [3]. Another reason to use cricket match score cards to generate news is the availability of domain knowledge which most of the people around South Asian regions have. Therefore, a meaningful output can be generated with the help of templates to be used. Since the output of the NLG system which is researched here, generates piece of text, it is possible to use this output as an input to a Text to Speech (TTS) system so that an automated narration of news in spoken form can also be produced in future.

There are several forms of cricket matches recognized by International Cricket Council (ICC). But Twenty20 (T20) cricket matches are getting popular day by day because of newly introduction of several T20 leagues such as Indian Premier League (IPL), ICC World T20 and so on. It is also noted that there are no researches were done in producing cricket summaries for T20 cricket matches. Therefore, it was decided to generate news for Twenty20 cricket matches and rules and other related information that would be unique to T20 matches should also be considered when developing the NLG system.

1.2 Problem Statement

Over the last few years there has been a rapid development in technology which leads to an enormous change in many of the expectations of the community. For example, if an event happened a day before, people expect to see a full article on it the next day itself. Most of the time people used to gather few points in their mind and try to deliver these points as an article. Therefore, automating this process may help to make it more efficient and professional. Today, there are many cricket matches that take place at the same time and it is important to report news on all these

matches. Reporting news requires significant amount of time and effort of the journalists and they need more domain knowledge as well. As mentioned above, due to abundant availability of data in databases about matches, it is difficult for journalists to report every match by analyzing those data. If this editorial process is automated, more working time of journalists can be reduced considerably while they can spend more on in depth reporting. Because of the quicker news update gained through an automated cricket new generation, cricket fans who are keen on cricket news will be satisfied with the news update.

Journalism is not just description; it is an event driven story. Journalists need to be highly skilled in recognizing, creating, organizing and communicating stories in natural language. It is difficult to get the quality of journalism through an automated process since there are limitations and restrictions in the automated process of news generation compared to manual journalism. Most of the automated journalistic pieces are created through a template with a shorter description. However, extending the journalism from a short descriptive report to a qualitative journalistic piece is challenging in since the absence of required data may also lead to less user friendliness in the news. These templates used in automated sport news reporting, are more domain independent and general purpose. This needs to be addressed by this research and it would discuss how the user-friendliness is achieved by changing the template sentences according to the situation of the match.

As it is mentioned earlier, Cricket is already popular among the member countries of International Cricket Council and it is getting popular among the other countries as well. Many researches were done for automated journalism, but the area of sports is not covered much. Furthermore, there is a need for a research to generate cricket news automatically according to the Sri Lankan style and it is also not covered by the current researches which were done related to sports journalism.

1.3 Response - Proposed Solution

To address the above mentioned problem, a NLG solution is proposed in this research. NLG systems can broadly be classified as Real or Standard NLG and Template based NLG. Standard NLG systems are theoretically well-founded systems which embody generic linguistic insights while the Template-based NLG systems are natural language generating systems that map their

non-linguistic input directly to the linguistic surface structure [4]. In the proposed solution for automated cricket news generation, it is decided to use the template-based NLG.

Implementing complete Natural Language Generation system that operates in realistic domain needs a proper architecture so that it can be maintained in future. As the other software, it would be easier to build and debug if it is decomposed into distinct, well-defined and easily-integrated modules [5]. There are several high-level architectures are present for Natural Language Generation systems. Pipeline architecture will be used in this proposed research. It is a sequential architecture of NLG systems and it can be depicted as below.

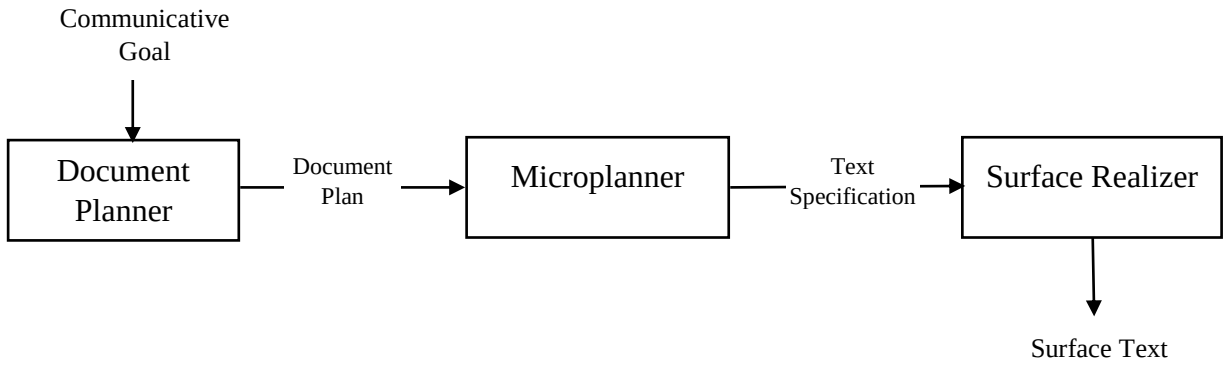

Figure 1.1: Typical NLG Pipeline

In addition, as shown in the figure 1.1 there are six (6) NLG tasks which are proposed by Reiter and Dale [7], attached with the pipeline architecture. They are Content determination, Text structuring, Sentence aggregation, Lexicalization, Referring expression generation, Linguistic Realization. These tasks are performed with each of the above-mentioned module in the pipeline architecture as mentioned below in the table 1.1.

Table 1.1: NLG Tasks

Module	**Content Task**	**Structure Task**
Document Planning	Content Determination	Document Structuring
Microplanning	Lexicalization	Aggregation
	Referring expression generation	
Surface Realization	Linguistic realization	Structure Realization

This architecture will be modified accordingly and would be used to define the research methodology.

1.4 Objectives

The main idea of this research is to generate cricket news automatically using information from structured database.

The intentions which are coupled with the main objective of this research are listed below:

- Generating journalist piece using Sri Lankan style of English Cricket news reporting
- Achieving a variation in the news summary according to the situation of the cricket match
- Understanding structured information on score cards, and restructuring them according to the specific format
- Generating grammatically and syntactically correct meaningful text according to the domain
- Compare and contrast the output text with the reference text.

In addition to above mentioned objectives, the research would discuss more on natural language generation systems and the possibilities of evaluation methods and so on.

1.5 Contributions

In this research, it is expected to build a system to generate news according to Sri Lankan style for international 20-20 cricket matches using the pipeline approach in NLG. It would show that how the template sentences will be selected according to scenario to produce a user friendly and domain dependent solution.

This thesis discusses how automated news generation would vary from manual editorial process when it comes to performances and other related aspects.

Chapter 2 : Literature Review

This chapter includes two parts. In the first section it describes the background of the technologies which the proposed system is built on while the second section contains the survey on related systems to automated journalism.

2.1 Natural Language Generation

Natural Language Generation (NLG) is a subfield of Artificial Intelligence and Computational Linguistics. It focuses on generating understandable texts in English or any other language by typically starting from nonlinguistic representation of information as input [7]. NLG can be regarded as an emerging technology since many real-world applications have been built based on this technology. From research perspective NLG is a subfield of Natural Language Processing (NLP).

Even though everybody would agree on what the output of a NLG system, but the input to a NLG system can vary substantially. Therefore, it is not easy to define NLG precisely. It is possible to distinguish NLG systems as two types depending on the input. They are D2T (Data to Text) and T2T (Text to Text) [8].

Data to Text – Normal input for D2T is a structured data which would be from a database, knowledge base, labeled corpus etc. Robot journalism is one good example for D2T system. They have a considerable impact in the fields of journalism and media studies.

Text to Text – Input to T2T system will be texts or isolated sentences and it includes learning how to express pieces of data in different or creative ways. Abstracted summarization system is one of the best examples for T2T system.

Difference between Natural Language Generation and Natural Language Understanding

NLU is the process of mapping human language to internal computer representation of information while NLG can be described as the inverse of this. But they share the same theoretical background since NLP is formed by combining these two areas [7].

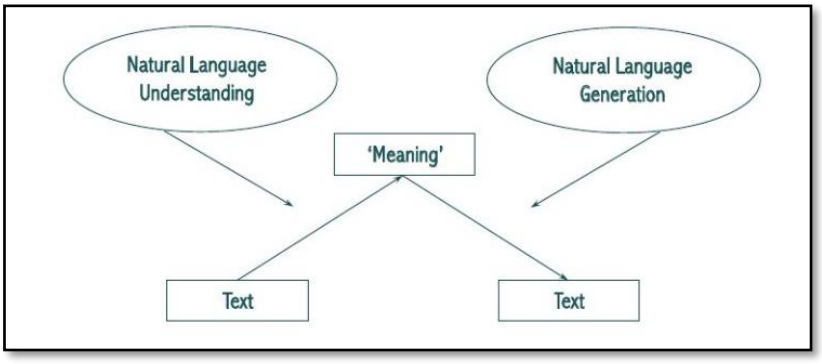

Figure 2.1: NLG vs. NLU

The figure 2.1 shows how NLU and NLG are combined to form NLP. According to the figure, text is converted to a meaningful output that can be understood by computers. This is done by the NLU process. This output can be a structure which is based on an intent or a domain. This meaningful output again can be converted to a text form which is human readable by the process Natural Language Generation. In this research project, only NLG is taken into consideration since the journalistic piece is generated using a score card which has a meaningful structure.

Classifying NLG systems according to the Communicative Goal

NLG is used when the canned text would not be sufficient, and a greater linguistic variation is required. These linguistic variations can be in different types as identified below[9].

Informative Text

In these types of systems, informative text will be generated by using factual data. Fog [13] is one such system which helps meteorologists to generate weather forecasts. This system generates textual weather forecasts from numerical weather simulation.

Textual Summaries

Using one or more sources textual summaries will be produced. This can be used in several domains such as sports, medical etc.

Persuasive Texts

These are the systems that try to persuade or take advantages of the user emotional state. An example for this kind of system is STOP [11]. It generates personalized letters which encourage people to stop smoking.

Simplified Text

These are the NLG Systems which helps peoples with cognitive disabilities and language barriers.

Dialogue Systems

These systems are designed to improve the communication between users and computers. There are so many such applications and chatbot is one of the best example. A chatbot or chatterbot is a software agent that allows the user to chat with a computer in human natural language. It creates the environment to the user to think that he/she is having a chat with another human being. This smart communication can be on textual or auditory ground.

Explanations

The output of these kind of systems explain the steps to be followed to execute an algorithm, process a transaction or solve a mathematical problem.

Recommendations

These systems will process user's preferences and opinions and thereby would give recommendations accordingly. AlethGen [12] is one such system which helps customer representatives to write the letters.

2.1.1 NLG Tasks

There are six NLG tasks identified in most of the NLG systems when converting the input to a human readable form of text [8].

Content Determination

At this stage it decides on the information to mention on the text. It filters out the most important information to be shown in the final output text. This contains choices. When it comes to the cricket domain, we may not need to include information about each and every ball. The content determination is available in most NLG systems and the approaches related to this task is closely related to the domain of the application.

Text Structuring

This will determine the order of information in the text. The application domain imposes the constraints on ordering preferences [10]. In the domain of cricket, before describing on special overs, hits, deliveries, balls in the match, it is better to start with general information such as when, where the match was played, toss, opening batsman etc.

Sentence Aggregation

Every message needs not be showed in separate sentences. By combining multiple messages into single sentence, it would be more fluid and readable. At the same time there are instances where the aggregation should be avoided. For example in cricket domain, when a player gets hattrick it is better to express it one sentence, than expressing it in three separate sentences for each out.

Lexicalization

Lexicalization is finding the correct words and phrases to express the information. Once the aggregation is done, the content of the sentence can be converted to natural language. The complication in lexicalization is that the same event can be expressed in several ways. In cricket domain, an event like player took a catch, many journalists would report this in different ways while giving the same meaning to the reader.

Referring Expression Generation

This action would select the words and phrases to identify domain entities. This would avoid ambiguity. According to Reiter and Dale, difference between lexicalization and REG is REG will contain the function of "discrimination task, where the system needs to communicate sufficient information to distinguish one domain entity from other domain entities" [2].

Linguistic Realization

This task involves ordering constituents of a sentence, as well as generating the right morphological forms. Here it would form a well-formed sentence. One of the challenge here is it would contain various linguistic components that may not be present in the input. This would convert the sentence to right morphological form.

There are different approaches in linguistic realization. i.e. human crafted templates, human crafter grammar-based system and statistical approach [8].

The above mentioned steps can further be categorized to 3 modules. They are Document Planning, Micro Planning and Realization [14].

- Document Planning
    - Content Determination
    - Text Structuring
- Microplanning
    - Aggregation
    - Lexicalization
    - Referring Expression Generation
- Realization
    - Linguistic Realization
    - Structure Realization

2.2 NLG Tools

There are several technical tools that can be applied in the NLG process, but these are mainly be used in the linguistic realization stage. However different programming languages supports different tools. Some of the tools are described below.

