

**FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO**



# **Automatic Generation of Sports News**

**João Pinto Barbosa Machado Aires**

DISSERTATION

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. Sérgio Sobral Nunes

June 27, 2016



# Abstract

Natural Language Generation (NLG) is a task of Natural Language Processing (NLP) that, from some non-linguistic representation of information, is able to construct understandable text in natural language automatically. In this dissertation, we provide a bibliographical review of the NLG area with a particular focus on the applications of NLG in the area of Journalism. We describe the tasks that usually compose an NLG system as well as the methodologies and approaches applied to perform those tasks. We provide a list of the most relevant tools and resources in this field and discuss the evaluation methodologies performed when it comes the time to evaluate the quality of an NLG system.

We present the GameRecapper, a data-to-text template-based system that generates Portuguese summaries of football games from structured input data. GameRecapper has a basic generation algorithm for creating the news pieces and makes use of domain data, linguistic functions, grammatical functions and a collection of sentence templates. Domain data provides additional information about the teams in order to achieve more variation in the output text. Linguistic functions translate numerical data into words and grammatical functions ensure the coherence and concordance of the text. The collection of sentence templates was built manually from an initial corpus written by actual journalists from a newsroom. Each template contains open slots for variable information. These sentence templates were divided into groups according to goal events of the game and to the characteristics of the game. GameRecapper's ability of knowing the impact of a goal event allowed us to achieve a significant amount of variation on the generated summaries.

We discuss and present an evaluation methodology to evaluate and analyze GameRecapper. In the evaluation of GameRecapper, our focus was to evaluate the quality of the produced text and to compare how users perceive a GameRecapper summary versus a human-authored summary. The results showed that GameRecapper is able to produce a grammatically correct and easy to read summary, given the average scores of intelligibility and fluidity criteria on our output text evaluation. On our evaluation of GameRecapper vs human-authored summaries, results show that, even though our generated summaries are not ready to be published online, GameRecapper is able to produce a complete and accurate match summary.



# Resumo

A Geração de Linguagem Natural (GLN) é uma tarefa na área do Processamento de Linguagem Natural (PLN) que, a partir de representações de informação não linguísticas, tem como objetivo criar sistemas informáticos capazes de produzir texto automaticamente, em linguagem natural. Nesta dissertação é feita uma revisão bibliográfica da área de GLN, com um destaque particular nas suas aplicações na área do Jornalismo. São descritas as tarefas que normalmente compõem um sistema GLN, assim como as metodologias e abordagens aplicadas para executar essas tarefas. É fornecida uma lista das ferramentas e dos recursos mais relevantes nesta área e são discutidas as metodologias de avaliação utilizadas para avaliar a qualidade de um sistema GLN.

Nesta dissertação, é apresentado o GameRecapper, um sistema de dados-para-texto, baseado em modelos, que gera sumários em português de jogos de futebol, a partir de dados estruturados. O GameRecapper tem um algoritmo de geração para criar as notícias, que faz uso de dados de domínio, funções gramaticais, funções linguísticas e um conjunto de modelos de frase. Os dados de domínio fornecem dados adicionais sobre as equipas, de forma a atingir uma maior variação no texto gerado. As funções linguísticas traduzem dados numéricos em palavras e/ou expressões e as funções gramaticais garantem a coerência e concordância do texto. O conjunto de modelos de frase foi construído manualmente, a partir de um corpus escrito por jornalistas de uma redação, e divididos de acordo com as características dos golos e com as características do jogo. Cada modelo de frase tem espaços para preencher com informação variável. A capacidade do GameRecapper conhecer o impacto de um golo no resultado permite atingir um grau de variação considerável nos sumários gerados.

Também é discutida e apresentada uma metodologia de avaliação que foi usada para avaliar e analisar o GameRecapper. Os objetivos principais da avaliação do nosso sistema são avaliar a qualidade do texto produzido e comparar a perceção dos leitores ao lerem um sumário gerado pelo GameRecapper a um sumário gerado por um jornalista. Os resultados demonstram que o GameRecapper é capaz de produzir um texto gramaticalmente correto e fácil de ler, dado os resultados médios obtidos nos critérios de inteligibilidade e de fluidez. Quanto à avaliação dos sumários gerados pelo GameRecapper, e em contraste com as notícias geradas por humanos, os resultados demonstram que, apesar das notícias ainda não estarem preparadas para serem lançadas online, o GameRecapper é capaz de produzir um sumário preciso e correto do jogo.



# Acknowledgments

First of all, I would like to express my gratitude to my supervisor Prof. Sérgio Nunes for all his support. His guidance was crucial on the development and on the writing of this thesis.

Besides my supervisor, I would like to thank all the team of the *www.zerozero.pt* newsroom for their availability and sympathy.

I would also like to thank all my colleagues and friends for all the sleepless nights, whether it was working together or having fun. I am sure that many of the friendships I made during this 6 years, will persist throughout our lives.

Last but not least, I would like to thank my family, specially my parents for always believing in me and for their unconditional support.

João Pinto Barbosa Machado Aires





*“Live as if you were to die tomorrow.  
Learn as if you were to live forever.”*

Mohandas Karamchand Gandhi



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Context and Motivation . . . . .	1
1.2	Objectives . . . . .	2
1.3	Organization . . . . .	2
<b>2</b>	<b>Natural Language Generation</b>	<b>5</b>
2.1	Historical Review . . . . .	5
2.2	Classification of NLG Systems . . . . .	6
2.2.1	Classification According to the Input of the System . . . . .	6
2.2.2	Classification According to the Communicative Goal of the System . . . . .	7
2.3	Design of an NLG System . . . . .	8
2.3.1	Text Planning . . . . .	9
2.3.2	Sentence Planning . . . . .	10
2.3.3	Linguistic Realization . . . . .	13
2.4	NLG Generic Methods . . . . .	15
2.4.1	Knowledge-Based Methods . . . . .	16
2.4.2	Statistical Methods . . . . .	17
2.4.3	Hybrid Models . . . . .	18
2.4.4	Techniques Applied in the Different Stages . . . . .	18
2.5	Evaluation Methodologies . . . . .	21
2.6	NLG Tools . . . . .	23
2.6.1	Natural Language Toolkit (NLTK) . . . . .	23
2.6.2	RealPRO . . . . .	23
2.6.3	SimpleNLG . . . . .	24
2.6.4	PyNLPl . . . . .	24
2.6.5	NaturalOWL . . . . .	24
2.6.6	OpenCCG . . . . .	25
2.7	Resources: Scientific Conferences and Events . . . . .	26
2.8	Applications of NLG in the Area of Journalism . . . . .	27
2.8.1	GoalGetter . . . . .	28
2.8.2	Multilingual Cricket Summary Generator . . . . .	29
<b>3</b>	<b>The GameRecapper System</b>	<b>33</b>
3.1	Introduction . . . . .	33
3.2	Description of GameRecapper . . . . .	35
3.2.1	Document Plan and Templates . . . . .	37
3.2.2	Generation Module Algorithm . . . . .	40

<b>4</b>	<b>Evaluation of GameRecapper</b>	<b>49</b>
4.1	Methodology . . . . .	49
4.2	Results and Discussion . . . . .	53
4.2.1	Evaluation of GameRecapper’s Output Text . . . . .	53
4.2.2	GameRecapper Summary versus Human-Authored Summary . . . . .	57
<b>5</b>	<b>Conclusion</b>	<b>61</b>
5.1	Summary . . . . .	61
5.2	Future Work . . . . .	62
	<b>References</b>	<b>63</b>

# List of Figures

2.1	NLG system architecture and associated activities proposed by Reiter and Dale (Source: [1]) . . . . .	8
2.2	Input for the system SUMGEN-W. Daily weather report (Source: [2]). . . . .	9
2.3	Initial data from SUMGEN-W before preprocessing stage (Source: [2]). . . . .	10
2.4	Message providing the type of relation and the arguments of the sentence. . . . .	12
2.5	Instructions to generate the output (Source: [3]) . . . . .	15
2.6	Graph of a scene of a dog close to a doghouse. Graph-based algorithm for referring expression generation (Source: [4]). . . . .	21
2.7	System architecture of GoalGetter (Source:[5]) . . . . .	29
2.8	Output and English translation of the GoalGetter system (Source:[5]) . . . . .	30
2.9	System architecture of Multilingual Cricket Summary Generator (Source:[6]) . . . . .	31
3.1	Match information of <i>www.zerozero.pt</i> . . . . .	34
3.2	Architecture of GameRecapper. . . . .	35
3.3	Information provided by the JSON file of a player. . . . .	39
3.4	Procedure done by the Generation Module for each goal event. . . . .	42
3.5	Match information of Académica vs Benfica provided by <i>www.zerozero.pt</i> . . . . .	44
4.1	Image of the GameRecapper summaries vs human-authored summaries survey. . . . .	53
4.2	Results of text quality evaluation. . . . .	53
4.3	Intelligibility and Fluidity average score for each final result. . . . .	54
4.4	Intelligibility and Fluidity average score according to the number of goals scored in a game. . . . .	55
4.5	Intelligibility and Fluidity average score according to the final outcome of the game. . . . .	55
4.6	Intelligibility and fluidity average score according to the goal difference between the winning and losing teams. . . . .	56
4.7	Distribution of the average scores of the summaries according to the completeness and readiness criteria. . . . .	58
4.8	Impact of the length of a summary in the completeness and readiness criteria (GameRecapper summaries). . . . .	59
4.9	Impact of the length of a summary in the completeness and readiness criteria (Human-authored summaries). . . . .	60



# List of Tables

2.1	Additional information on the NLG Tools described in Subsection 2.6 . . . . .	25
3.1	Possible outcomes of a football match that have distinct sentence templates. . . .	37
3.2	Player stats and corresponding added weight to select the player with the most impact on the final score. . . . .	38
3.3	GameRecapper sentence templates. . . . .	41
3.4	Distinct news pieces that GameRecapper can generate for a specific final result. .	41
4.1	Distinct final results contained between the rounds 25 until 29 and how many times that final result occurred. . . . .	50
4.2	Meaning of each intelligibility rating (Source: [7]). . . . .	51
4.3	Meaning of each fluidity rating (Source: [7]). . . . .	51
4.4	Distribution of games between the surveys according to the final result . . . . .	52
4.5	Additional information on the results of text quality evaluation. . . . .	54
4.6	Scores of GameRecapper and human-authored summaries on completeness and readiness to be published online. . . . .	57
4.7	Average scores of GameRecapper and human-authored summaries . . . . .	57





# Abbreviations

D2T	Data-to-Text
FLM	Factored Language Model
LM	Language Models
MTT	Meaning-Text Theory
NLG	Natural Language Generation
NLP	Natural Language Processing
RST	Rhetorical Structure Theory
T2T	Text-to-Text
TAG	Tree-Adjoining Grammars



# Chapter 1

## Introduction

In the first chapter, the context and motivation that originated the interest for the study as well as the proposed objectives are presented.

### 1.1 Context and Motivation

Natural Language Generation (NLG) is a task of Natural Language Processing (NLP) that, from some non-linguistic representation of information, is able to construct understandable text in natural language automatically [1]. Over the last years, with the popularization of the internet and the explosion of social media, the number of data increased significantly which led to the need to store, manipulate and quickly analyze very big data collections. Computer systems use representations which are easy for them to manipulate, however, in many cases, some of these representations of information require a considerable amount of expertise to interpret. This means that there is a need for systems which can present such information in an understandable form to every user. Natural language technology can be used when the best presentation of the data is in natural language. An example is to generate textual weather forecasts from graphical weather maps representations [8]. Natural Language Generation systems can also be used to help in the creation of routine documents, also known as Authoring Aids systems [1]. Many professionals that do not see document production as their main responsibility, spend a lot of their working time producing documents. For example, as Ehud Reiter and Robert Dale wrote "[a] doctor, for example, may spend a significant part of her day writing referral letters, discharge summaries and other routine documents. Tools which help such people quickly produce good documents may considerably enhance both productivity and morale." [1].

NLG is becoming popular among researchers in the area of Computational Journalism (application of computation's concepts and techniques in journalism). As Arjen van Dalen said "[due] to commercial pressures and higher profit expectations, there is a broader trend in journalism to lower the variable costs involved in news production by using more short-term contracts, freelance work, outsourcing and impersonal relations between writers or low-paid news work in content farms." [9]. Nowadays, it is possible to create algorithms that generate news stories automatically based

on NLG techniques, without human interference. These "robot journalists" can produce thousands of articles with virtually no variable costs. Aspects like personality, analytic skills and creativity become more important, if professional routine tasks can be automated [10]. Sports reporting is described as ideal to apply NLG techniques due to the abundant availability of statistics and the formulaic templates and stock phrases which can be used for game reports [9].

The main purpose of this dissertation is to implement automatic generation of sports news pieces. Today, there is a need to report on a lot of football matches and, many of them, are played at the same time. A significant amount of human resources and working time would be needed for journalists to watch every match that they have to make a report. Due to the abundant availability of information that is stored in databases, journalists are able to make a report based just on that information. However, if this process was automated it would save a lot of working time of journalists. As a result, journalists would have more time for in-depth reporting. This dissertation had the collaboration of ZOS, Lda., creator of the project *zerozero.pt* which has one of the biggest football database of the world. This project was performed as part of a dissertation thesis of the Integrated Master in Electrical and Computers Engineering from the Faculty of Engineering of the University of Porto.

## 1.2 Objectives

The idea of this dissertation is to generate journalistic pieces using information from structured databases. Our focus is to produce a system able to make a coherent and well-written summary of a football match. Since our collaborators' project *zerozero.pt* has mainly a Portuguese audience, for this dissertation we will focus on generating Portuguese summaries.