NLTK (Natural Language Toolkit)

NLTK is a suite of open source program modules, ready-to-use computational linguistic courseware. It covers most of the symbolic and statistical natural language processing. This toolkit mainly designed to be used with python language. This toolkit satisfies several requirements such as Ease of use, consistency, extendibility, documentation, simplicity and modularity [31]. This has very shallow learning curve and it is one of the primary purposes of this toolkit. There are several types modules defined here. Some of them are parsing modules, tagging modules, finite state automata, type checking and visualization modules.

This NLTK library will be used in several places in the automated sports news generation such as in preprocessing and so on.

SimpleNLG

This is a tool that can be used to generate natural language text in English Language. It helps to produce syntactically structured sentences and linearize them. This tool is mainly designed for large scale data to text NLG systems whose goals is to summarize large volumes of numeric and symbolic data.

Simple NLG is a Java Library and it defines set of lexical and phrasal types. This tool differs with most of the other tactical generators because it provides a transparent API to carry out low-level tasks such as inflection and syntactic combination, while making no commitments about input specifications or input-output mappings [32].

Natural OWL

Natural OWL is an open source multilingual natural language generator and it generates linguistically annotated OWL ontologies. OWL is a standard of defining ontologies in to a semantic web. This is also implemented in Java and XML is used for the communication at the processing stage. NaturalOWL follows the pipeline architecture where generation carried on in three stages i.e. document planning, micro planning and referring expression generation and surface realization.

In document planning, NaturalOWL would select all the ontologies that can be directly mapped with the instance that is taken for consideration. In the microplanning stage, one or more microplans are specified per language. NaturalOWL microplans are templates which are having slots to be filled [33].

PyNLPI

PyNLPI is a python library for NLP and is used for the basic tasks like extracting n-grams and frequency lists and the build simple language models. This has several packages and modules and licensed under GPL (General Public License) [34].

2.3 Template Based Approach

Template based systems are the natural language generation systems which maps nonlinguistic input directly to linguistic surface structure [15]. This linguistic structure has gaps and well-formed

text would be an outcome when all those gaps are being filled with linguistic structure that do not contain gaps.

According to Busemann and Horacek, there are two orthogonal methods which help for the efficient development in NLG systems, such as 1) general, reusable, linguistically motivated surface realization components 2) simple task-oriented template-based systems [16]. It is stated that use of these two methods are also limited. Because, using surface realization component is problematic since domain oriented and linguistically motivated ontologies are different. At the same time, it is observed that existing template-based techniques are inflexible [16].

Natural Language Test Case Templates (NLTCT) is a template based NLG methodology [17]. Through this it is able to achieve some kind of a flexibility through a template based NLG method. It generates NL description from each Z test case where it has parametrized operations. It can insert translation rules for these. Following figures 2.2 and 2.3 shows an example for Z test case and how it has been converted to NL description.

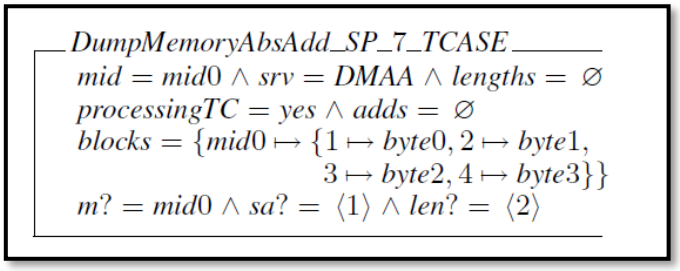

Figure 2.2: Z Test Case

Service (6,5) will be tested in a situation that verifies that:

- the state is such that:
  - the on-board system is currently processing a telecommand and has not answered it yet.
  - the service type of the telecommand is DMAA.
  - the set of sequences of available memory cells contains only one sequence, associated to a memory ID, which has four different bytes.
  - the set of starting addresses of the chunks of memory that have been requested by the ground is empty.

- the input memory ID to be dumped is the available memory ID, the input set of start addresses of the memory regions to be dumped is the unitary sequence composed of 1, the set of numbers of memory cells to be dumped is the unitary sequence composed of 2.

Figure 2.3: Converted Z test case

Although NLTCT is said to be inflexible but one of the limitations identified here is the domain independence. At the same time, it is difficult to maintain several test cases when the domain becomes larger.

Reiter and Dale had claimed that template based system have been substandard in respect to maintainability, output quality, variation and well-foundedness [7]. They are difficult to maintain, update as well. According to author's opinion, these are related to pure template based systems, if it can maintain a variation in templates many of these drawbacks can be avoided.

There are several template-based NLG systems. GoalGetter is one such example which generates soccer reports [18]. It is a data to speech system which consists of two modules. 1) Language Generation Module (LGM) 2) Speech Recognition Module [15]. When it comes to templates, we would give emphasis on LGM. A syntactical structured template is used to convert typed data structure to a natural language text. Syntactical structured template is shown by σ = (S, E, C, T) where S denotes the syntax tree, E symbolizes the additional structures such as NP (Noun Phrase) and PP (Prepositional Phrase). C denotes the conditions of applicability of S while T indicates the set of topics. The following figure 2.4 shows a sample syntax tree of GoalGetter [18].

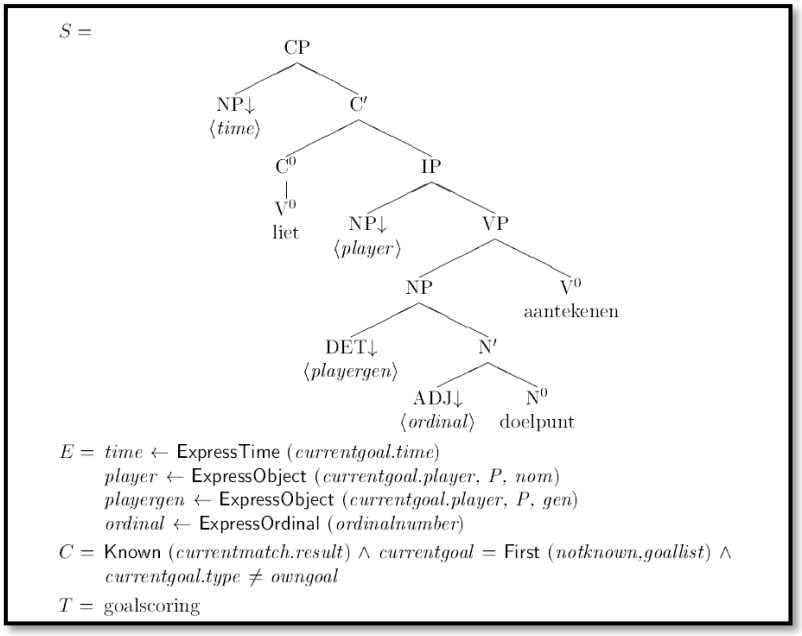

Figure 2.4: Sample syntax tree of GoalGetter[18]

These syntax trees have interior nodes which represent non-terminal symbols and frontier nodes characterize terminal and non-terminal symbols. The non-terminals are the gaps to be filled in the template [15].

It is a known fact that template-based systems have low maintainability [16]. The output quality and the variability in output can be achieved if template-based systems have the same generative power as the standard NLG systems [15]. One of the advantages of template-based systems is that the templates are good when the linguistic rules are not available or for constructions which have unpredictable meaning or highly specific conditions to use.

There are ways that limited variation can be achieved through template-based systems. They are using parameterized templates and Template Specialization Hierarchies [15]. Parameterization is a flexible way to guide and control the text generation process according to linguistic preferences such as styles, rhetorical structures. Here it dynamically sorts out the application of templates according to parameters. XTRAGEN [19] application is one of the examples which uses this method. Following figure 2.5 shows how the parameters are placed in templates and figure 2.6 shows how the parameters are set at the run time in XTRAGEN.

```
<template id="explainExpert"
          category="explain">
   <parameters>
      <parameter name="level"
                 value="expert">
      <parameter name="verbosity"
                 value="low">
   </parameters>
   ...
</template>
```

```
<template id="explainNovice"
          category="explain">
   <parameters>
      <parameter name="level"
                 value="novice">
      <parameter name="verbosity"
                 value="low">
   </parameters>
   ...
</template>
```

Figure 2.5: Parameters are placed in template - XTRAGEN

```
generator.addParameter("level",
                       "expert");
```

Figure 2.6: Run time in XTRAGEN

In template specialization hierarchies, decision tree traversal is taking place according to applicability conditions. EXEMPLAR [20] is one of the best examples uses this method which it finds the most specific exemplar for the current context. Using these above mentioned methods, advantages such as efficiency, simplified system architecture, full control over output and reduced demand on knowledge acquisition and representation can be achieved through a template based NLG systems.

2.3.1 Applications of NLG in Related to Journalism

Automated journalism is relatively new and technologically innovative field, but this area is touched through several researches. Los Angeles Times was the first newspaper to report the earthquake happened in 2014 close to Beverly Hills, Califonia. This was reported within 3 minutes and it was automatically generated by a 'robot journalist' which converts the input parameters to a pre-defined template [21]. This was a data-to-text generation system. Automated journalism needs data in machine readable format like spreadsheet or the data should be converted to a machine-readable format through machine learning or any other technology.

Automated journalism cannot be used for the domains with lack of data and when quality of data is poor. Additionally, there are six requirements [22] identified in automation of journalistic process. Namely,

Transparency – Making public the steps taken to process the data, the analysis code, the model or inferences made with it, software used, or data sources.

Accuracy – Journalistic NLG system must only contain factual and non-misleading data while achieving the basic standards of journalistic accuracy.

Modifiability and transferability of the system – It is the flexibility to cope up with the new knowledge in journalism while it is transferable to other domains so that budget and resources can be reduced significantly.

Fluency of Output – Coherent and fluent natural sounding output text is expected from a NLG based journalistic piece, since it helped to gain the customer satisfaction.

Data Availability – Data must be available in order to produce accurate reports. This affects the domain, topicality and the speed of the content generation.

Topicality of the news - To ensure user interest, the generated news should be given a relevant, meaningful and attractive topic

2.3.2 Automated Journalism Related to Sports Domain

Sports is one of the ideal domain for automated journalism because of rich data availability and input formation and style in news reporting. Several sports are being researched in sports automated journalism and some of them are baseball, soccer, football and cricket.

The games recap of Little League Baseball was provided the media coverage of Associated Press (AP). Using an artificial software from Automated Insights with the data provided by MLB Advanced Media (MLBAM), the official statistics provider, this robot reporter is produced [23]. There are similar applications and researches and some of them are listed below.

GoalGetter

GoalGetter is a data to speech system which generates spoken summaries of football matches in Dutch [24]. In order to generate, it takes in the information in tabular format and this is similar to a system called STREAK [25] which generates summaries of basketball instead football. STREAK generates summaries using a revision based approach to summarization. In the first pass, it generates draft with fixed information like who won, lost and in the second pass, opportunistically add in information, as allowed by the form of the existing text. The difference of GoalGetter from STREAK is that STREAK generates a written output while GoalGetter produces a spoken output.

GoalGetter retrieves information from Teletext which broadcasts text along television signal and decodes in the receiver end [24]. Main difference with the other NLG systems is that this does not use the pipeline architecture. Generation process and prosody needs 3 other sources except the domain data such as syntactic template, knowledge state and context state. Following figure 2.7 shows the Language Generation Module architecture of GoalGetter.

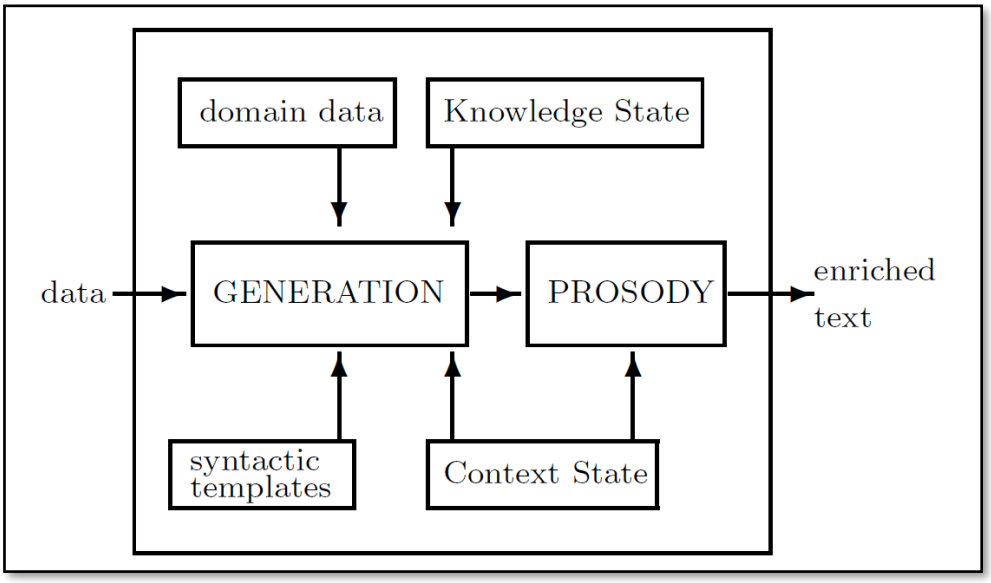

Figure 2.7: LGM Architecture of Goal Getter

Generation module contain the basic generation algorithm of LGM. Teletext data will be parsed to a machine-readable format and will be input to generation module. Additionally, to this input it needs domain data which contains fixed background data on the relevant domain. For example, in GoalGetter those are the data about the football teams and the players.

At the same time generation modules needs syntactic templates which are syntactic tree structures containing open slots for variable information. Each template is used during specific condition and interplay between these condition determines the generated text. Two records are kept during generation. Knowledge state shows which part of the input data structure is expressed by system and which part is not. This is done by labelling the all fields in input data structure. The other record is the Context State. It records various aspects of linguistic context.

GameRecapper System

The GameRecapper system is a template-based system that generates Portuguese summaries of football matches from structure input data such a game sheets taken from www.zerozero.pt website [9]. This data is converted to JSON tree and the implementation of GameRecapper is also based on GoalGetter system and the architecture of GameRecapper is shown in figure 2.8.

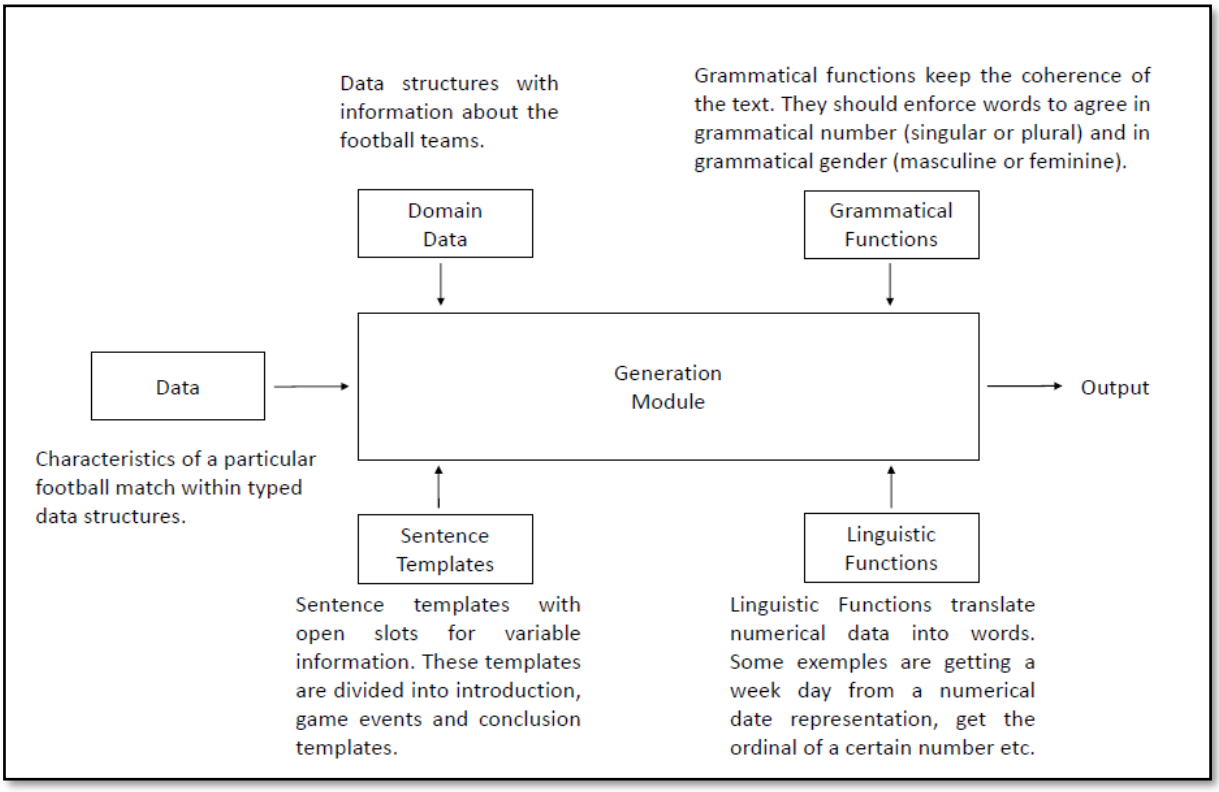

Figure 2.8: GameRecapper Architecture

GameRecapper generation module is basically the algorithm that creates the news. It utilizes the domain data, linguistic function, grammatical function and a collection of sentence templates. As in GoalGetter, domain data would provide the additional data to produce more variations in the output text while grammatical function ensures the coherence in the text. Linguistic function would convert the numeric values to words since this domain contains more numbers in the input data. The templates are divided into groups according to several types of events in the game such as first goal, game end etc. or different characteristics of the game such as home team became the winners etc [9].

The GameRecapper is producing grammatically correct and easy to read summaries with an average intelligibility score of 92.91% [9]. Since it uses larger amount of similar template sentences and it makes the content unnatural and it is prone to more repetition of information thereby reducing the average fluidity.

Automatic Generation of Multilingual Sports Summaries

Multilingual Sport Summary Generator investigates the suitability of template based system to generate multilingual sport summaries. The system generates English and Bangla Cricket Summaries using score card, which is very much similar to the requirement in this proposed research [26]. They have used pipeline architecture since it has low dependency while maintaining clear separation between each step and allows the extendibility of modules.

Input of the system is a cricket match score card taken from cricinfo. The architecture used in this system is shown in the figure 2.9.

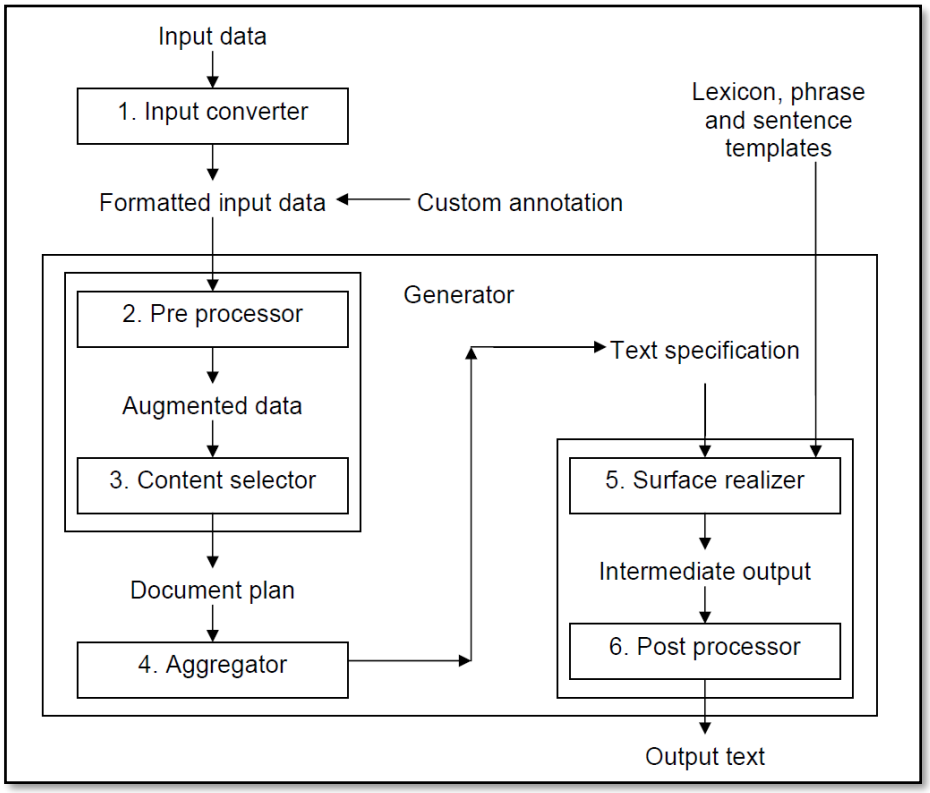

Figure 2.9: Multilingual System Architecture

Additionally, to the input, the system is fed with custom annotations which includes the background information which cannot be deduced by in the input score card itself. In this system contains six (6) custom tags which can be used to describe batting conditions and bowling conditions [26].

Preprocessor would create the news items from player portion and previous performances of the player and then would be passed to the content selection to select only the needed using content

selection rules. The order of the news is determined by the document plan and aggregator module recognizes the items to be aggregated to improve the meaning. However, referring expression generation is not performed in this system. Using the phrase templates and the output from aggregator module, it creates natural language sentences. Finally, the post processor outputs the generated text after applying all the other rules such as capitalization etc. [26].

Main purpose of this system is to utilize the portability of languages by easily generating summaries of cricket by adding similar template with different languages. This system has two levels of templates such as phrase templates and sentence templates to increase the variations in the output text.

Tamil Cricket Summary Generator

In this research cricket summaries are generated in Tamil using cricket match score card. This system makes use of two parameters namely, interestingness and humanness. The cricket summary generator has the following components [27].

1. Data Gathering and Modelling Module
2. Data Mining and Data Analytics Module
3. Summary Generator
4. Evaluator

In the data gathering module it would collect the data from input URL and it parses the web page and extracts the statistical data needed. Modified version of Apriori algorithm is used here and thereby interestingness of the match will be calculated during Data Mining and Analytics stage. The interestingness of the match is calculated on the weighted average of the scores according to the factors identified including the winning margin, team history, individual records made, high run rate, series state, relative position in international ranking, reaction in social networks and so forth. Summary Generator would create the summary using typical modules in a NLG system. such as Content Determiner, Aggregator, Tamil Morphological Generator and Layout Determiner. Summaries are compared with human written summaries at the Evaluator component. This degree of similarity between the generated summary and human authored summary shows the humanness feature of the system.

Chapter 3 : Research Methodology

The main purpose of this research is to generate a journalistic piece on a cricket match using cricket score card for the given match. This chapter would present the approach of template-based NLG system to fulfil the given purpose. As it is mentioned above, the main input to the system will be a 20-20 cricket match score card retrieved from cricinfo.

Following figure 3.1 shows a templated score card for a 20-20 match taken from cricinfo website.

India won by 7 wkts

England Innings 198-9 (20)

Batsman		R	B	4s	6s	SR		
Jason Roy	c Dhoni b D Chahar	67	31	4	7	216.13		
Jos Buttler (wk)	b S Kaul	34	21	7	0	161.90		
Alex Hales	c Dhoni b Hardik Pandya	30	24	3	2	125.00		
Eoin Morgan (c)	c Dhoni b Hardik Pandya	6	9	0	0	66.67		
Ben Stokes	c Kohli b Hardik Pandya	14	10	2	0	140.00		
Jonny Bairstow	c Dhoni b Hardik Pandya	25	14	2	2	178.57		
David Willey	b Umesh	1	2	0	0	50.00		
Chris Jordan	run out (Dhoni)	3	3	0	0	100.00		
Liam Plunkett	c Dhoni b S Kaul	9	4	0	1	225.00		
Adil Rashid	not out	4	3	1	0	133.33		
Extras		5 (b 0, lb 4, w 0, nb 1, p 0)						
Total		198 (9 wkts, 20 Ov)						
Did not Bat	Jake Ball							

Fall of Wickets

198-9 (Chris Jordan, 20), 194-8 (Liam Plunkett, 19.3), 183-7 (David Willey, 18.3), 181-6 (Jonny Bairstow, 17.6), 177-5 (Ben Stokes, 17.4), 140-4 (Alex Hales, 13.6), 134-3 (Eoin Morgan, 13.2), 103-2 (Jason Roy, 9.2), 94-1 (Jos Buttler, 7.5)

Bowler	O	M	R	W	NB	WD	ECO	
Deepak Chahar	4	0	43	1	1	0	10.75	
Umesh Yadav	4	0	48	1	0	0	12.00	
Siddarth Kaul	4	0	35	2	0	0	8.75	
Hardik Pandya	4	0	38	4	0	0	9.50	
Yuzvendra Chahal	4	0	30	0	0	0	7.50	

India Innings 201-3 (18.4)

Batsman	R	B	4s	6s	SR

Rohit Sharma	not out	100	56	11	5	178.57	
Shikhar Dhawan	c J Ball b Willey		5	3	1	0	166.67
Lokesh Rahul	c Chris Jordan b J Ball		19	10	1	2	190.00
Virat Kohli (c)	c & b Chris Jordan		43	29	2	2	148.28
Hardik Pandya	not out	33	14	4	2	235.71	
Extras		1 (b 0, lb 0, w 1, nb 0, p 0)					
Total		201 (3 wkts, 18.4 Ov)					
Did not Bat	Suresh Raina, MS Dhoni, Deepak Chahar, Umesh Yadav, Siddarth Kaul, Yuzvendra Chahal						

Fall of Wickets

151-3 (Virat Kohli, 14.5), 62-2 (Lokesh Rahul, 5.2), 21-1 (Shikhar Dhawan, 2.1)

Bowler	O	M	R	W	NB	WD	ECO	
David Willey	3	0	37	1	0	0	12.33	
Jake Ball	3	0	39	1	0	0	13.00	
Chris Jordan	3.4	0	40	1	0	1	10.91	
Liam Plunkett	3	0	42	0	0	0	14.00	
Ben Stokes	2	0	11	0	0	0	5.50	
Adil Rashid	4	0	32	0	0	0	8.00	

Match Info

Match	Eng vs IND, 3rd T20I, India tour of England, 2018
Date	Sunday, July 08, 2018
Toss	India won the toss and opt to bowl
Time	01:00 PM GMT
Venue	County Ground, Bristol
Umpires	T Robinson, R Bailey
Third Umpire	Michael Gough
Match Referee	D Boon
England Squad	
Playing XI	Liam Plunkett, Eoin Morgan (c), Adil Rashid, Jos Buttler (wk), Jonny Bairstow, Chris Jordan, Jason Roy, Ben Stokes, David Willey, Alex Hales, Jake Ball
Bench	Moeen Ali, Joe Root, Sam Curran
India Squad	
Playing XI	MS Dhoni (wk), Suresh Raina, Rohit Sharma, Virat Kohli (c), Shikhar Dhawan, Siddarth Kaul, Umesh Yadav, Deepak Chahar, Yuzvendra Chahal, Lokesh Rahul, Hardik Pandya
Bench	Dinesh Karthik, Bhuvneshwar Kumar, Manish Pandey, Kuldeep Yadav, Krunal Pandya

Figure 3.1: Scorecard

The first two paragraphs of the summary report news presented by for the above match is given below in figure 3.2.