We also intended to review the state of the art of Natural Language Generation in order to understand what tasks usually compose an NLG system and what methodologies are used to perform those tasks. Another main objective of this dissertation is to discuss the possibilities of evaluation methodologies of an NLG system so we can perform a meaningful evaluation since our generated news have to go through a validation process in order to analyze and document the results of the algorithm.

## 1.3 Organization

This dissertation is divided in five chapters. Chapter two presents a bibliographic review on Natural Language Generation. In the first section, an historical review is made where the evolution of the area since the first experiments until now is shown. Section two presents the classification of NLG systems adopted. Section three discusses the design and tasks of an NLG system. Section four demonstrates the most used generic methods implemented in NLG systems. Section five is focused on the current evaluation methodologies for the NLG applications. Section six lists tools used to approach the task of NLG. Section seven specifies the scientific conferences and events in NLG. Finally, Section 8 addresses the impact of NLG in the field of Journalism.

Chapter three presents the GameRecapper system. In the first section, we discuss why generating a news piece that makes a game summary is difficult. Section two discusses in detail the different aspects of the system such as the general architecture, the document plan, the sentence templates and the Generation Module algorithm.

Chapter four presents the methodology followed to evaluate GameRecapper and the corresponding results and Chapter five summarizes the present dissertation, provides our conclusions and presents future work perspectives in order to improve the quality of the GameRecapper system.



## Chapter 2

# Natural Language Generation

Chapter two presents a bibliographic review on Natural Language Generation. In the first section, an historical review is made where the evolution of the area since the first experiments until now is shown. Section two presents the classification of NLG systems adopted. Section three discusses the design and tasks of an NLG system. Section four demonstrates the most used generic methods implemented in NLG systems. Section five is focused on the current evaluation methodologies for the NLG applications. Section six lists tools used to approach the task of NLG. Section seven specifies the scientific conferences and events in NLG. Finally, Section 8 addresses the impact of NLG in the field of Journalism.

### 2.1 Historical Review

In the 1970, one of the main focuses in Natural Language Processing research was trying to identify user's opinions, objectives and plans in order to achieve a proactive and extended interaction between users and expert systems for consultation and command, where the system's responses should be collaborative [11]. This motivated an increase of research on discourse, especially dialogue, and on generation, mainly on multi-sentence text generation. These were connected because collaborative response, e.g. in advice giving systems, depends on modeling the user's opinions, goals and plans, and can naturally lead to paragraph-length outputs, for instance in providing explanations [12]. The NLG field has been in development ever since and it is not surprising that the first NLG systems which translated data into very simple texts with little or no variation started to appear, such as advice giving systems or synthesized weather forecasts [13, 14]. Over the years, the conceived NLG systems became even more complex, more linguistic insights were included and several methodologies were developed for generating more varied text.

Nowadays, we can state that NLG is a consolidated research field, given how many systems were implemented and the range of application-domains in which they were utilized [15]. There is an increasing need for text in natural language which handles all types of information. The methodologies and approaches used in the NLG field try to answer real life demands and that is why most of the NLG systems came up with a practical application. Some of these practical

applications include generating weather reports from meteorological data in multiple languages [8] or even generating custom letters to provide answers for users' questions [16]. The biggest drawback of NLG has to do with the lack of standardization of methodologies and approaches to build an NLG system. Usually the techniques used to develop an NLG system are determined by the application in hand (domain of application and communicative goal) and by the level of complexity and variation desired in the output text.

Major providers of natural language generation technology in the US, Automated Insights and Narrative Science, began by developing algorithms to automatically write recaps of baseball games. Sports served as an ideal starting point due to the availability of data, statistics, and predictive models that are able to, for example, continuously recalculate a team's chance of winning as a game progresses.

## 2.2 Classification of NLG Systems

A NLG system can be classified according to multiple criteria. We adopted the classification developed by M. Vicente et al. which concluded that there are two main elements to distinguish NLG systems [3]:

- The input of the system
- The communicative goal of the system

### 2.2.1 Classification According to the Input of the System

In NLG, it is possible to distinguish two types of systems depending on their input: data-to-text (D2T) systems and text-to-text (T2T) systems. While the input of D2T is a collection of data that does not make up a text (e.g. numerical data representing meteorological information), in T2T systems the output is obtained by extracting the key information from a text.

**Data-to-text** Usually, the input type of a D2T system is structured data whether it is numerical data, labeled corpus, databases, knowledge bases or log archives. Some authors use the word concept when referring to this type of non-linguistic representation of information, therefore these systems can also be named as concept-to-text. Some examples of D2T system are: SumTime [17], a system that generates weather forecast texts from numerical weather prediction data, and GoalGetter [5], which generates reports of football matches in Dutch.

**Text-to-text** The input data of T2T can be texts or isolated sentences. There are a lot of NLG applications that use T2T systems such as generating textual summaries or simplified texts. Barzilay and Sauper created a system that automatically generates Wikipedia articles from a collection of internet documents [18].



### 2.2.2 Classification According to the Communicative Goal of the System

NLG systems can also be distinguished according to the communicative goal of their construction. The most relevant are:

**Informative Texts** The purpose of the system is to generate informative texts from factual data. FoG [8] and SumTime [17] create weather forecasts taking as input numerical information from simulation systems that represent parameters like temperature, precipitation level and wind speed from different places and different hours of the day. Another example is SkillSum [19], a system that generates basic skills reports to help people with poor basic numeracy and literacy skills.

**Textual Summaries** This type of systems produce textual summaries from one or more data sources. This summaries can be associated to different fields: medical summaries [20], engineering [21], financial [22], sports [23], patents [24], among others.

**Simplified Texts** Systems that aim to help people with oral or writing problems, derived from cognitive difficulties or language barriers. Some examples are: systems that produce text to help aphasic people [25] or systems that allow visually impaired people to examine graphics [26].

**Persuasive Texts** Systems that try to persuade or take advantage of the user emotional state. Examples of these systems are: STOP [27], a system that generates tailored smoking cessation letters; systems that aim to decrease anxiety from cancer's patients by giving them personalized health information [28].

**Dialogue Systems** Dialogue system's main purpose is to improve human-machine communication. Users interact directly with the system that creates sentences conditioned by the previous context. There are lots of applications such as: automatic question generator system for academic writing support [29] or adaptable tutorial dialogue system to improve knowledge on certain subjects [30].

**Explanations** The output of this system is an explanation of the steps that the system went through to execute an algorithm, process a transaction or solve a mathematical problem. P.Rex [31] is an example of natural language proof explanation system.

**Recommendations** Systems which create recommendations by processing information related to users' preferences and opinions. Shed [32] is an online diet counseling system that provides personalized diets based on the user profile.

## 2.3 Design of an NLG System

In the previous section, we distinguished NLG systems according to their input data and their communicative goal. We can describe an NLG system as a group of tasks that transmit information to a certain audience to achieve a specific goal, in natural language. Therefore, characterizing the input, the tasks and the output of the system is as important as specifying its context and communicative goal. In order to the system successfully achieve its purpose each task should overview those aspects.

The design of NLG systems is an open field where a broad consensus does not exist [15]. There is a diversity of architectures and implementations which depend on the problem for which the NLG system is created. It's hard to identify common elements and to provide a complete abstraction which is applicable to most NLG systems. However, a lot of effort has been made in trying to define a general architecture for the context of NLG. As our interest is to have a starting point that help us to determine what tasks compose an NLG system in order to build a system that generates summaries of football matches, after reviewing the literature we came to the conclusion that the most consensual architecture is the one proposed by Ehud Reiter and Robert Dale. According to them, an NLG system performs seven tasks and the interaction between them can be represented by a three module architecture (Figure 2.1):

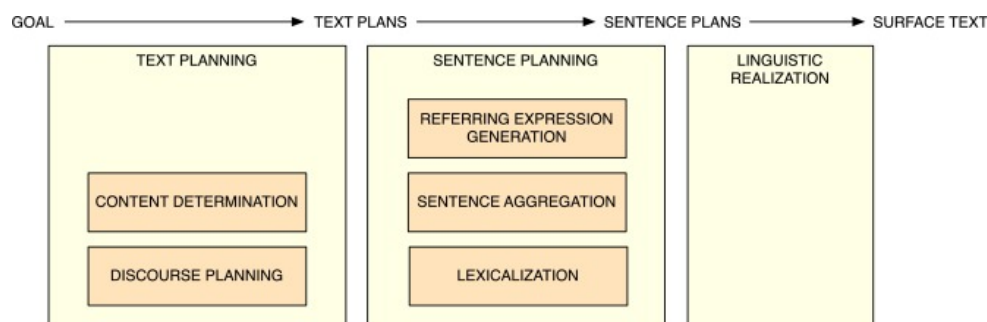


Figure 2.1: NLG system architecture and associated activities proposed by Reiter and Dale (Source: [1])

- **Text Planning**

- Content Determination
- Discourse Planning

- **Sentence Planning**

- Sentence Aggregation
- Lexicalization

– Referring Expression Generation

- **Linguistic Realization**

### 2.3.1 Text Planning

Text planning stage, also known as Macroplanification stage [3], is responsible for organizing the available information in order to choose which information should appear in the system's output (Content Determination); and determine a structure for the text (Discourse Planning).

In Section 2.2.1 systems were distinguished according to their type of input. In this sense, we established a difference between systems that take as input text and those that take as input structured data. If the system is going to transmit conclusions from an inquiry, the system input should be the users' answers to each question (STOP system [27]). If the system is going to make recommendations, the input should have descriptions of the elements that are going to be recommended, the user search history, user preferences or a concrete question about what the system is going to do, as in the MATCH system [33] solicitation "Compare restaurants in Manhattan". Another example, the SUMGEN-W [2] system makes monthly weather reports and is backed up by a database with accumulated information. In this system, the information enters the database as daily weather reports (Figure 2.2).

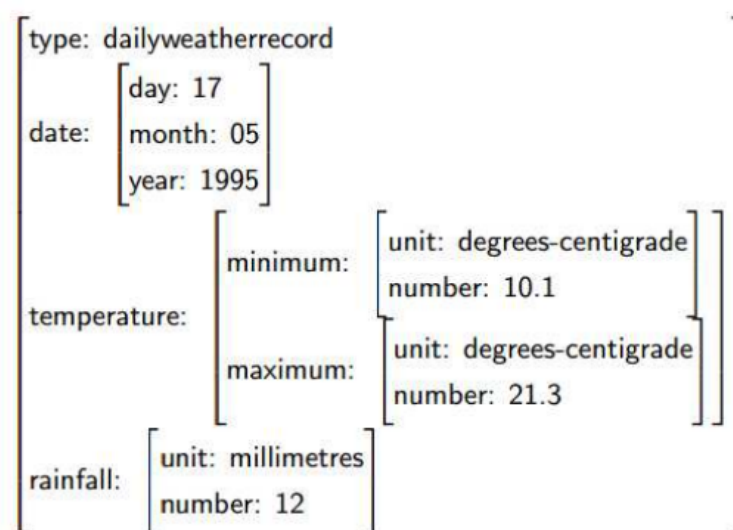


Figure 2.2: Input for the system SUMGEN-W. Daily weather report (Source: [2]).

These examples show that within the task of choosing the system input and in the text planning stage we should maintain a broad perspective. In this way, we should consider:

- **Knowledge base** → Information about the domain
- **Communicative goal** → Purpose of the text to be produced (explain, define, compare, etc.).

- **User model** → Characterization of the intended audience (novice, expert).
- **Discourse history** → Record of what has been communicated so far.

All of these aspects won't only affect the output of the system but also the intermediate stages as well.

Some authors refer a preprocessing stage before text planning. This stage is required when the data needs to be analyzed and interpreted. The analysis is responsible to find data patterns and the interpretation is responsible to insert relevant sentences to expand the application domain. For example, the SUMGEN-W daily weather report (Figure 2.2) could be the result of processing a data collection like the one shown in Figure 2.3.

```
95,121,1,5,1.00,100,-12.4,-4.406,-1.72,-1.298,1016,17.22,17.45,20.09,-13.58,-5.044,102.3,-2,149.9,92.8,0,.019
95,121,1,5,1.25,115,-12.49,-7.15,-2.168,-4.877,1016,17.13,17.45,20.1,-12.94,-4.867,103.1,-2,23.3,-195,0,.026
95,121,1,5,1.50,130,-12.93,-6.463,-2.053,-4.583,1016,17.02,17.22,20.15,-13.21,-4.801,101.9,-2,359.3,24.8,0,.043
95,121,1,5,1.75,145,-12.31,-5.145,-1.764,-3.092,1016,16.75,17.04,20.19,-12.98,-4.607,102.9,-2,330.6,1.416,0,.056
95,121,1,5,2.00,200,-14.71,-5.679,-2.034,-2.545,1016,16.52,16.94,20.21,-12.99,-4.658,104.3,-2,21.330.2,1.059,0,.066
95,121,1,5,2.25,215,-15.44,-6.819,-2.178,-4.025,1016,16.34,16.69,20.25,-13.62,-4.819,103.6,-2,329.9,0,0,.067
95,121,1,5,2.50,230,-12.69,-8.59,-2.624,-7.07,1016,16.26,16.65,20.29,-14.03,-4.859,104,-2,329.9,0,0,.073
95,121,1,5,2.75,245,-10.04,-4.78,-1.758,-1.46,1016,16.08,16.46,20.35,-14.59,-4.945,103.9,-2,329.9,0,0,.073
```

Figure 2.3: Initial data from SUMGEN-W before preprocessing stage (Source: [2]).

**Content Determination** - Content Determination is responsible for choosing which information from the input data must be displayed in the final output. This task analyzes and filters the input data in order to select the most important information.

**Discourse Planning** - In order to achieve a coherent text it is necessary to properly structure their elements. Cohesion and coherence are the most important factors to make a collection of messages become a discourse [3]. A. Ramos-Soto et al. define Discourse Planning as “the process by which the set of messages to be verbalized is given an order and structure. A good structuring can make a text much easier to read.” [15]. Again, as in all NLG activities, it is important to take into account extra-linguistic aspects. A text that explains a procedure will have a different structure than a text that compares two procedures. If the context is considered, there shall be a connection between previous structures so the users don't get confused.