Rohit Sharma's record third T20I hundred following Hardik Pandya's career-best haul of 4 for 38 gave India the T20I series as they beat hosts England by seven wickets in the decider in Bristol on Sunday (July 8). Led by Pandya's four-fer, the bowlers paved way for India to claw back into the game after a blistering start from the English openers. With Rohit at his best, India hardly had to break a sweat as they overhauled the target of 199 with more than an over to spare.

After playing a spectator at the other end to KL Rahul's century in the series opener, Rohit showed that the flicks and the swivel pulls were all in place as he launched India's chase in style despite losing his opening partner Shikhar Dhawan early. England erred by feeding him with short balls, helping Rohit find his momentum in no time. Rahul, who opened his account with a copybook straight drive, took over for a brief moment in the game as he lofted Liam Plunkett and pulled Jake Ball into the stands in the space of three deliveries. But that's how long his show lasted as Jordan pulled off a stunner at wide of long-off fence to break the budding partnership.

Figure 3.2: Human Authored Cricket News

It is obvious that it is not easy to generate a news as above. It needs domain and background knowledge in order to produce a news with relevant content. For example some of the information available in the first sentence such as Rohith Sharma's thirst T20 hundred and Hardik Pandaya's career best wickets, cannot be found from the score card. They need to be input somehow to the system to generate the output as above. However, output as above which is rich in language and user friendliness is difficult to be expected from an automated system. The intention of the proposed system is to generate natural language summary which utilize the given input and extract the key semantic concepts.

Moreover, this research is based on generating cricket news from the Sri Lankan context of sports journalism. As mentioned above within Sri Lanka, automated journalism is not practical since it is not been touched by any researches. However, there is a clear difference between Sri Lankan cricket news as in figure 3.3 which is a summary on the above match mentioned in the Daily Mirror when compared with the cricket summaries that are published in the internationally recognized sports web sites like in cricinfo (Cricinfo match summary for the above match is shown in the figure 3.4). Unlike in international cricket news reports, Sri Lankan cricket news mostly contain the match summary without any in depth reporting and expert judgements.

South Africa captain Faf du Plessis scored an unbeaten century to lead his side to a comfortable eight-wicket victory over Sri Lanka in the first One-Day International at The Wanderers on Sunday.

After winning the toss and electing to field, South Africa bowled the visitors out for 231 in 47 overs before chasing down their victory target for the loss of two wickets and with 67 balls remaining.

Kusal Mendis (60 from 73 balls) top-scored for Sri Lanka before he was one of a trio of victims for leg-spinner Imran Tahir (3-26 in 10 overs), the pick of the home bowlers.

Seamer Lungi Ngidi (3-60) also made a solid return to the team after four months on the sidelines through injury.

South Africa never looked troubled in their reply as Du Plessis (112 in 114 balls) and Quinton de Kock (81 from 72 balls) put on 136 at more than a run-a-ball for the second wicket.

Du Plessis' century was his 11th in ODI cricket and continues his fine form in the 50-over format with 259 runs in his last four innings, during which time he has only been dismissed once.

The second fixture in the five-match series will be played in Pretoria on Wednesday.

Both teams are using the series to fine-tune their selections ahead of the World Cup in England and Wales that starts in late May.

Figure 3.3: Daily Mirror Cricket Summary for a match

South Africa *232 for 2 (du Plessis 112*, de Kock 81) beat* ***Sri Lanka*** *231 (Mendis 60, Oshada 49, Tahir 3-26, Ngidi 3-60) by eight wickets*

Faf du Plessis led a resounding comeback in the grand scheme of the tour as South Africa utterly dominated Sri Lanka for their first win against the visitors. South Africa went 1-0 up in the five-match ODI series thanks to du Plessis' century, and his rapid 136-run stand with Quinton de Kock that left Sri Lanka with no chance after they had been bowled out for 231. Their eight-wicket win was completed with 11.1 overs to spare.

Du Plessis has scored at least one half-century in each of his last five games across formats, and he looked a man in supreme touch after walking in at 14 for one at the end of the second over. He barely mistimed a ball early in his innings and any confidence Sri Lanka could have drawn from getting Reeza Hendricks early - caught behind off the inside edge - slowly evaporated as he got on top of anything in the off-stump channel and punished even marginal errors in length with regal drives which often came in clumps. He first hit Vishwa Fernando out of the attack with front-foot drives either side of extra cover, and then took on Thisara Perera in the allrounder's first over.

Three boundaries - a back-foot punch, a step out and drive past mid-off, and then a mean slap in front of point - in a row against Thisara definitively grabbed the momentum for South Africa, but an opportunity was handed next ball. Looking to cut again, du Plessis only got a thick edge on a length ball that rose at him and offered a gentle job that was headed straight for Lakshan Sandakan at short third man. Sandakan merely had to get the reverse cups out and hold at chest height, but he seemed to be caught by surprise and fluffed the only chance Sri Lanka would get.

Figure 3.4: CricInfo Summary for a given match

The architecture of the proposed system is based on standard pipeline architecture of NLG systems as mentioned in the introduction chapter. The reason behind using the pipeline architecture is that the phases of the architecture has low dependency and each component has well defined task that would transform the data in a specific manner. One of the disadvantages of this architecture is that it is one way unless it is customized. One such example would be it is not easy to change the content selected at the content determination stage at latter stages. Nevertheless, the generation is one-way pass and it would not need to go back and change what has done.

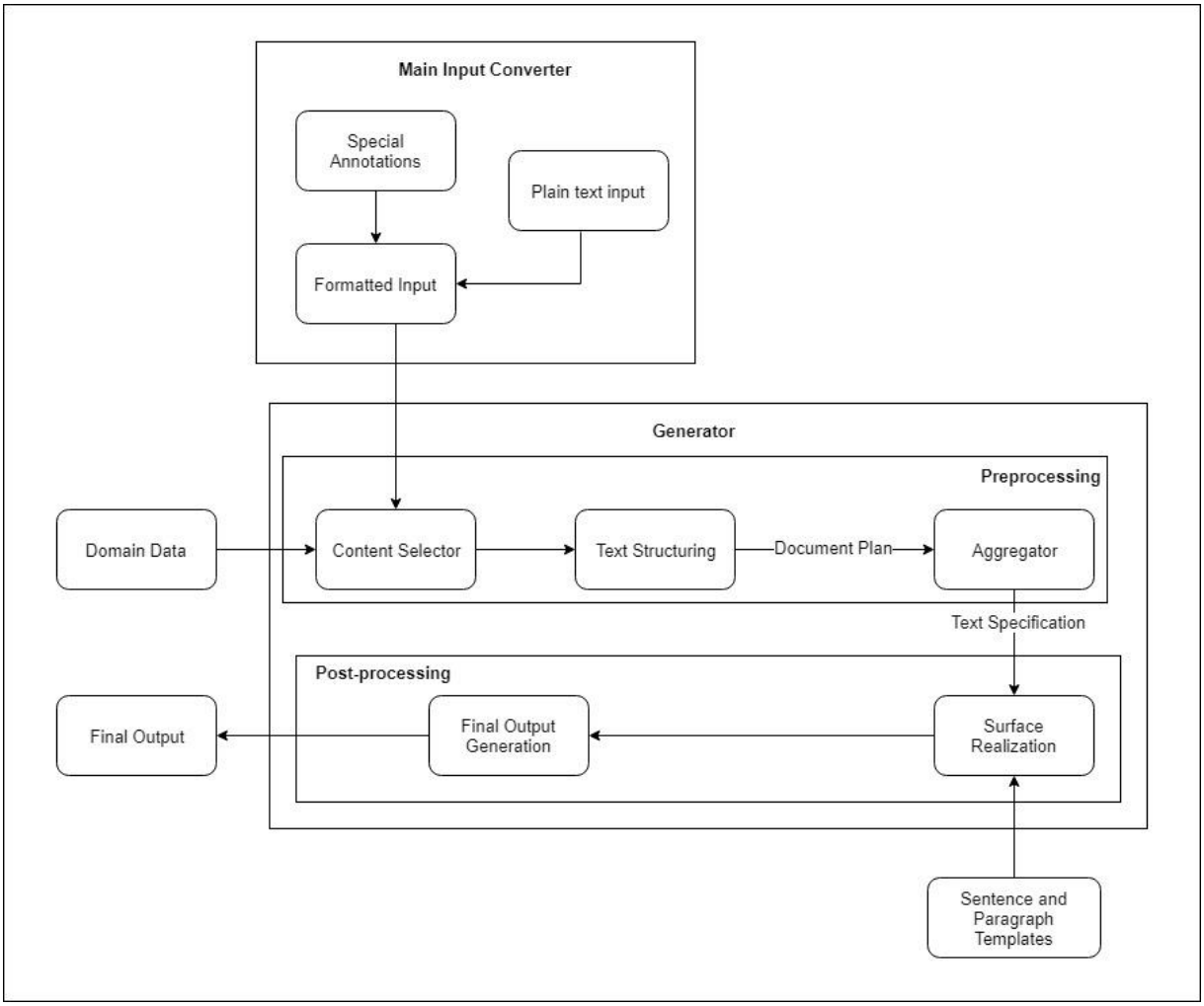

Figure 3.5: Architecture

Above figure 3.5 depicts the main architecture design for the proposed system in this research. According to the architecture, the generation module is mainly based on the traditional pipeline architecture. As shown, the input generator works as a separate module since the text generation process only interests on the final input. This final input needs to be in a proper standard structure and in this case, it would be a json format. This json tree would characterizes the basic information about a selected match.

The domain data and paragraph and sentence templates are another source of input to generator module. This domain data module contains different information that helps to generate the final output. For example, it would contain the information about the teams. The coach of the team, the

players who are not playing for the given match from the team, fresh players of the team are some of the information about teams.

Sentence and paragraph templates are one of the main input to generator module. These are incomplete sentences or paragraphs with open slots for variables. There are several similar templates which would be used according to the instance. Choosing the correct template is a task for the generator module. Variation in the final output depends on the sentence templates choices that is made. Templates are divided in to groups according to some criteria such as where to be used in the final output. For example, sentence templates for the introduction would be grouped together.

3.1 Main Input Converter

As mentioned above score card for the relevant cricket match would be the main input to the system. But this score card is in the plain text format. The plain text is converted to JSON tree by a parser while an algorithm will be used to make this task fulfilled. This JSON tree acts as the main input to the text generator module and this JSON tree must be properly structured so that it would be easier to be utilized by the generator. The subsections of the JSON tree would be the game overview with the game results and teams played, scores by the two teams, background information such as location of the game etc.

Corresponding to each player there will be a separate node and each of these nodes consists of leaf nodes which contains the performance during the match. Special annotations will also be attached with each player node if there is anything special to be included. The special annotation tags will contain the specific background information on each player. For an example with this player node system is able to deduce that the player got out by playing a poor shot. These custom tags would make the final output text more meaningful.

3.2 Content Selector

The content selector module would get the input and would manipulate according to some criteria given.

There are several approaches for content selection. Some of them are rule based content selection and trainable content selection [28]. In trainable approach to content selection, learning based method is used to choose the content as an independent task or jointly with the surface realization NLG task. In order to use learning method there should be a significant amount of input data. Therefore, in this proposed system, the rule based approach would be used and trainable approach can be implemented in future and compare it with the current system.

3.3 Text Structuring

After the content selection is performed, text structuring module is responsible to order these items that were chosen. According to sub sections of the final output such as introduction, result, overview etc., the text selected will be organized. Similar concept will be grouped together according to a rule based method. Content selection and Text structuring will be performed together.

3.4 Aggregator

Aggregator would get the output of the text structuring module and determine which of the items can be output together from the list. Here duplicate items will be discarded and similar items in each list after text structuring will be aggregated together based on the concepts. The purpose of this module is to form a single sentence with the consecutive items which are potentially candidates for aggregation.