### 2.3.2 Sentence Planning

Sentence planning, also known as Microplanification stage [3], takes as input the discourse plan where the messages that should appear in the final text are specified. Sentence planning consists in selecting which information is placed in any sentence and choosing the right words to express the information in the right way. This stage usually combines aggregation, lexicalization and referring expression generation described below. The output of this stage is the specification of

the text contained at the discourse plan. The final text should be completely characterized after this specification.

### 2.3.2.1 Sentence Aggregation

This activity groups several messages together in a sentence. The sentence aggregation process takes as input a tree-structured plan whose leaf nodes are messages. The aggregation system must decide both what messages to aggregate to form each sentence, and also what syntactic mechanism should be used to combine the messages. A good aggregation significantly improves the fluidity and readability of a text. The result must prevail the concision and the simplicity to produce a coherent text.

Dale and Reiter proposed several mechanisms to perform sentence aggregation, including the following [1]:

- Simple Conjunction: Doesn't change the lexical and syntactical content of the components.

"John is from Manchester. Steven is from London."

"John is from Manchester and Steven is from London."

- Ellipsis: If the two messages being aggregated have a common constituent, it may be possible to elude the repeated constituent.

"John bought a ball. Steven bought a ball."

"John and Steven bought a ball."

- Embedding: This involves embedding one clause as a constituent of another.

"The next game is the Manchester game. It starts at 2pm."

"The next game is the Manchester game, which starts at 2pm."

- Set Formation: If the messages being grouped together are identical except for a single constituent it may be possible to replace these with a single sentence plan that contains a conjunctive constituent.

"John bought a ball. John bought a pen. John bought a pencil."

"John bought a ball, a pen and a pencil."

Sometimes there are alternative, usually domain-dependent, ways of describing sets. For example, instead of explicitly saying: "John is afraid of bees, mosquitoes and flies.", the set of elements could be replaced by a single concept "John is afraid of insects."

In an NLG system the adequate mechanisms to perform this tasks should be selected. As in other stages, we should take into account the user model (can require less complicated texts), the system's requirements (providing a limited space favors the concision of the text), etc.

### 2.3.2.2 Lexicalization

Lexicalization is the task responsible for choosing specific words or concrete syntactical structures to refer to the content selected in the previous tasks. When lexicalization can choose from a variety of options, we should consider aspects like user's knowledge and preferences, the consistence of the lexical components and the relation with the tasks of aggregation and referring expression generation of each phrase.

Consider the message presented in Figure 2.4. This message mentions one domain relation, **GAMESTART**, and four domain entities: the home team **Manchester United**, the away team **Liverpool**, the stadium **Old Trafford** and the time **1400**. Lexicalization involves finding a word or phrase that can communicate a concept such as **GAME** to the reader (e.g. "*starts*" or "*begins*"). For example, "Manchester United vs Liverpool starts at 1400 at Old Trafford."



Figure 2.4: Message providing the type of relation and the arguments of the sentence.

### 2.3.2.3 Referring Expression Generation

Referring expression generation function is to determine the appropriate way to refer to the concepts and objects contemplated in the document plan in order to avoid ambiguity. A discourse must be able to distinguish entities by finding the particular characteristics of each one. Defining the problem of referring expression generation is one of the most consensual tasks in the NLG field.

In the example shown in Figure 2.4, the generation of referring expression is responsible for finding a noun phrase that identifies an entity. For example, we could use *2pm* for **1400**, or we could use the expression *The Red Devils* for **Manchester United**. The amount of information needed to do this will depend on the current discourse context. For example consider the italicized words in the following context:

1. The next game is *Manchester United vs Liverpool*. It starts at 2pm. Many TV channels will broadcast *this game*.

Here the entity **GAME** is initially referred to by the names of the teams competing against each other, which is a standard way of introducing into a discourse. The second reference to **GAME** uses the pronoun *it* because this is a standard way of referring to an object that was mentioned recently. The final reference is a definite description (*this game*), which is a standard way of referring to an entity when it has already been introduced, but where the context rules excludes the use of a pronoun. Generating each form of reference provokes different issues according to Reiter and Dale:

- **Initial Introduction** : The generation of initial references to objects hasn't had much focus among NLG researchers. The two common strategies used are to simply give the name of the object (if it has a name) or to describe the physical location of the object.
- **Pronouns** : Pronoun use and pronoun interpretation have been significantly studied in the NLG literature. However, pronoun generation has not had the same amount of focus. In many cases, a simple algorithm that works surprisingly well is to use a pronoun to refer to an entity if the entity was mentioned in the previous clause, and there is no other entity that the pronoun could possibly refer to [1]. In spite of being a fairly conservative algorithm, since it will not generate a pronoun in many circumstances where one could be used, it has the advantage that it does not often inappropriately insert a pronoun.
- **Definite Descriptions** : From a practical perspective, a simple but useful algorithm is to begin by including in the description a base noun describing the object (for example, game), and, if necessary, add adjectives or other modifiers to distinguish the target object from all other objects mentioned in the discourse [1]. For example, if the discourse has just discussed the Manchester United vs Liverpool game and no other games, then *the game* can be used to refer to Manchester United vs Liverpool. However, if the discourse also mentioned the Everton vs Swansea game, then a definite description for Manchester United vs Liverpool should add information to distinguish both games (e.g. the Manchester game).

### 2.3.3 Linguistic Realization

A. Ramos-Soto et al. defines Linguistic Realization as a "task, which directly matches the one defined in the general architecture, applies grammatical rules to produce a text which is syntactically, morphologically and orthographically correct" [15]. We can see this module as the encoder of syntactic and morphological knowledge. Some examples of this syntactic and morphological knowledge are [1]:

- **Rules about verb group formation** : The job of the linguistic realization is to construct an appropriate verb group based on parameters like tense specification (e.g. , past, present or future), the tense function (e.g. , interrogative or imperative), polarity (e.g. negated). Below,

some examples if the message to be transmitted concerns about the relationship between the concept **NEXT-GAME**, and the domain entity *Manchester United vs Liverpool*.

- Simple Present Tense, Negated:  
The next game is not Manchester United vs Liverpool.
- Future Tense, Question:  
Will the next game be Manchester United vs Liverpool?
- Past tense:  
The [last] game was Manchester United vs Liverpool.

- **Rules about agreement** : Linguistic realization should enforce words to agree in grammatical number (singular or plural). For example, *The next game is* in singular but we say *The next games are* in plural.
- **Rules about syntactically required pronouns** : Some syntactic rules require pronouns to be used in sentences. For example, we say *Yesterday, Peter hurt himself.* instead of *Yesterday, Peter hurt Peter.* if the person Peter hurt was Peter himself.

These are some examples of the particular features of the English language which the linguistic realization task can handle. This allows the rest of the NLG system to work with a much clearer and simpler model of the language, unaffected by these details.

### 2.3.3.1 Structure Realization

Some researchers add up a last step in the NLG task that is called structure realization [3]. This task should convert the output in order to present it in a particular platform . For example, in some cases, the output will be shown at a web page and, for that, needs HTML tags. This is one example but there are many more possibilities. Therefore, the usual actions of this task are: including tags in the document (HTML,  $\text{\LaTeX}$ , RTF, *SABLE*<sup>5</sup>) or creating tree-structures to include proper attributes to the final output (punctuation, bullets, etc...). An example of one of those tree-structures which is shown in Figure 2.5 [3].

An example of a multi-mode output system, and for that, needs a stage like this one, is MATCH system [33], a recommendation system that offers information about restaurants in New York. In this system, structure realization facilitates the geopositioning of the locals within a map and establishes the text format. For example, for a solicitation as "*Show me Italian restaurants in the West Village*", first, the system will provide a map with the geographic locations of Italian restaurants within West Village. Then, the user should decide among the options provided and, for that, the system provides a solicitation "*Compare these restaurants*" if the user makes a circular outline between the restaurants. In that case, MATCH generates another output, but this time, in voice or text form such as the one shown by M. Johnston et al. [33]: "*Compare-A: Among the selected restaurants, the following offer exceptional overall value. Ugual's price is 33 dollars. It has excellent food quality and good decor. Da Andrea's price is 28 dollars. It has very good food*



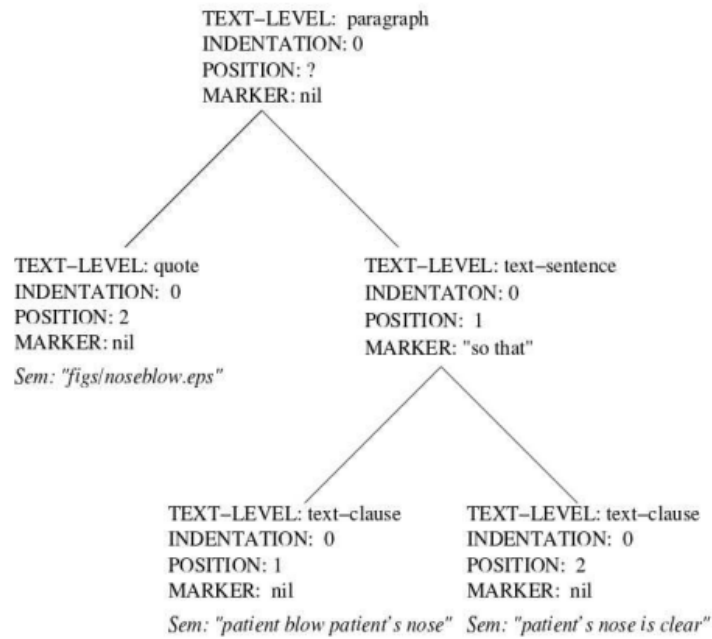


Figure 2.5: Instructions to generate the output (Source: [3])

*quality and good decor. John's Pizzeria's price is 20 dollars. It has very good food quality and mediocre decor."*

## 2.4 NLG Generic Methods

After discussing the tasks performed by an NLG system, this section provides an analysis on previous methods and approaches to Natural Language Generation and investigates their advantages and disadvantages. Adopting the classification done by M. Vicente et al. , we can differentiate knowledge-based methods from statistical methods [3]. On the one hand, knowledge-based methods are methods that use resources with strong linguistic influence such as dictionaries, thesaurus, rule sets or templates. From these resources is extracted morphological, lexical, syntactic and semantic information. On the other hand, when a system is build based on statistical methods, the information is backed up by a corpus and by the probabilities associated to each sentence of that corpus, which can be labeled or not. If the applied corpus is adequate, statistical-based methods are less restricted to a certain domain or language compared to knowledge-based approaches because they do not have to strictly follow rule sets or templates that usually take into account particularities of the specific context domain. It is worth noting that these approaches are not exclusive and in most situations the techniques applied to a system are associated to the specific NLG application. Therefore, some hybrid approaches were created that combine statistical and knowledge approaches. Furthermore, this section describes some of the techniques applied in NLG systems according to the stages where they are applied.

### 2.4.1 Knowledge-Based Methods

The common element between knowledge-based systems is their ability to explicitly represent knowledge. For that purpose, these systems make use of tools such as ontologies, rule sets, thesaurus or templates. NLG systems can also be template-based, i.e. utilize shallow processing where base structures of the output sentences are defined as templates that are transformed and values from the input data are filled in to generate the final output. The difference between knowledge-based and template-based approach resides in the fact that knowledge-based approaches require detailed grammatical and linguistic information to be embedded within the system.

M. Vicente et al. consider that the most relevant theories applied in knowledge-based or hybrid systems are [3]:

**RST: Rhetorical Structure Theory** RST is one of the main theories applied in NLG systems and is related to the cohesion of texts and the structure of sentences and paragraphs [34]. RST is intended to describe texts, rather than the processes of creating or reading and understanding them. It posits a range of possibilities of structure (i.e. various sorts of "building blocks" which can be observed to occur in texts). RST underlies on the idea that is possible to recursively decompose any text in a set of elements which establish rhetorical or discursive relations (schemas) between them. Furthermore, the analysis of rhetorical relations consider the intentions of who originates the communication as well as the desired effect on who receives it. The most relevant elements are called nucleus and the elements that depend on the nucleus are called satellites.

**SFG: Systemic Functional Grammar** Systemic-Functional Linguistics (SFG) is a theory that views language as "a network of systems, or interrelated sets of options for making meaning" [35]. A central notion is 'stratification', such that language is analyzed in terms of three levels of abstraction: Semantics, Lexico-Grammar and Phonology-Graphology. SFG starts at social context, and looks at how communication functions both act upon, and is constrained by, this social context. Functions of language are referred to as metafunctions. This theory proposes three general functions: the ideational, the interpersonal and the textual.

- Ideational metafunction (the propositional content);
- Interpersonal metafunction (concerned with speech-function, exchange structure, expression of attitude, etc.);
- Textual metafunction (how the text is structured as a message, e.g., theme-structure, given/new, rhetorical structure)

Usually, the last two metafunctions are not covered in other linguistic theories.

**TAG: Tree-Adjoining Grammars** A Tree Adjoining Grammar consists of a set of elementary trees, divided in initial and auxiliary trees that incorporate semantic content [36]. These trees constitute the basic building blocks of the formalism. Operations of adjunction and substitution are defined which build derived trees from elementary trees. The string language of a TAG is defined as the yields of all the trees derived by a TAG.

**MTT: Meaning-Text Theory** Linguistic models in MTT operate on the principle that language consists in a mapping from the content or meaning (semantics) of an utterance to its form or text (phonetics) [37]. Intermediate between these poles are additional levels of representation at the syntactic and morphological levels. Representations at the different levels are mapped, in sequence, from the unordered network of the semantic representation (SemR) through the dependency tree-structures of the Syntactic Representation (SyntR) to a linearized chain of morphemes of the Morphological Representation (MorphR) and, ultimately, the temporally-ordered string of phones of the Phonetic Representation (PhonR) (not generally addressed in work in this theory). The relationships between representations on the different levels are considered to be translations or mappings, rather than transformations, and are mediated by sets of rules, called "components", which ensure the appropriate, language-specific transitions between levels.

**Centering Theory** A theory that relates focus of attention, choice of referring expression, and perceived coherence of utterances within a discourse segment [38, 39]. The main objective is to provide an overall theory of discourse structure and meaning. Each discourse segment exhibits both local coherence (i.e. coherence among the utterances in that segment) and global coherence (i.e. coherence with other segments in the discourse). Corresponding to these two levels of coherence are two components of attentional state: the local level models changes in attentional state within a discourse segment, and the global level models attentional state properties at the intersegmental level. In NLG systems, the centering theory affects the selection and use of pronouns and descriptions.

### 2.4.2 Statistical Methods

As mentioned before, statistical methods are based on the extracted probabilities of a base text whether it is a corpus or a text from the web. One of the main tools for statistical methods are Language Models (LM) [40]. A statistical LM is a probability distribution over sequences of words. Given such a sequence, say of length  $n$ , it assigns a probability  $P(w_1, \dots, w_n)$  to the whole sequence. Having a way to estimate the relative likelihood of different phrases is useful in many natural language processing applications. A good LM can decide if a sentence is well-constructed, according to the associated probability of that sentence. In such cases, we say that LM accepts the sentence. The sentence is rejected when the associated probability is low.

In the task of NLG, a good LM can predict how the input (or part of the input) is going to be converted within the system. The main quality element of a LM is the size of the corpus or

database because the amount of contexts and domains that a word can be used is proportional to size of the learning corpus. In the task of NLG, three of the most used LM are:

**N-Gram Models** A n-gram [41] is a contiguous sequence of n items from a given sequence of text or speech. A n-gram model is a type of probabilistic model for predicting the next item in such a sequence in the form of a  $n - 1$  order (Markov chain order). The implementation of such models is widely used in recognition and learning algorithms. However, one of the main limitations of n-grams is the lack of any explicit representation of long range dependency.

**Models based on Stochastic Grammar** Stochastic Grammar [42] assigns a probability to each grammatical rule. The probability of a derivation (parse) is the product of the probabilities of the productions used in that derivation. Models based on stochastic grammar present naturally language restrictions. They also allow model dependencies as long as desired, even though the definition of such models and parameters can be very hard in complex tasks.

**FLM: Factored Language Model** Factored Language Model [43] is an extension of a conventional LM. In a FLM, each word is viewed as a vector of k factors:  $w_i = \{f_i^1, \dots, f_i^k\}$ . These factors can represent morphological classes or any lexical, syntactic or semantic characteristic. A FLM provides a probabilistic model  $P(f|f_1, \dots, f_N)$ , where the prediction of a factor f is based on N parents  $\{f_1, \dots, f_N\}$ . For example, if  $w$  represents a word token and  $t$  represents a grammatical class (POS:Part-Of-Speech), the expression  $P(w_i|w_{i-2}, w_{i-1}, t_{i-1})$  gives a model for predicting current word token based on a traditional n-gram model as well as the part of speech tag of the previous word. A major advantage of factored language models is that they allow users to specify linguistic knowledge such as the relationship between word tokens and POS.

### 2.4.3 Hybrid Models

Hybrid models (e.g. Nitrogen [44] or FERGUS [45]) combine knowledge and statistical methods to perform on the different tasks of a NLG. Other example of an hybrid system is the FLIGHTS [46] system, that presents personalized information of flights to each user (e.g. checks is the user is a student or if a user flies frequently). In its development, multiple knowledge bases are considered (user models, domain models and discourse history) to perform the content selection that should appear in the output. Then, the output is structured according to a template and the final text is generated by the OpenCCG framework [47]. The OpenCCG framework is a tool that applies n-grams and FLM, internally. Therefore, the selection, structure and generation of the text is performed in different stages.

### 2.4.4 Techniques Applied in the Different Stages

This subsection describes some of the techniques applied in NLG systems according to the stages where they are applied. These relations are not strictly defined and, when comes to the time

of constructing the system, the application of such techniques is fairly flexible. This subsection provides a general description of the techniques as well as concrete examples of systems that applied them, pointing out their relations with the statistical methods and knowledge methods approached discussed above.

#### 2.4.4.1 Text Planning Techniques

Text Planning is the initial stage of a NLG system. As mentioned before, this stage performs the content selection and document planning tasks. Some researchers allude to a possible general classification of the techniques associated to this stage depending if the content selection is done before or after document planning. As a result, if the system has the structure and just selects the content to complete it, we say it is a top-down strategy. Otherwise, if we have the content and it's their composition that set the document planning, we say it is a bottom-up strategy [48]. McDonald and Bolc proposed a triple division of techniques [49]:

The first mechanism is called direct replacement. This mechanism is responsible to add information to a basic scheme in order to make the document planning. It starts with a data structure that is gradually transformed into a text. The semantic coherence of the final text comes from the semantic coherence of the original structure. The main disadvantage of this mechanism it's his lack of flexibility since it is not possible to change the structure once the structure is generated.

The second mechanism is planning with rhetorical operators. Rhetorical operators or attributes precede of RST, and they also establish rhetorical relations among the elements to be included in the final text. The procedure starts with an analysis of the communicative goal and consists in expanding the main objective to achieve an hierarchical tree-structure where the end nodes are the propositions and the operators are the derivation rules.

Finally, we have the text schemes technique. This technique was originated in a system that applied it called TEXT [50]. The author detected regularities in text generating after analyzing a lot of examples. The author concluded that given a communicative goal, the information tends to be transmitted in the same order. To reflect this, the author coined the concept "*schematta*" and combined it with the use of rhetorical attributes. *Schemattas* are schemes which determine possible combinations of attributes, forming patterns and templates. Therefore, given a communicative goal such as describe or compare, the system is able to select a scheme that provides a plan indicating where and what should be said.

#### 2.4.4.2 Sentence Planning Techniques

Sentence planning phase is responsible for message aggregation, lexicalization and generation of reference expressions. As explained before, aggregation consists in grouping several messages together in a sentence in order to increase the coherence and fluidity of the text. Generally, this task requires a set of composition rules that will generate multiple possible outputs. Therefore, the system needs a selection function.

Given a rule set and the information units, there are some systems that generate multiple output options but there are as well systems that just produce one possible output option. *ASTROGEN*<sup>6</sup> is an example of a system that just produces one possible output option [51]. On the opposite side, SPOT is a system capable of selecting the most adequate option after generating multiple possibilities [52]. The system incorporates learning techniques which rate the multiple aggregations from a noted corpus. There are also systems that apply evolutionary approaches which are based in biological evolutionary techniques [53]. There is also some work based on tree-structured dependencies and Rhetorical Structure Theory (RST) [54].

The following task in sentence planning is lexicalization. Template-based lexicalization directly associates a unique word or phrase to each concept or relationship. For example, SUMGEN-W selects the text *"very much warmer than average"* when the average temperature lies within a specific range. Other systems have developed powerful graph-rewriting algorithms which use general *dictionaries* or thesaurus to select the adequate words or phrases [55]. There are also knowledge bases as WordNet [56] and FrameNet [57] to perform more complex inferences. Another approach is based on stochastic methods where a stochastic generator is able to select the adequate words corresponding to a collection of semantic representation based on classifiers [58]. The final task is referring expression generation. This task selects words or expressions which identify entities from the domain. This tasks should be unequivocal, but some techniques have been developed that relaxes the exigency level [59]. In this case, we call it simple referring expression generation.

One of the most used mechanisms is the incremental algorithm [60] or one of its variants as the context-sensitive algorithm [61]. The input of this algorithm includes the entity that needs to be acknowledge as well as an entity collection called contrast collection. Every entity have attributes associated to them (Figure 2.6). Then, it starts to iterate a list of attributes among the ones selected to form the final expression and it excludes entities from the contrast collection. Another technique that is widely used and applied is graph-based algorithm [4]. The edges of the graph are labeled with properties and attributes. The idea is that is possible to systematically generate all the sub-graphs from a directed graph. Starting from the sub-graph that only has the vertex, representing the object to reference, the algorithm starts to perform a recursive expansion by adding the adjacent edges to the sub-graph that is active in that moment.

#### 2.4.4.3 Linguistic Realization Techniques

The first work that presented corpus-based statistical techniques to perform linguistic realization techniques was developed by Langkilde and Knight in 1998 [44]. This method applies a n-gram model that decides which word transformations should be applied (whether to use the plural, if is a question or not). The method chooses the output which probability is higher.

Other approach is the application of the spanning tree algorithm [62]. The procedure starts with a word collection that forms a graph. This technique is able to convert it in an acyclic graph. The sibling nodes are ordered by a n-gram LM. In this way, we obtain several distinctive possible

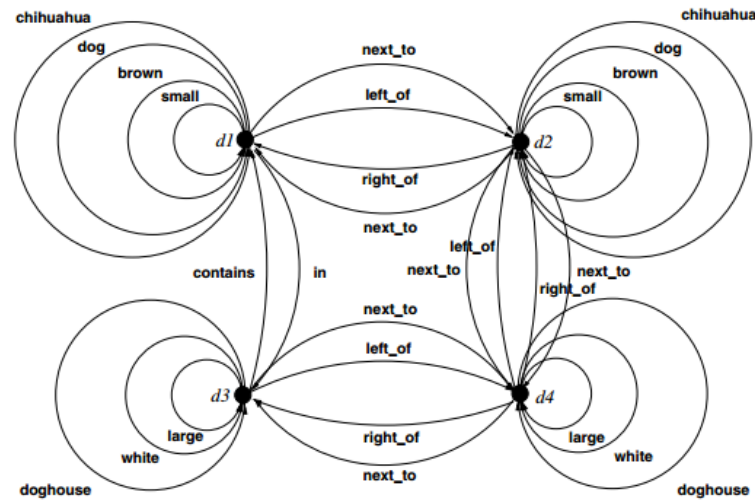


Figure 2.6: Graph of a scene of a dog close to a doghouse. Graph-based algorithm for referring expression generation (Source: [4]).

tree-structures. To complete the process, it is applied an argument satisfaction model that select the best structure to grammatically order the words, enabling the generation of the final text.

## 2.5 Evaluation Methodologies

There are multiple possibilities when comes to evaluate an NLG system [63]. On the one hand, it is possible to evaluate the effect of a system on something that is external to it, for example, the effect on human performance or the value added to an application. This is called an *extrinsic evaluation*. On the other hand, *intrinsic evaluation* focus on the quality of the produced text. We can also distinguish manual evaluation from automatic evaluation. In the NLG field, it is usual to perform manually the extrinsic evaluation and perform automatically the intrinsic evaluation [63]. Usually, manual evaluation produces good results. Especially, if the criteria used are the adequacy and fluency. However, this type of evaluation is very expensive, time consuming, and can be difficult to carry out [64]. An example of manual evaluation was the evaluation performed in the STOP system. They performed inquiries to evaluate the task effectiveness (e.g. how many users actually stopped smoking, how much time it took, etc.). STOP's evaluation needed 20 months and costed 75 thousand pounds to complete [27]. Regarding intrinsic evaluation which values the properties of the system without taking into account external effects usually requires comparing the system's output with some reference text or corpus, using metrics or ratings. In the task of evaluating NLG systems, it is possible to take into account aspects related to the system's complete functioning or with the system's modules functioning [65]. Regarding system's evaluation, we should take into account if the output is appropriate according to the communicative goal, the discourse structure, coherence, ambiguity and vocabulary quality.



Both evaluations might seem linked since a system able to produce a high quality text, should successfully accomplish its communicative goal. But intrinsic evaluations tend to favor natural generation system that do not diverge much, while users often prefer output with more variation. Below, are some examples where intrinsic and extrinsic evaluations produced divergent results:

1. **A. S. Law et al. [66]** - Graphical representations of medical data were compared with textual descriptions of same data. In intrinsic assessments doctor rated higher the graphs but in extrinsic diagnostic text performed better.
2. **A. S. Lennox et al. [67]** - The STOP system generated tailored smoking-cessation advice based on the user's response to a questionnaire. Despite the intrinsic evaluation rated high the quality and the intelligibility of the produced texts, extrinsic evaluation showed that the system was not able to make their patients to stop smoking.

Most NLG systems' evaluation is done using one of 3 basic techniques [68]:

1. Trained assessors evaluate the quality of system outputs according to different criteria, typically using rating scales [9].
2. Automatic measurements of similarity between reference and system outputs, for example BLEU [69] and METEOR [70].
3. Human assessment of the degree of similarity between reference and system outputs [71].

BLEU and METEOR, are metrics that perform automatic evaluation. BLEU (BiLingual Evaluation Understudy) is a metric that evaluates how close is an automatically generated text from a reference text, previously generated by a human. The closer they are the greater the text quality in evaluation. METEOR (Metric for Evaluation of Translation with Explicit Ordering) uses sentences as unit element for the evaluations. The metric creates an alignment between components (words) from the sentence under test and a reference test. The content of this metric is obtained by calculating the harmonic mean of the accuracy of alignments, along with the number of most significant alignments. These algorithms allow faster and cheaper evaluation of NLG systems. However, when the sample is small, these metrics have proved to be inaccurate. Therefore, a subject that is open for debate within the NLG area is how to perform a good evaluation [15]. The majority of evaluation methods are quantitative [72]. This means that they try to have numerical ratings to evaluate how well the system is performing according to different criteria. Other kind of approaches is qualitative evaluations in which experts are asked to share their opinion about the system's output. A. Ramos-Soto et al. described as an ideal evaluation scenario one that could conjugate "[a] quantitative evaluation together with a qualitative one and, after the system has been deployed in its target domain, an extrinsic evaluation" [15]. That is because, according to them, the intrinsic evaluation would tell how ready the NLG system is to produce an high quality text in a real environment. After correcting the flaws exposed in the results of the intrinsic evaluation, the extrinsic evaluation would check on the success level of the NLG system to achieve its communicative goal.