3.5 Surface Realization

The Surface Realization Module would take the output of the aggregator module together with the linguistic functions, grammatical functions and templates. The linguistic functions would change the numeric or date format to text format. The linguistic functions will convert the data with different types formats such as numeric, date to text format. The grammatical functions would keep

the coherence and concordance of the text. Singular and plural words and grammatical gender (male and female) words would be handled here. Sentence or phrase text will be used in this proposed system as it is mentioned above. For an example, for the introduction which contains the match summary with background information, paragraph templates will be used. Most of the time grammatical functions will be helpful in sentence templates than paragraph templates because it contains the most common contents and the templates themselves are linguistically motivates with few gaps to be filled.

3.6 Final Output Generation

In this final output generation, it would get the output of the surface realization module and some minor validation checks will be performed. If these minor validations return false, it would make the relevant changes. Capitalization of the first letter of a sentence is one such validation check. Output of this module will be final summary generated on the given domain.

Chapter 4 : Research Design

This chapter discusses how the above mentioned methodology can be applied in the proposed system.

4.1 Data Collection

As stated above the main input which is the score card will be taken from the cricinfo websites. This score card is in a plain text format. There are two other major input to the system which are domain data and templates. The domain data will be taken from the same cricinfo site while the fetched values will be stored in a database. Domain data contains data about team players and other information about teams. The templates are designed according to DailyMirror [28] cricket news.

4.2 Templates Design

As mentioned above there are two types of templates designs. i.e. Sentence Templates and Paragraph Templates. A cricket match news piece should contain both information related to the match as well as some background information.

Templates which are based on the basic information related to the match will be designed based on batting, bowling and fielding actions of the players. Key semantic concepts of the basic information about the match are put in to templates according to the below structure as in figure 4.1.

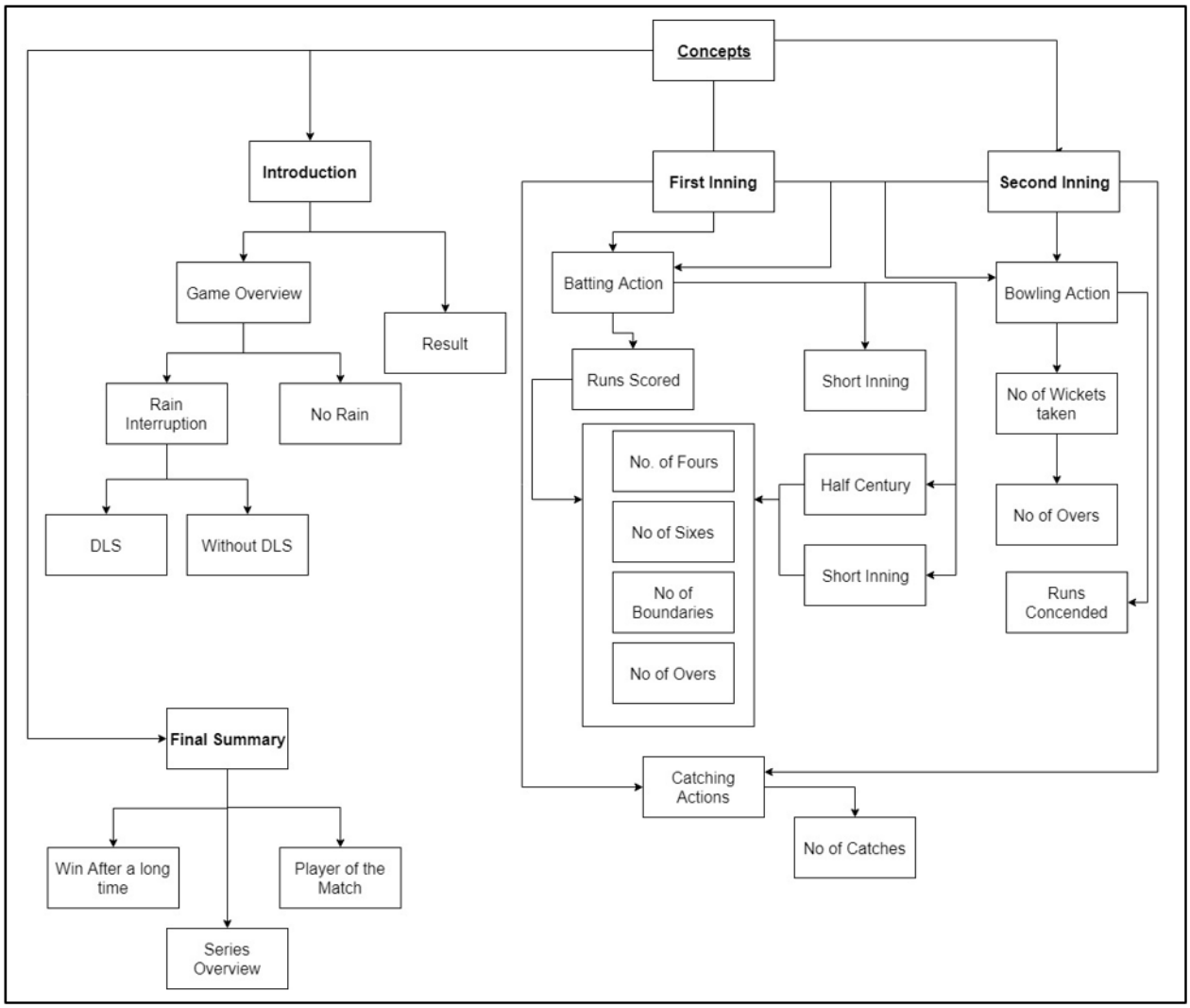

Figure 4.1: Key Semantic Concept

Addition to the above information that can be generated from the templates, with the help of domain data there can be some other information be generated.

The background information should also be included in the journalistic piece of a cricket match. This information should be injected to the system either as database values or from the runtime inputs from the user. These values cannot be taken from the scorecard itself. Some of the information that are being used in this cricket news generations are given below.

- When and where the match happened.
- Were there any injured players
- Whether the match was interrupted by the rain
- Whether Duckworth - Lewis - Stern method was used in the given match.

- Whether the winning team won this match after a long time

Templates of the system will be saved in an excel file and the situation where the templates to be used will be decided on the keys(annotation) given to each template. Each template may be based on one or more annotation. These keys denote the conditions and event related to the match summary. For the same set of annotations there can be more than one template. There are gaps in each template and each gap is given a variable name. Assigning a value to a variable will be done in the generator module. Through that variation of the output can be achieved. Some defined set of templates are shown below.

Table 4.1: Some set of templates

Condition	**Template**
MatchOverview	#(Teams.Inning1.Name) and #(Teams.Inning2.Name) played out a full match at #(Venue), on #(Date), with the #(Result.Won) emerging victors #(Description) in #(Series). #(Toss.Captain) won the toss and asked #(Teams.Inning1.Captain) #(Teams.Inning1.Name) to bat first.
MatchOverview	For #(Series), Team met at #(Venue) on #(Date) where #(Result.Loss) were beaten by #(Result.Won) comprehensively. After winning the toss #(Toss.Captain) invited #(Teams.Inning1.Name) to bat first.
Won, Century	#(Batsman.Name) pulled the match in #(Result.Won)'s favor by scoring a century of #(Batsman .Runs) runs #(Batsman.Out) from #(Batsman .Balls) balls with #(Batsman .Boundaries) boundaries
Won, MaxRuns/ Loss, MaxRuns	#(Batsman.Title) #(Batsman.Name) top-scored with #(Runs) runs off #(balls) balls which included #(Boundaries) boundaries
FirstInning, TopWickets	#(Bowler.Name) (#(Result.Won)) restricted #(Result.Loss) Inning to #(Loss.Runs) by taking #(Bowler.Wickets) wickets bowling #(Bowler.Overs) overs
Manofthematch	#(Ending.ManoftheMatch) became the player of the match.

Manofthematch	#(Ending.ManoftheMatch) was awarded as man of the match that's to his amazing innings

According to above templates, it can be clearly seen that these templates are fixed. Therefore, it is possible to extend this system to different types of languages. For that each template in English can be converted to any other language like Sinhala without changing the any variables defined within the template.

Chapter 5 : Generation System

introduced the research methodology, how the process defined would solve the research problem with tools and techniques will be discussed in this chapter. Then the processes defined in methodology will be elaborated more in detail here.

5.1 Overview

First phase of the pipeline which is the input converter is responsible for converting the input to a collection of key value pair. The output of this phase will be JSON formatted tree. The reason behind using a tree structure is that it is easy to order information in logical hierarchical manner. Unlike xml, Json can be easily manipulated using python and does not need XSLT for the transformation.

The json tree which is made out of the score card will also be manipulated with some background information such as player details. This formatted json output will be the real output to the generator module. A part of json output which is generated through the input module for a particular match is given below in figure 5.1. The full output is given in the appendix.


```
{
   "result":"India won by 7 wkts",
   "first_inning":{
      "Team":"England",
      "Score":"198-9",
      "Overs":"20"
"basic":{
      "Match":"Eng vs IND, 3rd T20I, India tour of England, 2018",
      "Date":"Sunday, July 08, 2018",
      "Toss":"India won the toss and opt to bowl",
      "Time":"01:00 PM GMT",
      "Venue":"County Ground, Bristol",
      "Umpires":"T Robinson, R Bailey",
      "Third Umpire":"Michael Gough",
      "Match Referee":"D Boon"
   }
}
```

Figure 5.1: Part of the json output for given match

Each player node is manipulated with some background information as below.

```
"Batsman4":{
            "Name":"Eoin Morgan",
            "Catch":"Dhoni",
            "Ball":"Hardik Pandya",
            "Out":"Out",
            "Runs":"6",
            "Balls":"9",
            "Fours":"0",
            "Sixs":"0"
            "Role": "Batsman",
            "More": "got out by playing a poor shot"
        }
```

Figure 5.2: Special annotations for a player node

Two tags such as “Type” and “More” are added to the player node as in figure 5.2 to describe more about the player within the match. If “More” tag is added to the any of the player node, the information about the player is also should be included in the journalistic piece as well. There are only specific number of special annotations that can be defined on the “More” tag. The “Role” tag is added to define playing role of a player whether he is a bowler, batsman or an allrounder. This will be useful when there is an extra ordinary performance by a player. For example, if a bowler scores a half century that should be included match summary.

If there is node as shown in the above figure 5.2, then it might be realized as below figure 5.3.

```
“Eoin Morgan got out for 6(9) trying to play a sluggish shot outside the off”
```

Figure 5.3: Corresponding output text

There are four tags each to describe the batting and bowling behavior while two tags to define catching actions.

However, this background information needed for the input module will be provided through a database. An SQLite database is chosen since it is very simple to handle, and very little amount of information will be saved in the database.

Typical pipeline architecture would contain preprocessing stage, but it is not defined in the system architecture defined. But preprocessing will also happen with the input module since it augments the player information and other background information to the main input. As stated above there are templates designed for basic background information about the team and matches. This related information will be taken by the user inputs. Some open-ended questions (yes-no questions) will be given to collect these data from the user.

5.2 Content Selection

The content selection module will be based on rule-based approach. According to some predefined rules the formatted input to the generator module will be manipulated and organized. The content selection is responsible for picking items that would only be needed for the realization. Other data will be ignored.

The formatted input contains all the information about every batsman and bowler, but all these data may not be needed for the realization. Therefore, choosing the batsmen and bowlers who have well performed or should be included in the final output will be selected by the content selection module based on some rules and fuzzy logic.

Some of the rules which are used for the content selection in batting perspective are given below.

- Players with maximum runs scored
- Players who have scored half century
- Players who have scored century
- Number of boundaries.
- Player with highest number of boundaries
- Runs scored by team captain if it over the threshold
- Runs played by the captain.

These threshold values will be hard coded in the system. When it comes to selection of bowling items there are another set of rules are defined. Some of them are

- Total number of wickets taken
- Players who took highest number of wickets
- Wickets taken by the captain
- Players who took wickets more than the threshold

There are several rules for the catching items as well. They are

- Player who has taken highest number of wickets
- Number of wickets taken by the wickets taken

Each rule is given weightage and this weightage will be used in text structuring stage to order the items accordingly to the weight with respect to each group i.e. batting, bowling and catching.

There are some other common rules in the content selection they are

- All the player nodes with the special annotation “More”.
- Background information with “Yes” value
- Overs finished by two teams

Some features like all out or all 20 overs are finished will be determined by the content selected by applying rules. For example, number of wickets lost by the team being processed is tested to determine whether the feature should be all out or number of overs.

After the content selection is done, in order to maintain disambiguity of player information, the duplicates will be removed as much as possible.

5.3 Text Structuring

The text structuring includes the ordering the data according to the several criteria. As mentioned earlier, the rules are defined with weights. For example, weightage is given as below for batting items in table 5.1

Table 5.1: Set of Rules

Rule	**Weight**
Players with maximum runs scored	9
Players who have scored half century	8

Players who have scored century	10
Number of boundaries	3
Player with highest number of boundaries	3
Runs scored by team captain if it over the threshold	4

According to weightage the items will be ordered in each batting, bowling and catching sections.