## 2.6 NLG Tools

This section presents a series of tools applied in the NLG stages (Table 2.1). There are a lot of web tools and free applications, especially oriented to the realization stage. A lot of system and resources are available in the web [73].

### 2.6.1 Natural Language Toolkit (NLTK)

NLTK is a Python library for building programs to work with human language data [74]. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet. NLTK was originally created in 2001 as part of a computational linguistics course in the Department of Computer and Information Science at the University of Pennsylvania. Since then it has been developed and expanded with the help of dozens of contributors. It has now been adopted in courses in dozens of universities, and serves as the basis of many research projects. NLTK provides several modules for several NLP tasks, such as: accessing corpora, string processing, part-of-speech tagging and parsing.

The creators of NLTK wrote a book called Natural Language Processing with Python. It guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analyzing linguistic structure, and more. The book is being updated for Python 3 and NLTK 3. NLTK is open source software. The source code is distributed under the terms of the Apache License Version 2.0. The documentation is distributed under the terms of the Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 United States license.

### 2.6.2 RealPRO

RealPRO is a tool that performs the realization stage [75]. The system's input must have an ASCII, HTML or RTF format, therefore is considered a D2T system. It involves a linguistic knowledge-based procedure, initially just in English language. After it is possible to upgrade to other idioms. The input data is a dependency tree-structured diagram. This diagram has two components:

- Syntactic relations which are represented by labeling the bows that relates the nodes.
- Lexemes that are represented by the labeling each node. The lexemes that are stored are the ones which have meaning. RealPRO is not able to perform syntactic analysis, therefore all the lexemes that add meaning must be specified.

Once the diagram is generated, the tool adds functional words which generate a second tree-structured diagram. Linear precedence rules are performed in the second diagram based on the bow's labels. Afterwards, the elements of the sentences are conjugated. Finally, the tool adds the punctuation marks and generates the required instructions to adapt the output to the desired format.

### 2.6.3 SimpleNLG

SimpleNLG is a tool focused in the realization stage developed exclusively for English language [76]. The tool is formatted as a java library. The main objective is giving help to write grammatically correct sentences.

The tool was constructed based on three basic principles:

- **Flexibility.** SimpleNLG is a combination of a scheme-based system and a sophisticated system. Through the combination of both, it is possible a more comprehensive syntactic coverage.
- **Robustness.** When an input is incomplete or incorrect, the tool is able to produce an output, even though it is probable that the output is not as intended.
- **Independence.** Morphological and syntactic operations are clearly distinguished.

The library provides an interface with which is possible to interact via java coding. Starting from a base element, equivalent to the main verb of the sentence, are performed concatenations to the other elements participating in the main action. Once the elements are aggregated, we need to indicate the tense function (past, present, future) and specification (interrogative, affirmative..). Finally the tool generates a sentence based on the given parameters. SimpleNLG is licensed under the terms and conditions of the Mozilla Public License (MPL).

### 2.6.4 PyNLPI

PyNLPI (pronounced as 'pineapple'), is a Python library for Natural Language Processing. It can be used for basic tasks such as the extraction of n-grams and frequency lists, and to build simple language models. It contains parsers for file formats common in NLP (e. g. FoLiA/Giza/Moses). The most notable feature of PyNLPI is a very extensive library for working with FoLiA XML (Format for Linguistic Annotation). The library is divided into several packages and modules and works both on Python 2.7 , as well as on Python 3. PyNLPI is licensed under the terms and conditions of GPL (General Public License).

### 2.6.5 NaturalOWL

NaturalOWL is a D2T tool that generates text with information from linguistically annotated OWL ontologies [77]. OWL is a standard to specify ontologies into the Semantic Web. NaturalOWL uses all three stages that were approached in section 2.3 to generate the text.

For text planning stage, NaturalOWL selects from the ontology all the logical facts that are directly relevant to that instance. The tool may be instructed to include facts that are further away in a graph representation of the ontology, up to a maximum (configurable) distance. The selected facts of distance one are then ordered by consulting ordering annotations which specify a partial order of properties. Second distance facts are always placed right after the corresponding directly

relevant facts. In the application domains, this ordering scheme was adequate, although in other domains more elaborate text planning approaches may be needed.

In the sentence planning stage, NaturalOWL lets the user to configure the maximum number of sentences that can be aggregated. Generally, NLG systems aggregate the maximum possible sentences in order to achieve better legibility. Finally, in realization stage, NaturalOWL takes the sentence planning output and represents it by adding punctuation symbols and capital letters where necessary.

### 2.6.6 OpenCCG

OpenCCG [47], the OpenNLP CCG Library, is a collection of natural language processing components and tools which provide support for parsing and realization with Combinatory Categorical Grammar (CCG). [78]. Combinatory Categorical Grammar (CCG) is a form of lexicalized grammar in which the application of syntactic rules is entirely conditioned on the syntactic type, or category, of their inputs. No rule is structure or derivation dependent. The OpenCCG realizer takes as input a logical form specifying the propositional meaning of a sentence, and returns one or more surface strings that express this meaning according to the lexicon and grammar. OpenCCG is a tool that applies n-grams and FLM described in section 2.4, internally.

Tools	Progr. Language	License	Last Update
NLTK <sup>1</sup>	Python	Apache License Version 2.0	June 2016
RealPro <sup>2</sup>	Java	Unkown	Unknown
SimpleNLG <sup>3</sup>	Java	Mozilla Public License	May 2016
PyNLPI <sup>4</sup>	Python	GNU Library Public License	June 2016
NaturalOWL <sup>5</sup>	Java	GNU Library Public License	April 2013
OpenCCG <sup>6</sup>	Java	GNU Library Public License	May 2016

Table 2.1: Additional information on the NLG Tools described in Subsection 2.6

<sup>1</sup><http://www.nltk.org/>

<sup>2</sup><http://www.cogentex.com/technology/realpro>

<sup>3</sup><https://github.com/simplenlg/simplenlg>

<sup>4</sup><https://pypi.python.org/pypi/PyNLPI/>

<sup>5</sup><https://sourceforge.net/projects/naturalowl/>

<sup>6</sup><http://openccg.sourceforge.net/>

## 2.7 Resources: Scientific Conferences and Events

Documentation related to techniques, associated theories, evaluation and new challenges can be found in the records of *International NLG Conference* (INLG) as well as in *European Workshop on Natural Language Generation* (ENLG):

- **INLG** - The International Natural Language Generation conference is the biennial conference of the Special Interest Group on Natural Language Generation (SIGGEN). It is held since 2000. The 9th INLG conference will be held in Edinburgh, Scotland in September 2016.
- **ENLG** - The European Natural Language Generation workshop is a biennial series of workshops on natural language generation that has been running since 1987. The last edition was the 15th ENLG held in Brighton, UK in September 2015.

NLG papers can be found at:

- **ACL** - The Association for Computational Linguistics is the premier international scientific and professional society for people working on computational problems involving human language, a field often referred to as either computational linguistics or natural language processing (NLP). The association was founded in 1962, originally named the Association for Machine Translation and Computational Linguistics (AMTCL), and became the ACL in 1968. Activities of the ACL include the holding of an annual meeting each summer and the sponsoring of the journal *Computational Linguistics*, published by MIT Press; this conference and journal are the leading publications of the field.
- **EACL** - The European Chapter of the ACL (EACL) is the primary professional association for computational linguistics in Europe. It provides a number of services to its members and the community:
  - The conference. EACL 2014 was held at the University of Gothenburg's Centre for Language Technology from April 26 to April 30, 2014. From 2014 the EACL conference will take place every three years. In addition, the ACL conference is hosted in Europe every third year.
  - Twice-yearly newsletter carrying news about activities organised or supported by EACL, and major European happenings and events.
  - Support for educational initiatives in the field – for example, EACL-sponsored introductory courses in CL at ESSLLI summer schools and studentships at specialist workshops.
- **ANLP** - The Association for Neuro Linguistic Programming is a UK organization to promote NLP. ANLP publishes *Rapport* magazine quarterly, which is available to non-members by subscription. ANLP has also published the proceeding of a NLP Research Conference held at the University of Surrey in 2008.

- **IJCAI** - The International Joint Conference on Artificial Intelligence, the main gathering of researchers in AI. IJCAI has been held biennially in odd-numbered years from 1969 until 2015.
- **AAAI** - The Association for the Advancement of Artificial Intelligence is a nonprofit scientific society devoted to advancing the scientific understanding of the mechanisms underlying thought and intelligent behavior and their embodiment in machines. Major AAAI activities include organizing and sponsoring conferences, symposia, workshops and publishing books, proceeding and reports.

## 2.8 Applications of NLG in the Area of Journalism

Now that we discussed the classification, the tasks and the approaches of a NLG system, now we will focus on the problem in hand. This section will focus on the applications of NLG in the area of Journalism and will provide some related work in generation of sports matches summaries.

Over time journalists were assisted by computer systems in several phases of news production. Computer systems helped at collecting, analyzing and organizing data but journalists always had the last word to say when it comes to create and publish the news. Nowadays, advances in the area of NLG allowed the creation of algorithms to automatically generate news stories without human interference, aside of programming the algorithm. Andreas Graefe named this phenomenon as “Automated Journalism” [79]. Automated news started in the 1980s from the domain of weather forecasting but rapidly spread over to other areas such as financial reporting, where the speed in which the information is provided is the key value. NLG also made had an impact on sports reporting, due to the abundant availability of statistics and the formulaic templates and stock phrases which can be used for game reports [9]. Diversification of individual interests has aroused the demand for the individual media in place of mass media. In order to satisfy the needs, media has to cover a huge amount of data. Algorithms can use the same data to tell stories from different angles, in multiple languages, and personalized to the needs and preferences of the individual reader. Also, software providers have started to release tools that allow users to automatically create stories from their own data. In fact, algorithms work very well on fact-based stories for which clean, structured and reliable data are available. In such situations, NLG algorithms may be preferred over manual document creation because they increase accuracy and reduce updating time.

Since Associated Press, one of the world’s largest and well-established news organizations, has started to automate the production of its quarterly corporate earnings reports, the use of algorithms to automatically generate news has shaken up the journalism industry [80]. Due to higher profit expectations there is a broader trend in journalism to lower the variable costs involved in news production. This, allied to the fact of the increasing availability of structured data and the demand for individual media, means automated journalism is likely to stay. One main concern is how automation of news will affect journalists and the way they work. First of all, it’s important to emphasize that automated journalism cannot be used in domains where no data is available and

difficult when the quality of data is poor. Algorithms apply predefined rules and statistical methods but cannot provide deeper interpretations of the information like explaining new phenomena or establishing causality. Thus, whether or not technology will replace or complement journalists will depend on the task and skills of the journalist. It is obvious that algorithms can save a lot of routine work. As a result, journalists will be able to focus on tasks that algorithms cannot do such as in-depth reporting, interviews, and investigating. Aspects like personality, analytic skills and creativity become more important, while professional routine tasks can be automated.

A lot of recent studies have focused on the credibility of automated journalism. Hille Van der Kaa and Emiel Kramer created four dutch robot written news items based on algorithms outlined in the data-to-speech system (D2S) and created by Theune [71, 5]. Two of those articles reported on a result of a football match and the other two articles reported on stock prices. The contents and sentences were the exact same for both topics and only the sources of the article was manipulated. One of them showed "This article is written by a computer" and the other "This article is written by a journalist". Each participant was randomly presented with one of the four texts and was asked to evaluate the perceived expertise and trustworthiness of the news writer and of the content of the new story. They found that news consumers perceive the trustworthiness and expertise of the computer writer and journalist equally. Arjen Van Dalen tried to analyze the reactions of the journalists to the launch of a network of machine-written sport websites [9]. He came to the conclusion that contrary to previous studies "the journalists do not reject the new development". In fact, "journalists see this new technology as a relevant development that leads them to reconsider their own professional skills".

Generation of sports match reports has been repeatedly a subject of investigation in the area of Natural Language Generation. We have multiple examples such as, GoalGetter [5], or Multilingual Cricket Summaries Generator [81] which will be described below. Bouayead-Agha et al. states that this is not by chance: "[such] commentaries constitute a popular genre and there is thus a demand for automation of their production" [7].

### 2.8.1 GoalGetter

GoalGetter is a Data-to-Speech (combines speech and language generation techniques) system that was created based on an existing one, the DYD (Dial Your Disk) system in order to show that the original system is easily portable regardless of the domain, the language and the speech output. GoalGetter system generates spoken summaries of football matches from texts stored in a Teletext page that contain data on one or more football matches. The database contains information about the teams and their players. The output of the system is a correctly pronounced text in Dutch which conveys the information on one of the matches of the Teletext page. GoalGetter has two main modules: the text generation module (TGM) and the speech output module. The major difference of GoalGetter with most other NLG systems is that it does not use a pipelined architecture. The general architecture of the TGM consists of two modules: Generation and Prosody, and three data resources: 1) a set of syntactic enriched templates; 2) the Knowledge State; and 3) the Context state (Figure 2.7).

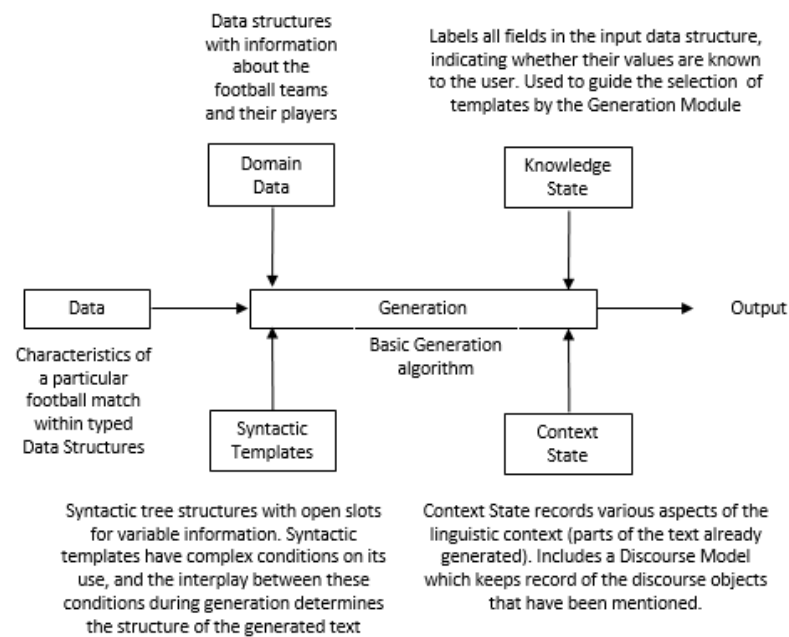


Figure 2.7: System architecture of GoalGetter (Source:[5])

The data concerning the results of a particular of a particular football match and the data on the teams and their players (the domain data) are one part of the input for Generation. Generation also uses a collection of syntactic templates internal to the system. Syntactic templates are tree structures that contain slots to be filled with the appropriate information from the input data. Some of these conditions are formulated as conditions on the Knowledge State. The Knowledge State acknowledges which data have been transmitted to the user, and which data have not yet been transmitted. The Context State records the parts of the text already generated. It includes a Discourse Model which keeps a record of the already mentioned discourse objects to be used for referring expression generation. A sample output and the respective English translation of GoalGetter is displayed in Figure 2.8.

### 2.8.2 Multilingual Cricket Summary Generator

Multilingual Cricket Summary Generator is a data-to-text NLG system created to assess the suitability of template-based approaches to language-independent NLG systems [6]. Multilingual Cricket Summary Generator generates summaries of cricket games in English and Bangla from cricket match scorecards that are saved as plain text files. The author followed Reiter and Dale architecture to design the system (Figure 2.9).

The text planning stage is formed by a pre-processor and a content selection task. The text

## Output:

Go Ahead EAGLES / ging op BEZOEK bij Fortuna SITTARD // en speelde GELIJK ///

Het duel eindigde in TWEE // • TWEE ///

VIJFENVEERTIG honderd TOESCHOUWERS / kwamen naar 'de BAANDERT' ///

<new-par>

De PLOEG uit SITTARD / nam na ZEVENTIEN MINUTEN de LEIDING / door een TREFFER van HAMMING ///

EEN minuut LATER / bracht SCHENNING van Go Ahead EAGLES / de teams op GELIJKE HOOGTE ///

Na ACHTENVEERTIG minuten / liet de AANVALLER HAMMING / zijn TWEDE doelpunt aantekenen ///

In de VIJFENZESTIGSTE minuut / bepaalde de Go Ahead EAGLES speler DECHEIVER de EINDSTAND / op TWEE // • TWEE ///

<new-par>

De wedstrijd werd GEFLOTEN door SCHEIDSRECHTER UILENBERG ///

Hij deelde GEEN RODE KAARTEN uit ///

MARBUS van Go Ahead EAGLES / liep tegen een GELE kaart aan ///

## Translation:

Go Ahead EAGLES / visited Fortuna SITTARD // and DREW ///

The duel ended in TWO // • ALL ///

FOUR thousand FIVE hundred SPECTATORS / came to 'de BAANDERT' ///

<new-par>

The TEAM from SITTARD / took the LEAD after SEVENTEEN MINUTES / through a GOAL by HAMMING ///

ONE minute LATER / SCHENNING from Go Ahead EAGLES / EQUALISED the score ///

After FORTY-EIGHT minutes / the FORWARD HAMMING / had his SECOND goal noted ///

In the SIXTY-FIFTH minute / the Go Ahead EAGLES player DECHEIVER brought the FINAL SCORE to TWO // • ALL ///

<new-par>

The match was OFFICIATED by REFEREE UILENBERG ///

He did NOT issue any RED CARDS ///

MARBUS of Go Ahead EAGLES / picked up a YELLOW card ///

Figure 2.8: Output and English translation of the GoalGetter system (Source:[5])

planning stage selects which content of the input data should appear in the summary and determines the structure of the selected content (document plan). The document plan is the input of the sentence planning stage, which is only composed by the aggregator module (referring expression generation was not implemented). The aggregator module selects which items of the document plan can be realized in the same sentence. The output of the aggregator module is supplied to the surface realizer, which makes use of a set of sentence and phrase templates to create the natural language sentences. Then, the post-processor applies the rules specified in the lexicon of the language to be generated, and produces the summary in natural language. The presented system is based on two levels of templates, i. e. sentence and phrase templates that are easily extensible for creating variation in the output.



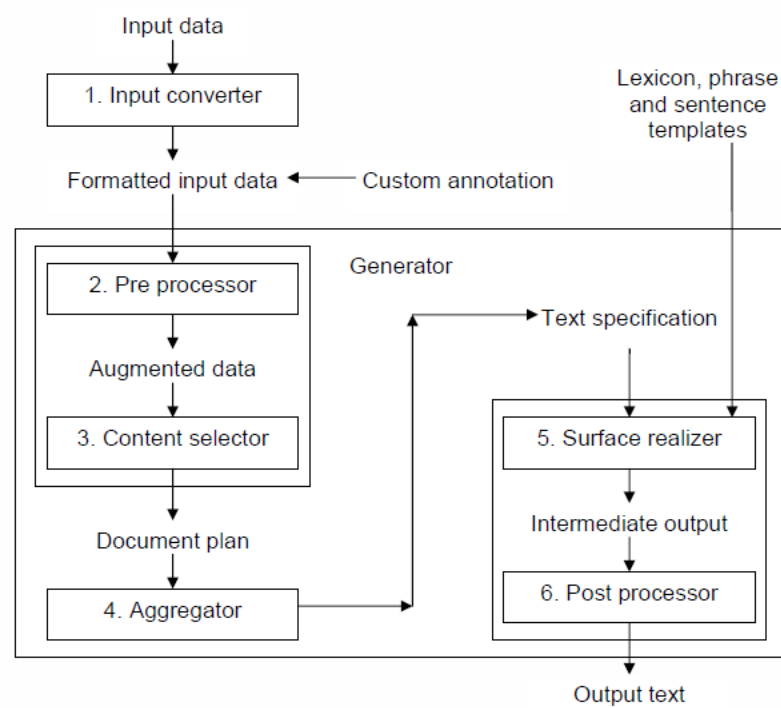


Figure 2.9: System architecture of Multilingual Cricket Summary Generator (Source:[6])



## Chapter 3

# The GameRecapper System

### 3.1 Introduction

As stated in Chapter 1, the main purpose of this dissertation is to generate journalistic pieces using information from structured databases. Today, there is a need to report on a lot of football matches and, many of them, are played at the same time. A significant amount of human resources and working time would be needed for journalists to watch every match that they have to make a report. Due to the abundant availability of information that is stored in databases, journalists are able to make a report based just on that information. However, if this process was automated it would save a lot of working time of journalists. As a result, journalists would have more time for in-depth reporting.

First, we start with an example of the match information input and the corresponding journalist match report. The match information is available in the *www.zerozero.pt* project and Figure 3.1 presents how the characteristics of a particular football match are displayed on the website.

The first 2 paragraphs of the corresponding journalist authored match report to the input data of Figure 3.1 are:

- **Portuguese** - "Em Arouca, a equipa de Lito Vidigal continua com o sonho da Europa bem presente, depois da vitória por 3x2 sobre a Académica, este sábado à tarde. Por sua vez, os estudantes, deram mais um passo atrás na corrida pela salvação.

E as coisas nem começaram mal para a formação de Coimbra, que marcou primeiro, aos 10 minutos, por Pedro Nuno, após cruzamento de Rafa Soares. A vantagem era justa, até porque a primeira dezena de minutos tinha sido de domínio da Académica."

- **Translation** - "In Arouca, Lito Vidigal's team keeps the Europe dream well alive, after the 3x2 victory against Académica, this saturday afternoon. On the other hand, the students took a step backwards in the fight to avoid relegation.

The game did not start badly for Coimbra's formation as they scored first, at the tenth minute, thanks to Pedro Nuno's goal, after a cross by Rafa Soares. Académica deserved the lead because they were controlling the game in the first ten minutes."

Saturday 2 April 2016 - 16h15 - Municipal de Arouca (POR) (Arouca) - 1537 Attendance  
Primeira Liga 2015/16 - League Matchday 28 - Carlos Xistra (POR)




## Arouca 3-2 Académica

18 Jubal Júnior  
39 Lucas Lima  
43 Artur Moreira

3-1

Pedro Nuno 11  
Gonçalo Paciência 62



MATCH REPORT REALTED: STADIUM REFEREE VIDEOS PHOTOS COMMENTS

**AROUCA**

1	Rafael Bracali	
3	Hugo Basto	
14	Gegé	
33	Jubal Júnior	⚽ 18'
6	Lucas Lima	⚽ 39'
22	Adilson Tavares	🟡 43' 🟡 74'
66	Nuno Coelho	🟡 63'
7	Artur Moreira	⚽ 43' ▼ 90+3
12	Mateus	
95	Walter González	▼ 73
11	Ivo Rodrigues	🟡 51' ▼ 63

**SUPLENTES**

85	Rui Sacramento	
24	Emiliano Albín	▲ 90+3
4	Sema Velázquez	
50	Nuno Valente	▲ 63
17	Roberto Rodrigo	
87	Zequinha	
99	Maurides	▲ 73

**TREINADORES**

🇵🇹 Lito Vidigal

**ACADÉMICA**

88	Pedro Trigueira	
5	Ricardo Nascimento	
13	João Real	
55	Rafa Soares	🟡 10' 62'
28	Nuno Piloto	🟡 49'
27	Pedro Nuno	⚽ 11' ▼ 69
21	Leandro Silva	🟡 52'
7	Marinho	▼ 57
43	Nii Plange	
9	Rabiola	🟡 45+1' ▼ 57
30	Rafael Lopes	

**SUPLENTES**

32	Lee Oliveira	
2	Aderlan Silva	
14	Iago Santos	
20	Rui Pedro	▲ 69
77	Hugo Seco	▲ 57' 🟡 66'
19	Gonçalo Paciência	▲ 57' ⚽ 62' 🟡 66'
17	Inters Gui	

**TREINADORES**

🇵🇹 Filipe Gouveia

Figure 3.1: Match information of *www.zerozero.pt*.

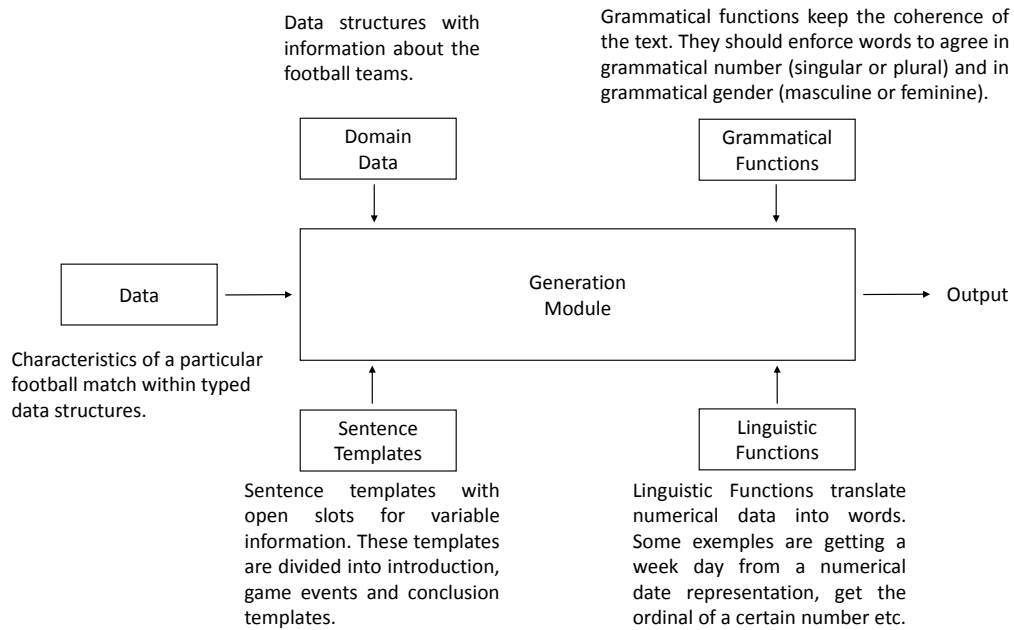


Figure 3.2: Architecture of GameRecapper.

After analyzing the match report, we can understand why automatically generating such a news piece is difficult without background knowledge that journalists have access to. For example, in the last sentence, the journalist stated that Académica deserved the lead at the tenth minute because they were controlling the game. It is impossible to deduce that from the input information since scoring the first goal does not mean that the team is controlling the game. The same observation holds for the first sentence in the first paragraph where the author mentions that Arouca keeps the Europe dream well alive. If on the one hand, we can observe that Arouca maintained the fifth position that gives access to the Europa League, we cannot say for a fact that this is a dream for Arouca's team. Based on this argument, we focused on utilizing the available data in the input to extract the key events and realize them into proper natural language sentences.