Since the news piece is basically divided in to four sections such as overview, first inning, second inning and conclusion, the items will be distributed as per the content. For example venue, date and time elements will be positioned in the introduction node. It is easy to select the items for the overview and conclusion since it is most of the time takes a fixed structure and content. But the two-team summary items need to be based on the content selected.

5.4 Aggregation

Aggregation is the phase which is accountable on decide on which items to aggregate together from the output taken form text structuring. This is the fourth stage of the data transformation. According to Kalina Bontcheva, values with the same property and domain it is able to perform sematic aggregation. Therefore, such values can be expressed within one sentence [35].

In this research context, it will try to aggregate the similar items based on the order and type. Since the items were ordered in prior modules according to the weightage, there is a possibility to aggregate the consecutive items accordingly and it is not compulsory to perform aggregation. In above what it is mentioned as types are ‘batting’, ‘bowling’ and ‘catching’. If the types are similar in consecutive items, they have the potential for aggregation. There are rules that conforms the aggregation. They are

- Items are in consecutive order.
- Items should be within same type or any type with common rules can be aggregated together.
- Items with the same special annotation “More” within the same inning
- Items come under the same rule defined in content selection
- More than three items cannot be aggregated together

For example, if there are two players who have players half century, within one sentence this can be mentioned. Another example would be if the captain node and the player with the highest batting score is the same, these two can be aggregated together in order to make a one sentence at the realization.

The output of this aggregation module will be forwarded to surface realization component.

5.5 Surface Realization

The surface realization is responsible to generate the output text in natural language from the internal representation format. Templates will be used within this module. There will be nearly 80 templates that would be used in here. In research design chapter, it was discussed how the templates were designed.

The keys in the templates will be mapped with the type of the news item. From that the suitable template for each situation will be selected. Since the variation of the output is one of the major objectives of the research, there can be multiple templates are provided for the same situation and system would select one template at a time randomly. After selecting one template for a given situation, assigning values to the variables based on the values of the internal representation will be done. This will be done by parsing through the given node to find the correct value to assign for the variable.

Below table 5.2 shows how few sentences are being realized from the sentence templates.

Table 5.2: Templates and Realized Output

Template	Realized Output
#(Teams.Inning1.Name) and #(Teams.Inning2.Name) played out a full match at #(Venue), on #(Date), with the #(Result.Won) emerging victors #(Description) in #(Series). #(Toss.Captain) won the toss and asked	India and England played out a full match at County Ground, Bristol, on Sunday, July 08, with the India emerging victors by 7 wickets in 3rd T20I, India tour of England, 2018. Virat Kohli won the toss and asked Eoin Morgan England to bat first.

#(Teams.Inning1.Captain) #(Teams.Inning1.Name) to bat first.	
For #(Series), Team met at #(Venue) on #(Date) where #(Result.Loss) were beaten by #(Result.Won) comprehensively. After winning the toss #(Toss.Captain) invited #(Teams.Inning1.Name) to bat first.	For the 3rd T20I, India tour of England, 2018, Team met at County Ground, Bristol on Sunday, July 08 where England were beaten by India Comprehensively. After winning the toss Virat Kholi invited England to bat first
#(Batsman.Name) pulled the match in #(Result.Won)'s favor by scoring a century of #(Batsman .Runs) runs #(Batsman.Out) from #(Batsman .Balls) balls with #(Batsman .Boundaries) boundaries	Rohit Sharma pulled the match in India's favour by scoring a century of 100 runs unbeated from 56 balls with 11 boundaries.
#(Batsman.Title) #(Batsman.Name) top-scored with #(Runs) runs off #(balls) balls which included #(Boundaries) boundaries	Kusal Perera top-scored with 78 runs off 81 balls which included eight boundaries
#(Bowler.Name) (#(Result.Won)) restricted #(Result.Loss) Inning to #(Loss.Runs) by taking #(Bowler.Wickets) wickets bowling #(Bowler.Overs) overs	Hardik Pandya (India) restricted England Inning to 198 by taking 4 wickets bowling 4 overs.
#(Ending.ManoftheMatch) became the player of the match.	Rohit Sharma (India) became the player of the match
#(Ending.ManoftheMatch) was awarded as man of the match that’s to his amazing innings	Kusal Janith was awarded as man of the match that’s to his amazing innings

5.6 Final Output Generation

As it is mentioned above minor validations of the final output will be done here. This is the final stage of the pipeline. Sentence capitalization and numeric values validations will be done here.

Some inbuilt functions in NLTK (string manipulation) will be used here in respect to the requirement.

Chapter 6 : Evaluation and Results

Generally, NLG evaluation is marked by great deal of variety and it is difficult to compare NLG systems directly. There is no straight forward method or standardized matrix that evaluates this metric. According to Reiter and Dale [2], the success or failure of the NLG system cannot be predicted using the results of a quality evaluation. NLG systems are usually evaluated using user acceptance test. But it is time consuming and there can be factors which would influence the comparison that cannot be considered in Natural Language Generation. However, this proposed system is implemented based on a component architecture. The Glassbox evaluation can be used to measure the performance of each module. Therefore, glassbox evaluations will be performed as unit test after implementing each module to ensure if the needed results are generated as desired. Black box testing is used to test the final output at the end.

There are two common approaches in evaluation methods of NLG system i.e. Intrinsic Evaluation and Extrinsic Evaluation [1]. The intrinsic evaluation assesses properties of the systems in their own right while extrinsic evaluation assesses the effect of the system on something that is external to it.

There are three basic intrinsic techniques which can be one out of three when it comes to evaluation.

1. Assessment by trained assessors of the quality of the system outputs according to different quality criteria, typically using rating scales.

2. Automatic measurements of the degree of similarity between system outputs and reference outputs.

E.g.: BLEU and ROOUGE [30]

3. Human assessment of the degree of similarity between system outputs and reference outputs.

For the evaluation of the proposed systems, automatic measurements like BLEU or ROUGH will not be used since word by word/phrase by phrase evaluations will not be suitable for a NLG system that would use sentence templates. One of the main objectives of the system is to have variations

in the output text. Therefore, it is not feasible to compare the human authored text with the generated output for an exact match.

Here there will be a customized method of evaluation which would compare the system generated text with two reference output texts. One summary would be an expert written while the other one be an average written summary. The summaries are compared using two parameters such as Nouns mentioned, and Events mentioned. The ultimate goal of the evaluation is to achieve the humanness parameter level. The humanness parameter level will be calculated based on the comparison. For the comparison, three types of scores are calculated.

1. **Similarity Score** – Ratio of number of nouns present in generated output and one of the reference outputs.

2. **Degree of Closeness** – The value taken by comparing the generated output text with reference output text under 2 parameters.

- Length of the text
- No of similar events (Manual)

3. **Data Count** – This value represents richness in data as a comparison.

It is not efficient to compare the generated summary with the human authored summary since the system does not have the access to most of the background information even though some of them are injected to the system at different modules. For an example strength of a partnership in match cannot be demonstrated through a generated output which uses only the score as the main input. Therefore, direct comparison may not be an accurate measurement to judge the performance of the system. This is the reason why three type of scores will be used for the evaluation.

As mentioned above there will be two reference text pieces and the expert written summary would be taken from cricinfo while the other one would be taken from ‘daily mirror’ local newspaper. There are two major reasons behind using two reference text pieces. Expert written summary comparison would help to check the quality of the content in respect to information included while average written summary would help to check if the Sri Lankan style of news reporting is achieved.

6.1 Evaluation Result

As it is mentioned earlier, the evaluation will be done based on 3 scores. To calculate these 3 scores there will be automatic and manual way of calculation.

6.1.1 Similarity score

An automatic calculation method is used here to represent the ratio of nouns of reference text to generated text. NLTK library functions (ie. Tokenizing, Pos-tagging) will be used to identify the nouns in both the text files.

This similarity score will be done twice for both the reference text files such as average written summary and expert summary. One such comparison is given below. The generate summary in the figure 6.1, the average written in the figure 6.2 and expert written summary in 6.3 are shown.

India and England met at County Ground, Bristol on Sunday, July 08 in 3rd T20I, India tour of England, 2018 India tour of Ireland and England 2018 and India defeated England by seven wickets. Virat Kohli won the toss and asked Eoin Morgan England to bat first.

Jason Roy looked in superb touch for his 67 from 31 balls, which included seven sixes and four fours. Jos Buttler made 34 off 21 balls with 7 boundaries. Alex Hales scored 30 in 24 shot. Captain Eoin Morgan got out playing a poor shot. Hardik Pandya (India) restricted England Inning to 198 by taking 4 wickets bowling 4 overs.

Rohit Sharma pulled the match in India's favour by scoring a century of 100 runs unbeated from 56 balls with 11 boundaries and 5 sixes. India Captain Virat Kohli scored blazing 43 runs off 29 balls before he was got out for bowl by Chris Jordan.

Rohit Sharma (India) became the player of the match. India won the three match series securing 3-0 lead.

Figure 6.1: Generated Summary

India beat England in the series decider in Bristol by seven wickets to clinch the T20I leg of the tour 2-1. Rohit Sharma, Hardik Pandya and MS Dhoni were among the chief contributors, across departments, as India chased down a record target of 199 to register their sixth series win on the trot.

Brief scores: England 198/9 in 20 overs (Jason Roy 67, Jos Buttler 34; Hardik Pandya 4-38, Siddarth Kaul 2-35) lost to India 201/3 in 18.4 overs (Rohit Sharma 100*, Virat Kohli 43, Hardik Pandya 33*) by seven wickets.

Figure 6.2: Average Written Summary [36]

Rohit Sharma's magnificent 100* leads India to series win

A long tour of England can bring contingencies, and India responded gloriously to the first of those to maintain their clean sheet in T20I bilateral series of three matches. Central to India's win was a man who might yet have to play a big part in the rest of the summer: allrounder Hardik Pandya stepped up with India missing two of their first-choice quicks to injury and dropping Kuldeep Yadav, who had taken five wickets two matches ago, because of the small straight boundaries.

Pandya bowled smartly, not giving England anything to drive and changing his pace often in his personal best analysis of 4 for 38, reducing a marauding England batting to 198 when they had looked good for 225. He was there with centurion Rohit Sharma when India needed to pull themselves out of a brief slowdown when the 15th and 16th over produced seven runs and the wicket of Virat Kohli. Pandya's 33 off 14 took off any pressure that might have been on his Mumbai Indians' team-mate as India chased the target down with more than an over to spare.

Figure 6.3: Expert Written Summary [37]

The similarity score comparing generated text with the average written summary for the above match was 33.33. At the same time when the generated text is compared with the above expert summary was 64.44. This is calculated in the following manner.

$$\frac{\text{(No of nouns identified in the reference text) – (No of nouns identified in the generated text)}}{\text{(No. of nouns identified in the generated text)}}$$

This evaluation was done for three matches and the results are given in figure 6.4.

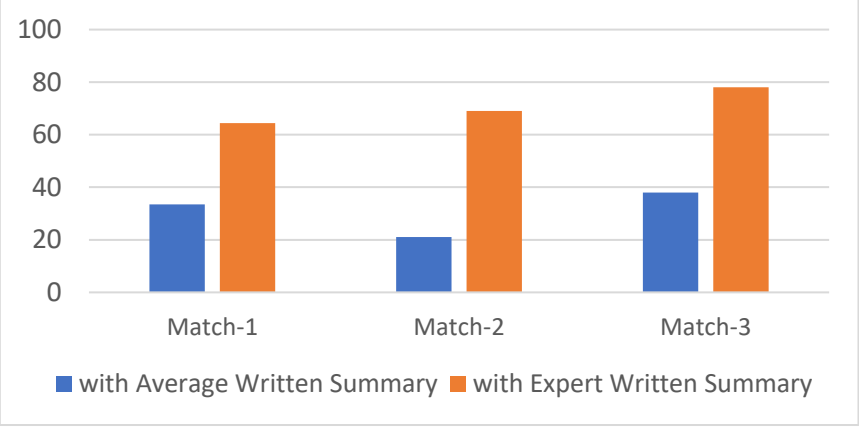

Figure 6.4: Similarity score evaluation for 3 matches

According to results, it is not fair to expect the same pattern of results for every match since the different nouns like synonyms can be used to demonstrate the same situation. However since the templates are being used to generate the text and these templates are created using the news articles found in Daily mirror and other Sri Lankan cricket news articles, the similarity score is comparatively low when the generated text is compared with average written summary than comparing it with expert written summary.

6.1.2 Data count

This evaluation method would compare the generated text with the reference text to assess the richness in data. The data count will be calculated based on proper nouns. For this calculation, again the NLTK library functions are used. In most of the cases, count in the generated text summary is higher than the average written summary. For 3 set of matches count will be given below table 6.1.

Table 6.1: Data Count Result

Match	Generated text	Average written summary	Expert written summary
Match 1	27	14	18
Match 2	33	37	66
Match 3	16	15	48

According the above results for the given three matches, richness in data is in a satisfactory level since the generated text has comparatively similar or higher value for number of proper nouns within the summary. But when it is compared with the expert written summary, it not easy to find a pattern with 3 matches.