## 3.2 Description of GameRecapper

In order to make summaries of game reports, our solution was a D2T template-based system similar to the text generation module of the GoalGetter [5] system (described in Subsection 2.8.1). The GameRecapper system generates Portuguese summaries of football matches. The data which forms the input of GameRecapper is retrieved from an API developed by *www.zerozero.pt* that transforms the information of a certain game on their webpage into a JSON tree structure. We chose to convert into a JSON tree structure since it allows a logical hierarchy and ordering of information. JSON was chosen over XML since JSON doesn't require templates such as XSLT for transformation and it is more readable. Our domain of application will be the Liga NOS 2015/2016.

Figure 3.2 provides a demonstration of the GameRecapper general architecture. The Generation module is the basic algorithm that creates the game summaries. It takes the JSON tree structure as input with data concerning the characteristics of a particular football match. As in GoalGetter, the GameRecapper makes use of domain data with information about the teams. For each team of the Liga NOS 2015/2016, the domain data module has information about the city of the team, their home stadium, and their nickname. This will allow us to achieve more variation in the generated news pieces since it will provide additional information about the teams that are mentioned. Let us take as example the output text describing the match of Figure 3.1, given below. We can find an example of the use of domain data in the sentence that describes the final goal, where Académica is referred to as **"the students team"** (Académica's nickname).

- **Output:**

*O Arouca bateu a Académica por 3-2, este sábado à tarde, no Estádio Municipal de Arouca. Pedro Nuno, aos 11 minutos, abriu o ativo para a equipa visitante. Aos 18 minutos, Jubal Júnior devolveu a igualdade ao encontro. Aos 39 minutos, o Arouca confirmou a reviravolta no marcador, com Lucas Lima a ser o marcador de serviço. O Arouca ampliou a vantagem aos 43 minutos, quando Artur Moreira colocou o resultado em 3-1, após passe de Adílson Tavares. A equipa dos estudantes fixaria o resultado final em 3-2 com um golo de Gonçalo Paciência, depois de uma assistência de Rafa Soares, aos 62 minutos. Com este resultado, o Arouca mantém o 5º lugar e passa a somar 44 pontos. Já a equipa de Filipe Gouveia continua com 23 pontos e mantém o 17º lugar.*

- **Translation:**

*Arouca beat Académica with a 3-2 victory, this Saturday afternoon, in the Estádio Municipal de Arouca. Pedro Nuno, at the 11th minute, opened the score for the away team. At the 18th minute, Jubal Júnior equalized the score. At the 39th minute, Arouca completed the comeback, with a goal by Lucas Lima. Arouca extended their lead at the 43rd minute, when Artur Moreira put the scoresheet at 3-1, after an assist by Adílson Tavares. The students team fixed the final result in 3-2 with a goal by Gonçalo Paciência, after an assist by Rafa Soares, at the minute 62. With this result, Arouca keeps the 5th position and has now 44 points. On the other side, Filipe Gouveia's team continues with 23 points and remains at the 17th position.*

The Generation module also makes use of a collection of templates that were manually annotated from an initial corpus. Each template contains open slots for variable information. Templates are divided into groups according to certain events on the game (e.g. first goal of the game and game has more than one goal) or characteristics of the game (e.g. the home team was the winning

team). The procedure to build the templates as well as their content and rules will be discussed in detail below.

The Generation module also makes use of grammatical functions and linguistic functions. Grammatical functions ensure the coherence of the text. They should enforce words to agree in grammatical number (singular or plural) and in grammatical gender (masculine or feminine). The linguistic functions translate some numerical data into words. For example, one linguistic function gives the name of the weekday from a numerical date representation.

### 3.2.1 Document Plan and Templates

We started by analyzing all the news pieces that made a match summary of all the first 21 rounds of Liga NOS 2015/2016 in order to create the structure of the generated news piece. We excluded all the news pieces that reported about more than one game because we noticed that they had very little information about each single game and were not very descriptive. In total, we collected 124 distinct match summaries. After analyzing them we made a document plan of the news piece to generate:

1. **Introduction:** First, the news piece will present the teams and the corresponding game final result. Sometimes the introduction will also make a reference to a particular player when he has a notable performance.
2. **Game Events:** After the introduction, the main game events (goals) are presented.
3. **Conclusion:** Lastly, the news piece will indicate the changes in classification of each team.

Win		Draw	
Home Team	Away Team	With Goals	Without Goals

Table 3.1: Possible outcomes of a football match that have distinct sentence templates.

After finishing the document plan, we started to build a collection of templates from the corpus. Starting with the introduction, we divided into four possibilities of introduction sentences according to the final result (Table 3.1). As we did not take user preference into account, the focus of the news piece will always be on the winning formation. A template for each case can be below:

- **Away team Win:**

```
artigo_equipa("O", away_team) + away_team + " deslocou-se ao reduto " + artigo_equipa("do",
home_team) + home_team + " para garantir a conquista dos 3 pontos com um triunfo por "
+ data["data"]["FINALRESULT"] + "."
```

- **Home team Win:**

```
artigo_equipa("O", home_team) + home_team + " recebeu e venceu, " + dia_jogo + ", " +
```

artigo\_equipa("o", away\_team) + away\_team + ", na partida da " + stage\_id + "ª jornada do campeonato."

- **Draw with goals:**

home\_team + " e " + away\_team + " empataram a " + home\_goals + " " + is\_plural("bola", int(home\_goals)) + ", na " + stage\_id + "ª jornada do campeonato, " + dia\_jogo + ".",

- **Draw without goals:**

home\_team + " e " + away\_team + " encontraram-se, " + dia\_jogo + ", em encontro para a " + stage\_id + "ª jornada do campeonato. O marcador manteve-se inalterado no final dos 90 minutos.",

After presenting the teams and the final result of the match, the introduction can also mention a notable performance of one player that plays on the winning team. The information provided by the input data is purely statistical. In football, a player can play very well without being directly linked to the game statistics. This kind of subjective analysis cannot realistically be generated by our system. Instead, we made a weight-based selection rule in order to find the player whose performance made the most impact on the final score. After analyzing the corpus, we noticed that a player that is directly linked to more than one goal is usually worth mentioning. For selecting the player to include in the output text, the rules listed in Table 3.2 were applied.

Player Stats	Added Weight
Goals	$GoalsScored \times 5$
Assists	$AssistsMade \times 3$

Table 3.2: Player stats and corresponding added weight to select the player with the most impact on the final score.

This selection rule is applied to all the players in the winning side. After finding the player who has the highest value, we will check if he is directly linked to two or more goals. If so, a sentence mentioning his performance will be generated in the output text. Below, examples of templates that could be generated are displayed.

- **Man of the match (Just Goals):**

Jogador + " foi o homem em destaque ao " + int\_to\_Goal(golos) + "na partida."

- **Man of the match (Just Assists):**

Jogador + " foi uma peça chave ao fazer " + str(assists) + " passes para golo."

- **Man of the match (Goals and Assists):**

Jogador + " foi o homem em destaque ao " + int\_to\_Goal(golos) + "e completar " + str(assists) + " ' ' + is\_plural("passe", assists) + ' para golo na partida.'



```

{
  "fk_equipa": "16",
  "numero": "21",
  "minutos": "90",
  "min_entrada": "0",
  "min_saida": "0",
  "fk_pais_jogador": "1",
  "fk_jogador": "480",
  "abrev": "João Pereira",
  "capitao": "0",
  "min_entrada_extra": "0",
  "min_saida_extra": "0",
  "amarelos": "",
  "vermelhos": "",
  "golos": "",
  "sgolos": "0",
  "tipo_equipa": "0",
  "titular": "1",
  "assists": "",
  "gpd": "",
  "gpf": "",
  "posicao": "2"
},

```

Figure 3.3: Information provided by the JSON file of a player.

Afterwards, we are going to write about the game events. GameRecapper only focuses on goal events for now. The JSON file provides information about the players stats in that particular game. Let us take Figure 3.3 as an example. For every player in the gamesheet we can check the player's name, how many goals he has scored and when they were scored.

For each goal, we collect the minute when it was scored, who scored it (game and team), who assisted and the current result of the game. This should be enough to describe a goal event but in order to achieve more variation of the generated news pieces we added different templates depending on additional information. We wanted GameRecapper to have distinct templates depending on two aspects: GameRecapper must recognize first goals, intermediate goals and last goals; and has to differentiate goal events depending on the impact of the goal event on the result. A goal can impact the result in 5 different ways:

1. **Increase an advantage** - When the team that scores the goal was already winning before that goal.
2. **Decrease an advantage** - When the team that scores the goal was losing by more than one goal.
3. **Draws the game** - When the team that scores the goal was losing by one goal.
4. **Breaks a draw** - When a team that during that game was never at a disadvantage, scores a goal that breaks a draw.
5. **Comeback on the game** - When a team that somewhere during the game was at a disadvantage, scores a goal that breaks a draw.

A first goal always breaks a draw. Therefore, GameRecapper has two distinct groups of templates for first goals: first goals that are also last goals, and those which are not last goals. The

following goals are either intermediate goals or last goals and they can impact the result in any of the 5 different ways described above. So, we made a group of templates for each impact on the result for either intermediate goals and last goals. This provides 10 different groups of sentence templates for the following goal events, since the sentence templates with the same impact on the current result are distinct if they are talking about an intermediate goal or a last goal. An additional sentence template was made for the scoreless games. In total, GameRecapper contains 13 different groups of sentence templates for goal events. This provides much more variation for the generated output.

Finally, the output text must specify the changes in classification for both teams. The JSON file contains the classification of each team before and after the game. We divided the sentence templates into three groups depending on the final score (home team win, away team win or draw). Examples of possible outputs are shown below.

- **Away team win:**

Com o triunfo, a turma de Manuel Machado subiu ao 9º lugar e passa a somar 34 pontos. Por seu lado, a equipa de Sérgio Conceição continua com os 34 pontos com que entrou para a 27.<sup>a</sup> jornada e desceu para o 10º lugar.

- **Home team win:**

Com este resultado, a Académica subiu ao 16º lugar e passa a somar 22 pontos. Já a equipa de Sérgio Conceição continua com 34 pontos e mantém o 7º lugar.

- **Draw:**

Com este empate, o U. Madeira soma 26 pontos e mantém o 15º lugar, enquanto o V. Setúbal passa a ter 29 pontos e mantém o 13º lugar.

Table 3.3 shows how many sentence templates GameRecapper has and their divisions. In total, 88 distinct sentence templates were built. In order to get a better view on how much variation the output text can achieve, a table that calculates how many news pieces can be generated depending on the final result was made (Table 3.4), for games with four or less goals. This calculation provides a minimum value since we did not take into account the reference to the player whose performance make the most impact on the final result.

As we can see, with this approach, we can achieve a good amount of variation on the generated output except for the scoreless game. It is hard to add variation on a game without goals since the information provided by the JSON tree structure is about major game events. Even if GameRecapper provided information about other events like sent offs and penaltis awarded, this might not increase the variation on a scoreless game.

### 3.2.2 Generation Module Algorithm

The Generation Module was written in Python language. We chose Python because we think it is excellent for processing linguistic data thanks to the good string-handling functionality. The

Introduction				
Win		Draw		
Home Team	Away Team	With Goals		Without Goals
12	9	4		5
First Goal				
Without Goals		Total Goals = 1	Total Goals > 1	
1		4	10	
Intermediate Goal				
Increases	Decreases	Breaks Draw	Draws Game	Comeback
7	4	4	7	2
Last Goal				
Increases	Decreases	Breaks Draw	Draws Game	Comeback
2	2	4	4	2
Conclusion				
Home Team Win		Draw	Away Team Win	
2		1	2	

Table 3.3: GameRecapper sentence templates.

generation module starts by asking which games the user wishes to generate a summary and how many news pieces the user wants for each game. Then, the algorithm starts reading the JSON file and checks the final result of the game in order to begin the generation of the introduction. After checking the final result, the generation module randomly chooses one of the sentence templates associated to that final result and writes it in a text file. If the game ended with a victory for any of the teams, the weight-based selection rule will be applied to all the players in the winning team. After finding the player who has the highest value, it checks if that value is equal or greater than 8 (this means he is directly linked to two or more goals) and, if so, a sentence template will be

	Away Goals			
	0	1	2	3
Home Goals				
0	5	72	360	2520
1	96	160	10080	42840
2	480	13440	11200	-
2	480	13440	11200	-
3	3360	57120	-	-

Table 3.4: Distinct news pieces that GameRecapper can generate for a specific final result.

randomly picked and written in the text file.

After the introduction is finished, the algorithm will write about the game events. As stated before, GameRecapper only takes into account the goal events for now. To do that, the generation module starts reading the JSON file and creates four lists:

- A list with the name of the goalscorers.
- A list with the time when the goals were scored (in minutes).
- A list with the scoring team (1 = home team, 2 = away team).
- A list with the names of the players who assisted for the goals (if the goal was assisted by someone writes the name of who assisted and if no one assisted does not write anything).