6.1.3 Degree of closeness

This is calculated based on two different parameters as mentioned above. Length of the text is easily calculated using an automated method by using inbuilt functions in python and the number of similar events is calculated manually. Similar events will change from one match to another. Same event may be written using different vocabulary. Therefore, manual evaluation is suitable for

that. Examples of the events taken into consideration are “Took a hattrick”, “poor shot by”, “DLS”, “won the toss”, “agreed to bat first” etc.

Length of the generated texts is compared with the other two reference texts and the statistics for this evaluation in 3 matches is given in the table 6.2.

Table 6.2: Length comparison in generated and reference texts

Length of the generated text	Length of the average written summary	Length of the expert written summary
941	513	1090
1121	1857	5053
980	1643	4333

Number of events showed different result in each case based on information available in table 6.3 and 6.4. Some of the reasons why the number of events in generated text was less due to the lack of background information. However, some matches show excess information of the results. For example, it may not need to include the number of runs scored over 30 when there are 2 half centuries played by two batsmen. But the generated text includes such information as well due to hard coded rules. Therefore, there is a vague pattern when it comes to number of events.

Table 6.3: No of event comparison with expert written summary

	No of events in generated text	No of events in expert written summary	No of similar events in generated text and average written summary	Percentage
Match 1	11	6	5	83%
Match 2	12	26	12	46%
Match 3	14	21	10	47%

Table 6.4: No of event comparison with average written summary

	No of events in generated text	**No of events in average written summary**	**No of similar events in generated text and average written summary**	**Percentage**
Match 1	11	10	7	70%
Match 2	12	7	7	100%
Match 3	14	9	9	100%

According to above three matches, the evaluation results does not show an exact pattern in the comparison between expert summary and generated summary. Number of similar events do not show the similar percentages in each case. However, the number of events generated by the system has a comparatively sufficient value in the most cases. At the same time this value is nearly equal or greater than the value taken from the average written summary. As mentioned above background information make a considerable amount of contribution for the number of events and it is lack in the generated summary than the human authored summaries. Since this research is on generating news in Sri Lankan style, the expectation of the system is achieved according to this evaluation. However, the percentage is calculated as in below.

$$= \frac{\text{No of similar events (when compare the generated text with the reference text)}}{\text{No of events in reference text}}$$

As stated in previous chapters, journalism is not just a description, it is an event driven story. The creativity is one of the main characteristics of a sports journalist [41] but it is not a main requirement in automated journalism as described in bibliographic review. Through background information it finds a way to engage readers with the lead and thereby the creativity is achieved. During the evaluation process it is identified that the creativity is lack in generated text, but the information delivered is rich in data about the current match than one of the reference texts.

Chapter 7 : Conclusion and Future Work

7.1 Conclusion

Main purpose of the thesis was to generate an automated cricket journalistic piece. In Sri Lanka, automated journalism is not in use at the moment and it had not been touched by the scholars in their research. Therefore, significant amount of time and effort is being used for the sports journalism since it needs the engagement of language and domain specialists. Most of the time Sri Lankan cricket news is just a summary of a match without any expert commentary. However, if this process is automated, a lot of working time of journalists could be saved and they can focus in-depth reporting with expert commentary if necessary.

This dissertation presented a template-based system which can generate cricket news from a structure format of text data in Sri Lankan style. In this thesis, it is investigated the suitability of templated based approach for generating the cricket news. In cases of languages other than English, the template-driven approach to narrative generation would be particularly useful since there are not enough programming libraries and packages to build the system. In this system, there are two levels of templates defined i.e. sentence template and phrase template.

After reviewing the literature, this system was implemented based on the typical NLG pipeline architecture. Basically, this has two major modules such as input module and generation module. The generation module is based on the pipeline architecture with the components such as content selection, text structing, aggregation, surface realization and final output generation.

Finally, it was noted that the evaluation of an NLG system is challenging due to lack of standardized methods. Therefore, we have come up with the evaluation plan which is suitable for this kind of system. According to the evaluation plan there are three scores which are to be calculated to compare actual output of the system with the reference text taken from two websites. They are similarity score, degree of closeness and data Count. According evaluation results, the similarity score gives a favorable result when the generate summary was compared with the average written summary. Therefore, it can be concluded that use of templates has a major contribution for getting low percentages for similarity score. Similarly, satisfactory values could be gained for data count and degree of similarity when compared with average written summary. However, evaluating generated summary with expert summary was not successful since with a

smaller number of matches, it was not possible to understand a pattern. Therefore, it is better to have human based evaluation to compare the generated summary with an expert written summary.

7.2 Future Work

There are some future research directions in respect to this automated cricket news generating system. Having said that the system is capable of extending it to different languages because of the usage of templates, a Sinhala news generation is also practically possible.

Referring expression generation is one of the NLG tasks where the system applies rules on the input data to determine appropriate places where phrases such as he, she, they, it can be used to replace the corresponding proper nouns.

Content selection is done based on rule-based approach and it also can be done with a learning method if a sizable input is given. In this system cricket commentary is not in use for the cricket news generation and in future it is possible to make use of cricket commentary for input generate in detailed cricket news.

News heading can be generated in future. At the moment it was done and since it must contain more creativity. For this a learning method can be followed.

Chapter 8 : References

[1] K. Staykova, "Natural Language Generation and Semantic Technologies", Cybernetics and Information Technologies, vol. 14, no. 2, pp. 3-23, 2014.

[2] Graefe, A. (2016). Guide to automated journalism.

[3] "ESPNcricinfo - Cricket Teams, Scores, Stats, News, Fixtures, Results, Tables", ESPNcricinfo, 2018. [Online]. Available: http://www.cricinfo.com/. [Accessed: 20- Jul- 2018].

[4] K. Deemter, M. Theune and E. Krahmer, "Real versus Template-Based Natural Language Generation: A False Opposition?", Computational Linguistics, vol. 31, no. 1, pp. 15-24, 2005.

[5] C. Mellish, D. Scott, L. Cahill, D. Paiva, R. Evans and M. Reape, "A Reference Architecture for Natural Language Generation Systems", Natural Language Engineering, vol. 12, no. 01, p. 1, 2006.

[6] [G. Godbole, "Pipeline Architecture for Natural Language Generation", 2016.

[7] E. Reiter and R. Dale, "Building applied natural language generation systems", Natural Language Engineering, vol. 3, no. 1, pp. 57-87, 1997.

[8] A. Gatt and E. Krahmer, "Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation", Journal of Artificial Intelligence Research, vol. 61, pp. 65-170, 2018.

[9] J. Aires, "Automatic Generation of Sports News", Postgraduate, University of Porto, 2016.

[10] D. Radev and K. McKeown, "Generating Natural Language Summaries from Multiple On-Line Sources", Association for Computational Linguistics, vol. 24, no. 3, 1998.

[11] E. Reiter, S. Sripada and R. Robertson, "Acquiring Correct Knowledge for Natural Language Generation", Journal of Artificial Intelligence Research, vol. 18, pp. 491-516, 2003.

[12] J. Coch, "Overview of AlethGen", Eighth International Natural Language Generation Workshop (Posters and Demonstrations), pp. 25-28, 1996.

[13] G. Godbole, "Natural Language Generation", International Journal of Engineering Research and General Science, vol. 3, no. 2, 2015.

[14] G. Wilcock, "Pipelines, Templates and Transformations: XML for Natural Language Generation", Proceedings of the Workshop on Human Language Technology for the Semantic Web, of the 2nd International Semantic Web Conference, 2003.

[15] K. Deemter, M. Theune and E. Krahmer, "Real versus Template-Based Natural Language Generation: A False Opposition?", Computational Linguistics, vol. 31, no. 1, pp. 15-24, 2005.

[16] S. Busemann and H. Horacek, "A Flexible Shallow Approach to Text Generation", 9th International Workshop on Natural Language Generation, Niagara-on-the-Lake, Canada, pp. 238-247, 1998.

[17] M. Cristiá and B. Plüss, "Generating Natural Language Descriptions of Z Test Cases", INLG '10 Proceedings of the 6th International Natural Language Generation Conference, pp. 173-177, 2010.

[18] K. Deemtery, E. Krahmerz and M. Theunez, "Plan-based vs. template-based NLG: a false opposition?", 2001.

[19] H. Stenzhorn, "XtraGen — A Natural Language Generation System Using XML- and Java-Technologies", 2002.

[20] M. White and T. Caldwell, "Exemplars: Practical, Extensible Framework for Dynamic Text Generation", In Proceedings of the Ninth International Workshop on Natural Language Generation, 1998.

[21] W. Oremus, "The First News Report on the L.A. Earthquake Was Written by a Robot", https://goo.gl/jeHRPX., 2014.

[22] L. Leppänen, M. Munezero, M. Granroth-Wilding and H. Toivonen, "Data-Driven News Generation for Automated Journalism", Proceedings of the 10th International Conference on Natural Language Generation, pp. 188–197, 2018.

[23] L. Kolodny, "AP Sports is using “robot” reporters to cover Minor League Baseball", TechCrunch, 2018. [Online]. Available: https://techcrunch.com/2016/07/03/ap-sports-is-using-robot-reporters-to-cover-minor-league-baseball/. [Accessed: 07- Nov- 2018].

[24] M. Theune, E. Klabbers and J. Odijk, "From Data to Speech: A General Approach", Natural Language Engineering, vol. 7, pp. 47-86, 2000.

[25] E. Goldberg, N. Driedger, and R. I. Kittredge. Using natural-language processing to produce weather forecasts. IEEE Expert: Intelligent Systems and Their Applications, 9(2):45–53, April 1994.

[26] F. M. Hasan, "Automatic generation of multilingual sports summaries", Postgraduate, Applied Science: School of Computing Science, 2011.

[27] J. Jai Hari Raju, P. Indhu Reka, K. K Nandavi and M. Karky, "Tamil Summary Generation for a Cricket Match", Tamil Computing Lab, College of Engineering Guindy, Anna University, 2011.

[28] "India defeat SL in T20 match", Dailymirror.lk, 2018. [Online]. Available: http://www.dailymirror.lk/mobile/article/india-defeat-sl-in-t20-match-20927.html. [Accessed: 30- Nov- 2018].

[29] A. Belz and E. Reiter, Comparing Automatic and Human Evaluation of NLG Systems, 1st ed. Trento, Italy: 11st Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference, 2006, pp. 3-7.

[30] E. Reiter, "How to do an NLG Evaluation: Metrics", Ehud Reiter's Blog, 2018.

[31] E. Loper and S. Bird, "NLTK: The Natural Language Toolkit", 2002.

[32] A. Gatt and E. Reiter, "SimpleNLG: A realisation engine for practical applications", Proceedings of the 12th European Workshop on Natural Language Generation, ENLG 2009, pp. 90-93, 2009.

[33] I. Androutsopoulos, G. Lampouras and D. Galanis, "Generating Natural Language Descriptions from OWL Ontologies: the NaturalOWL System", Journal of Artificial Intelligence Research, vol. 48, pp. 671-715, 2013.

[34] "Welcome to PyNLPl's documentation! — PyNLPl 1.2.8 documentation", Pynlpl.readthedocs.io, 2019. [Online]. Available: https://pynlpl.readthedocs.io/en/latest/. [Accessed: 01- May- 2019].

[35] K. Bontcheva, "Generating Tailored Textual Summaries from Ontologies", In 2nd European Semantic Web Conference, ESWC 2005, vol. 3532, pp. 241–256, 2019

[36] "Video: India beat England", Dailymirror.lk, 2019. [Online]. Available: http://www.dailymirror.lk/ 4553448 /tech. [Accessed: 04- May- 2019].

[37] "Recent Match Report - England vs India 3rd T20I 2018 | ESPNcricinfo.com", Espncricinfo.com, 2019. [Online]. Available: http://www.espncricinfo.com/series/18018/report/1119545/england-vs-india-3rd-t20i-ind-eng-2018. [Accessed: 04- May- 2019].

[38] J. Robin and K. McKeown, "Empirically designing and evaluating a new revision-based model for summary generation", Artificial Intelligence, vol. 85, no. 1-2, pp. 135-179, 1996. Available: 10.1016/0004-3702(95)00125-5.

[39] K. Staykova, "Natural Language Generation and Semantic Technologies", Cybernetics and Information Technologies, vol. 14, no. 2, pp. 3-23, 2014. Available: 10.2478/cait-2014-0015.

[40] Y. Tagawa and K. Shimada, "Comparison of Template- and Neural-based Methods for Sports Summary Generation", Journal of Natural Language Processing, vol. 25, no. 4, pp. 357-391, 2018.

[41] "Characteristics of Good Sports Reporters", Career Trend, 2019. [Online]. Available: https://careertrend.com/info-8564802-characteristics-good-sports-reporters.html. [Accessed: 02- Jun- 2019].

Chapter 9 : Appendix

The Output JSON File from the Input Converter Module


```
{
   "result":"India won by 7 wkts",
   "first_inning":{
      "Team":"England",
      "Score":"198-9",
      "Overs":"20",
      "Captain":"Eoin Morgan",
      "Players":{
         "Batsman1":{
            "Name":"Jason Roy",
            "Catch":"Dhoni",
            "Ball":"D Chahar",
            "Out":"Out",
            "Country":"England",
            "Runs":"67",
            "Balls":"31",
            "Fours":"4",
            "Sixs":"7",
            "Boundaries":11
         },
         "Batsman2":{
            "Name":"Jos Buttler",
            "Ball":"S Kaul",
            "Out":"Out",
            "Country":"England",
            "Runs":"34",
            "Balls":"21",
            "Fours":"7",
            "Sixs":"0",
            "Boundaries":7
         },
         "Batsman3":{
            "Name":"Alex Hales",
            "Catch":"Dhoni",
            "Ball":"Hardik Pandya",
            "Out":"Out",
            "Country":"England",
            "Runs":"30",
            "Balls":"24",
            "Fours":"3",
            "Sixs":"2",
            "Boundaries":5
         },
         "Batsman4":{
            "Name":"Eoin Morgan",
            "Captain":"Yes",
            "Catch":"Dhoni",
            "Ball":"Hardik Pandya",
            "Out":"Out",
            "Country":"England",
            "Runs":"6",
            "Balls":"9",
            "Fours":"0",
            "Sixs":"0",
            "Boundaries":0
         },
         "Batsman5":{
```

```
      "Name":"Ben Stokes",
      "Catch":"Kohli",
      "Ball":"Hardik Pandya",
      "Out":"Out",
      "Country":"England",
      "Runs":"14",
      "Balls":"10",
      "Fours":"2",
      "Sixs":"0",
      "Boundaries":2
   },
   "Batsman6":{
      "Name":"Jonny Bairstow",
      "Catch":"Dhoni",
      "Ball":"Hardik Pandya",
      "Out":"Out",
      "Country":"England",
      "Runs":"25",
      "Balls":"14",
      "Fours":"2",
      "Sixs":"2",
      "Boundaries":4
   },
   "Batsman7":{
      "Name":"David Willey",
      "Ball":"Umesh",
      "Out":"Out",
      "Country":"England",
      "Runs":"1",
      "Balls":"2",
      "Fours":"0",
      "Sixs":"0",
      "Boundaries":0
   },
   "Batsman8":{
      "Name":"Chris Jordan",
      "Out":"RunOut",
      "Country":"England",
      "Runs":"3",
      "Balls":"3",
      "Fours":"0",
      "Sixs":"0",
      "Boundaries":0
   },
   "Batsman9":{
      "Name":"Liam Plunkett",
      "Catch":"Dhoni",
      "Ball":"S Kaul",
      "Out":"Out",
      "Country":"England",
      "Runs":"9",
      "Balls":"4",
      "Fours":"0",
      "Sixs":"1",
      "Boundaries":1
   },
   "Batsman10":{
      "Name":"Adil Rashid",
      "Out":"NotOut",
      "Country":"England",
      "Runs":"4",
      "Balls":"3",
      "Fours":"1",
      "Sixs":"0",
      "Boundaries":1
   }
```

```
    },
    "Ballers":{
       "Baller1":{
          "Name":"Deepak Chahar",
          "Overs":"4",
          "Runs":"43",
          "Wickets":"1",
          "Country":"India"
       },
       "Baller2":{
          "Name":"Umesh Yadav",
          "Overs":"4",
          "Runs":"48",
          "Wickets":"1",
          "Country":"India"
       },
       "Baller3":{
          "Name":"Siddarth Kaul",
          "Overs":"4",
          "Runs":"35",
          "Wickets":"2",
          "Country":"India"
       },
       "Baller4":{
          "Name":"Hardik Pandya",
          "Overs":"4",
          "Runs":"38",
          "Wickets":"4",
          "Country":"India"
       },
       "Baller5":{
          "Name":"Yuzvendra Chahal",
          "Overs":"4",
          "Runs":"30",
          "Wickets":"0",
          "Country":"India"
       }
    }
 },
 "second_inning":{
    "Team":"India",
    "Score":"201-3",
    "Overs":"18.4",
    "Captain":"Virat Kohli",
    "Players":{
       "Batsman1":{
          "Name":"Rohit Sharma",
          "Out":"NotOut",
          "Country":"India",
          "Runs":"100",
          "Balls":"56",
          "Fours":"11",
          "Sixs":"5",
          "Boundaries":16
       },
       "Batsman2":{
          "Name":"Shikhar Dhawan",
          "Catch":"J",
          "Ball":"Willey",
          "Out":"Out",
          "Country":"India",
          "Runs":"5",
          "Balls":"3",
          "Fours":"1",
          "Sixs":"0",
          "Boundaries":1
```

```
    },
    "Batsman3":{
       "Name":"Lokesh Rahul",
       "Catch":"Chris",
       "Ball":"J Ball",
       "Out":"Out",
       "Country":"India",
       "Runs":"19",
       "Balls":"10",
       "Fours":"1",
       "Sixs":"2",
       "Boundaries":3
    },
    "Batsman4":{
       "Name":"Virat Kohli",
       "Catch":"&",
       "Ball":"Chris Jordan",
       "Out":"Out",
       "Country":"India",
       "Runs":"43",
       "Balls":"29",
       "Fours":"2",
       "Sixs":"2",
       "Boundaries":4
    },
    "Batsman5":{
       "Name":"Hardik Pandya",
       "Out":"NotOut",
       "Country":"India",
       "Runs":"33",
       "Balls":"14",
       "Fours":"4",
       "Sixs":"2",
       "Boundaries":6
    }
 },
 "Ballers":{
    "Baller1":{
       "Name":"David Willey",
       "Overs":"3",
       "Runs":"37",
       "Wickets":"1",
       "Country":"England"
    },
    "Baller2":{
       "Name":"Jake Ball",
       "Overs":"3",
       "Runs":"39",
       "Wickets":"1",
       "Country":"England"
    },
    "Baller3":{
       "Name":"Chris Jordan",
       "Overs":"3.4",
       "Runs":"40",
       "Wickets":"1",
       "Country":"England"
    },
    "Baller4":{
       "Name":"Liam Plunkett",
       "Overs":"3",
       "Runs":"42",
       "Wickets":"0",
       "Country":"England"
    },
    "Baller5":{
```

```
            "Name":"Ben Stokes",
            "Overs":"2",
            "Runs":"11",
            "Wickets":"0",
            "Country":"England"
         },
         "Baller6":{
            "Name":"Adil Rashid",
            "Overs":"4",
            "Runs":"32",
            "Wickets":"0",
            "Country":"England"
         }
      }
   },
   "basic":{
      "Match":"Eng vs IND, 3rd T20I, India tour of England, 2018",
      "Date":"Sunday, July 08, 2018",
      "Toss":"India won the toss and opt to bowl",
      "Player Of The Match":"Rohit Sharma",
      "Time":"01:00 PM GMT",
      "Venue":"County Ground, Bristol",
      "Umpires":"T Robinson, R Bailey",
      "Third Umpire":"Michael Gough",
      "Match Referee":"D Boon"
   }
}
```

Final Json output to match with the templates for a given match


```
{
   "MatchOverview":{
      "Teams":{
         "Inning1":{
            "Name":"England",
            "Captain":"Eoin Morgan"
         },
         "Inning2":{
            "Name":"India",
            "Captain":"Virat Kohli"
         }
      },
      "Result":{
         "Won":"India ",
         "Description":"by 7 wickets",
         "Loss":"England"
      },
      "Toss":{
         "Won":{
            "Name":"India",
            "Decision":"to bowl",
            "Captain":"Virat Kohli"
         },
         "Loss":{
            "Decision":"to bat",
            "Name":"England",
            "Captain":"Eoin Morgan"
         }
      },
      "MatchSeries":{
         "Series":"India tour of England, 2018",
         "Match":"3rd T20I"
      },
      "Date":"Sunday, July 08, 2018",
      "Venue":"County Ground, Bristol"
   },
   "first_inning":{
      "Batting":{
         "HalfCentury":{
            "Batsman1":{
               "Name":"Jason Roy",
               "Catch":"Dhoni",
               "Ball":"D Chahar",
               "Out":"Out",
               "Country":"England",
               "Runs":"67",
               "Balls":"31",
               "Fours":"4",
               "Sixs":"7",
               "Boundaries":11
            }
         },
         "Century":{
         },
         "MoreRuns":{
            "Batsman2":{
               "Name":"Jos Buttler",
               "Ball":"S Kaul",
               "Out":"Out",
               "Country":"England",
               "Runs":"34",
               "Balls":"21",
               "Fours":"7",
               "Sixs":"0",
```

```
            "Boundaries":7
         },
         "Batsman3":{
            "Name":"Alex Hales",
            "Catch":"Dhoni",
            "Ball":"Hardik Pandya",
            "Out":"Out",
            "Country":"England",
            "Runs":"30",
            "Balls":"24",
            "Fours":"3",
            "Sixs":"2",
            "Boundaries":5
         }
      },
      "GotOutPlayingPoor":{
         "Batsman4":{
            "Name":"Eoin Morgan",
            "Captain":"Yes",
            "Catch":"Dhoni",
            "Ball":"Hardik Pandya",
            "Out":"Out",
            "Country":"England",
            "Runs":"6",
            "Balls":"9",
            "Fours":"0",
            "Sixs":"0",
            "Boundaries":0,
            "More":"got out playing a poor shot",
            "Role":"batsman"
         }
      },
      "Max":{

      },
      "HighestBoundaries":{
         "Name":"Jason Roy",
         "Catch":"Dhoni",
         "Ball":"D Chahar",
         "Out":"Out",
         "Country":"England",
         "Runs":"67",
         "Balls":"31",
         "Fours":"4",
         "Sixs":"7",
         "Boundaries":11
      },
      "Captain":{
         "Name":"Eoin Morgan",
         "Captain":"Yes",
         "Catch":"Dhoni",
         "Ball":"Hardik Pandya",
         "Out":"Out",
         "Country":"England",
         "Runs":"6",
         "Balls":"9",
         "Fours":"0",
         "Sixs":"0",
         "Boundaries":0,
         "More":"got out playing a poor shot",
         "Role":"batsman"
      }
   },
   "Bowling":{
      "Captain":{
```

```
        },
        "MoreWickets":{
           "Baller3":{
              "Name":"Siddarth Kaul",
              "Overs":"4",
              "Runs":"35",
              "Wickets":"2",
              "Country":"India",
              "More":"bowled with consistent line and length"
           }
        },
        "More":{
           "Baller3":{
              "Name":"Siddarth Kaul",
              "Overs":"4",
              "Runs":"35",
              "Wickets":"2",
              "Country":"India",
              "More":"bowled with consistent line and length"
           }
        },
        "MaxWickets":{
           "Name":"Hardik Pandya",
           "Overs":"4",
           "Runs":"38",
           "Wickets":"4",
           "Country":"India"
        },
        "HighestRate":{
           "Name":"Hardik Pandya",
           "Overs":"4",
           "Runs":"38",
           "Wickets":"4",
           "Country":"India"
        }
     },
     "MaxWickets":{

     }
  },
  "second_inning":{
     "Batting":{
        "HalfCentury":{

        },
        "Century":{
           "Batsman1":{
              "Name":"Rohit Sharma",
              "Out":"NotOut",
              "Country":"India",
              "Runs":"100",
              "Balls":"56",
              "Fours":"11",
              "Sixs":"5",
              "Boundaries":16
           }
        },
        "More":{
        },
        "MoreRuns":{
           "Batsman4":{
              "Name":"Virat Kohli",
              "Catch":"&",
              "Ball":"Chris Jordan",
              "Out":"Out",
              "Country":"India",
```

```
            "Runs":"43",
            "Balls":"29",
            "Fours":"2",
            "Sixs":"2",
            "Boundaries":4
         },
         "Batsman5":{
            "Name":"Hardik Pandya",
            "Out":"NotOut",
            "Country":"India",
            "Runs":"33",
            "Balls":"14",
            "Fours":"4",
            "Sixs":"2",
            "Boundaries":6
         }
      },
      "GotOutPlayingPoor":{

      },
      "Max":{

      },
      "HighestBoundaries":{
         "Name":"Rohit Sharma",
         "Out":"NotOut",
         "Country":"India",
         "Runs":"100",
         "Balls":"56",
         "Fours":"11",
         "Sixs":"5",
         "Boundaries":16
      }
   },
   "Bowling":{
      "Captain":{

      },
      "More":{
         "Baller3":{
            "Name":"Chris Jordan",
            "Overs":"3.4",
            "Runs":"40",
            "Wickets":"1",
            "Country":"England",
            "More":"bowled with consistent line and length"
         }
      },
      "MoreWickets":{
      },
      "MaxWickets":{
         "Name":"David Willey",
         "Overs":"3",
         "Runs":"37",
         "Wickets":"1",
         "Country":"England"
      }
   },
   "MaxWickets":{

   }
},
"Ending":{
   "ManoftheMatch":"Rohit Sharma"
}
}
```