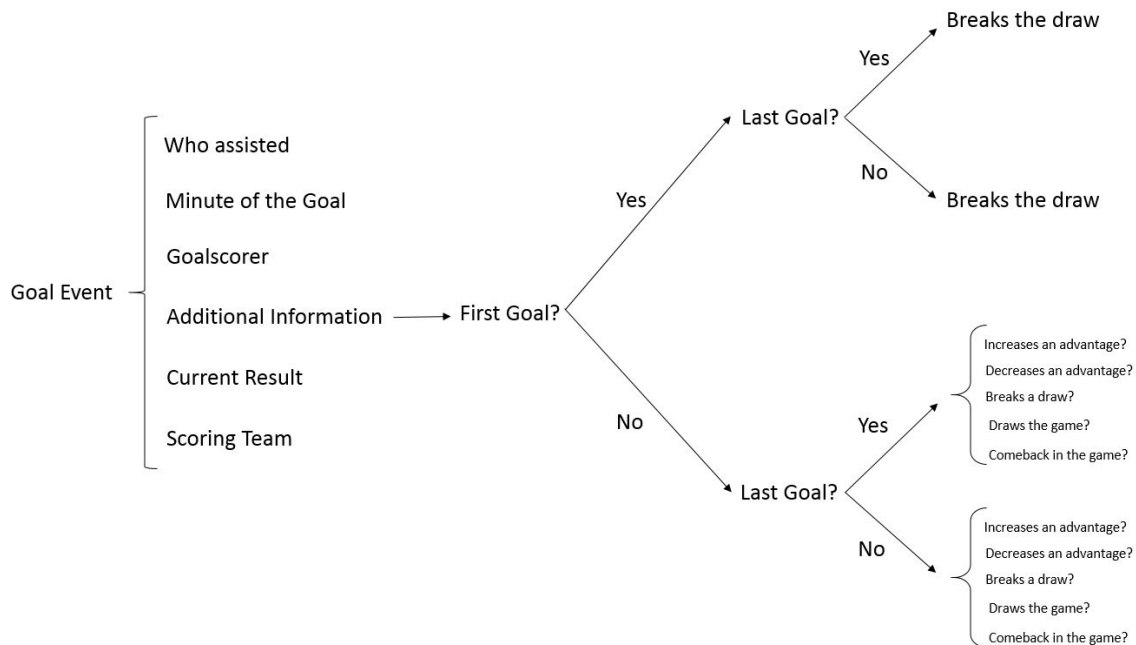


Figure 3.4: Procedure done by the Generation Module for each goal event.

After the lists are chronologically ordered, we can start the procedure to describe the goals of the game. For each goal in the list, Figure 3.4 indicates the process done by the Generation Module to describe a goal event. If the game has no goal events, the sentence template that represents a scoreless game will be automatically picked. For games with one or more goals, the algorithm first reads the first element on each list (information about the first goal) and will randomly select a sentence template depending on whether the first goal is also the last goal or not. Before progressing to the next goal event, the algorithm updates the current score of the game. For the following elements on the list that are not the last goals, the Generation Module will assess the impact of that goal on the current result of the game. Then, it will randomly pick an intermediate

goal sentence template corresponding to that goal event impact on the current result and update the current result. For the last element on the list (last goal of the game), the procedure is quite the same, but, instead of picking an intermediate goal sentence template, it will randomly choose a final goal sentence template depending on the impact of the last goal on the final result.

Before the match summary is completed, we need to describe the classification changes. As stated in the previous subsection, we divided the sentence templates into three groups depending on the final score (home team win, away team win or draw). As we already know the final result, we just need to randomly pick an according sentence template. These sentence templates make use of two functions to express the changes in the classification. The first one checks if the teams ranked up, ranked down or maintained their position in order to express it in the output. The other function expresses how the sentence must start: if Team A wins the game and ranks up or maintains the classification the output will be: "With this victory, Team A was able to maintain the 5th position.." but if Team A wins and goes down on the classification the output should be: "Despite winning, Team A dropped to the sixth place...".

This procedure is repeated as many times as the selected number of match summaries to generate, for the same game. If the user selected more than one game (the algorithm can generate a match summary for every game in a round), the Generation Module just closes the JSON file after finalizing all the match summaries for that game and starts reading the following game JSON file.

### 3.2.2.1 Generation Module Algorithm Example

We will now illustrate the generation module algorithm with an example. Let us take the gamesheet from the **Académica vs Benfica** game provided in Figure 3.5. For simplicity purposes, we will not mention any data pre-processing. First, the algorithm opens the JSON file and checks the goals scored by the home team and by the away team. The number of goals scored by the away team (Benfica) is bigger than the home team (Académica) so, the algorithm will randomly choose an introductory away team victory sentence template. Let us suppose that the sentence chosen was:

#### Introduction sentence template :

```
artigo_equipa("O", away_team) + away_team + " foi ao " + estadio + ", " + artigo_cidade("em", cidade) + cidade + ", derrotar " + artigo_equipa("o", home_team) + home_team + " por " + data["data"]["FINALRESULT"] + ", em jogo da " + stage_id + "ª jornada do campeonato."
```

The algorithm will check the JSON file and save the name of the home (Académica) and away (Benfica) teams in separate variables, and the round number (29) is saved as the *stage\_id*. After that, the algorithm will check in the domain data the names of the stadium of Académica and the city where the stadium is located. The function **artigo\_equipa** is a linguistic function that checks the gender (masculine/feminine) of the team provided as argument in order to choose the right definite article. The same applies to **artigo\_cidade**, but this function checks the gender of the city of the home team. After filling the slots, the sentence template is transformed into:

FICHA DO JOGO	
<b>ACADÉMICA</b>	
88  Pedro Trigueira	
43  Nii Plange	90+1'
5  Ricardo Nascimento	21'
13  João Real	
14  Iago Santos	45'
55  Rafa Soares	
28  Nuno Piloto	
27  Pedro Nuno	17' 28' 65'
7  Marinho	73'
9  Rabiola	87'
30  Rafael Lopes	
<b>SUPLENTES</b>	
32  Lee Oliveira	
2  Aderlan Silva	87'
23  William Gustavo	
20  Rui Pedro	65'
48  Artur Taborda	
77  Hugo Seco	73'
10  Ivanildo	
<b>TREINADORES</b>	
Filipe Gouveia	
<b>BENFICA</b>	
1  Ederson Moraes	
34  André Almeida	85'
14  Victor Lindelöf	
33  Jardel	
19  Eliseu	80'
7  Andreas Samaris	71'
21  Pizzi	39' 61'
85  Renato Sanches	
10  Nico Gaitán	
11  Kostas Mitroglou	39'
17  Jonas	
13  Paulo Lopes	
4  Luísão	
5  Ljubomir Fejsa	
30  Anderson Talisca	71'
18  Eduardo Salvio	
39  Mehdi Carcela	61'
9  Raúl Jiménez	80' 85'
<b>TREINADORES</b>	
Rui Vitória	

Figure 3.5: Match information of Académica vs Benfica provided by *www.zerozero.pt*.

### Introduction sentence template after filling the slots :

*O Benfica foi ao Estádio EFAPEL, em Coimbra, derrotar a Académica por 1-2, em jogo da 29ª jornada do campeonato.*

After writing the sentence into the text file, the algorithm will apply the selection rule to all the players of Benfica. None of the Benfica players participated in two goals, so every player will have an added-weight lower than 8 and no sentence will be written about the player with the most impact on the final result. Next, the generation module will start to write about the goal events. Before writing the sentences, the algorithm creates 4 lists:

- **Goalscorer list** : [ Pedro Nuno; Kostas Mitroglou; Raúl Jimenez];
- **Minutes list** : [ 17; 39; 85];
- **Scoring team list**: [ 1; 2; 2];
- **Assist list** : [ - ; Pizzi; André Almeida]

Then, the algorithm starts a by reading the first element of each list. The current result is initiated as 0-0 and the algorithm starts the procedure depicted in Figure 3.4. After checking that the first goal was not the last goal, the generation module chooses randomly a sentence according to that. The sentence chosen was:

**First goal sentence template :**

"Os homens de " + TREINADOR\_MARCOU + " chegaram ao golo aos " + minuto\_golo + " minutos, numa finalização do " + POSICAO + " " + marcador + who\_assisted(minuto\_golo) + "."

The variable *TREINADOR\_MARCOU* is filled by checking the coach name of the scoring team (the scoring team is provided by the scoring team list) in the JSON file. *minuto\_golo* is the first element of the minutes list, *marcador* is the first element of goalscorer list. The JSON file provides the position of the player and is represented by a number ( 1 - goalkeeper until 4 - striker). *POSICAO* calls a linguistic function that translates into a word the position of the goalscorer (midfielder in this case), given the number provided by the JSON file (3). No one assisted for the first goal so the function *who\_assisted* returns null.

**First goal sentence template after filling the slots :**

*Os homens de Filipe Gouveia chegaram ao golo aos 17 minutos, numa finalização do médio Pedro Nuno.*

After writing the sentence in the text file, the current result is updated to 1-0. The algorithm will continue and starts to look at the second element of the lists. Now, we are in the presence of an intermediate goal that is not the last goal. This is the second goal of the game and the scoring team of the second goal is different from the scoring team of the first goal. Therefore, this is a goal that draws the game. The selected sentence is, for example:

**Intermediate goal that draws the game sentence template :**

"Aos " + minuto\_golo + " minutos, " + marcador + who\_assisted (minuto\_golo) + " disparou para o golo do empate."

This time, the assist list element is different from null, so the *who\_assisted* function will return **", depois de uma assistência de " + assist +",** where assist is the second element of the assist list. The sentence after filling the slots is:

**Intermediate goal that draws the game sentence template after filling the gaps :**

*Aos 39 minutos, Kostas Mitroglou, depois de uma assistência de Pizzi disparou para o golo do empate.*

The generation module writes the sentence in the text file, updates the current score to 1-1 and starts reading the last element of the lists (last goal). The number of goals scored until now is 2. As the number of current goals scored by the home team is equal to the number of current goals scored by the away team, if the third element of the scoring team list is equal to the second element, this goal makes a comeback in the game. So the algorithm will randomly select a sentence according to that:

**Last goal that makes a comeback sentence template :**

"A cambalhota no resultado acabou mesmo por acontecer aos " + minuto\_golo + " minutos, com " + marcador + who\_assisted(minuto\_golo) + ", a ser o marcador de serviço e o responsável pelo " + data["data"]["FINALRESULT"] + " final."

**Last goal that makes a comeback sentence template after filling the slots :**

*A cambalhota no resultado acabou mesmo por acontecer aos 85 minutos, com Raúl Jiménez, depois de uma assistência de André Almeida, a ser o marcador de serviço e o responsável pelo 1-2 final.*

Finally, we will write about the classification changes of the teams. As mentioned before we are in the presence of an away team victory so the generation module selected the following sentence:

**Last goal that makes a comeback sentence template :**

```
begConclusion(CLASSIFAWAY_PRE, CLASSIFAWAY_POS) + "este resultado, " + artigo_equipa("o",
away_team) + away_team + " " + changeClassification(CLASSIFAWAY_PRE, CLASSIFAWAY_POS)
+ " e passa a somar " + POINTSAWAY + " pontos. Já a equipa de " + data["data"]["MATCHREPORT"]
["treinador_casa"]["abrev"] + " continua com " + POINTSHOME + " pontos e " + changeClassi-
fication(CLASSIFHOME_PRE, CLASSIFHOME_POS) + "."
```

The JSON file provides the classification of both teams before and after the game and the total points of each team after the game. For example, for the away team we save those values in the variables *CLASSIFAWAY\_PRE*, *CLASSIFAWAY\_POS* and *POINTSAWAY*. The function *begConclusion*, takes as input the classifications of the winning team before and after the game. If the classification after the game is equal or higher than the classification before the game, the function returns: “Com ”, otherwise the function returns: “Apesar d”. Then, the function *changeClassification* lexicalizes the classification change to the reader. If the classification after the game is higher, it returns “subiu ao ”; if the classifications are equal, it returns “mantém o ”; and if is lower, it returns “desceu para o”. In this case, the sentence template turns into:

**Last goal that makes a comeback sentence template after filling the slots :**

*Com este resultado, o Benfica mantém o 1º lugar e passa a somar 73 pontos. Já a equipa de Filipe Gouveia continua com 23 pontos e mantém o 17º lugar.*

After writing the sentence to the text file, the summary is complete. This process can be repeated as many times as the number of summaries the user selected to generate. Below, the complete summary of Académica vs Benfica is presented.



**Complete Summary of Académica vs Benfica :**

*Benfica foi ao Estádio EFAPEL, em Coimbra, derrotar a Académica por 1-2, em jogo da 29ª jornada do campeonato. Os homens de Filipe Gouveia chegaram ao golo aos 17 minutos, numa finalização do médio Pedro Nuno. Aos 39 minutos, Kostas Mitroglou, depois de uma assistência de Pizzi disparou para o golo do empate. A cambalhota no resultado acabou mesmo por acontecer aos 85 minutos, com Raúl Jiménez, depois de uma assistência de André Almeida, a ser o marcador de serviço e o responsável pelo 1-2 final. Com este resultado, o Benfica mantém o 1º lugar e passa a somar 73 pontos. Já a equipa de Filipe Gouveia continua com 23 pontos e mantém o 17º lugar.*



## Chapter 4

# Evaluation of GameRecapper

In this chapter, we present the methodology followed to evaluate GameRecapper and discuss the corresponding experimental results.

### 4.1 Methodology

As stated in Subsection 2.5, there are multiple possibilities when it comes to evaluate an NLG system. Our focus was to evaluate the quality of the produced text and to compare how users perceive a GameRecapper summary versus a human authored summary. For the evaluation of GameRecapper we opted to do a manual evaluation. The reason why we did not use an evaluation metric, such as BLEU and ROUGE, was that we do not think it is appropriate to use a word-by-word/phrase-by-phrase comparison on a system that uses sentence templates extracted from a corpus. This would give high scores to our system that might be inaccurate since the output sentences would be exactly like the ones on our corpus (our reference text). We could use the summary that was actually published online in *www.zerozero.pt* as our reference text, but a significant portion of human authored summaries uses background knowledge that we do not have access in the input data. In sports journalism, it is important to achieve variation in the output text. Our purpose is not to generate a summary exactly like a human authored summary, but to have a system that with some kind of variation can generate a coherent, fluid and accurate text with the same communicative goal.

As the corpus used for creating the sentence templates was built by analyzing all the news of the first 21 rounds of Liga NOS 2015/2016 that reported about one and only one game, we chose to generate summaries for all the games from the round 25 until the round 29. In total, 44 match summaries were generated. The methodology adopted to assess the quality of the produced text was based on the evaluation methodology done by Bouayad-Agha et al. to evaluate the quality of the text output of their system [7]. The 44 match summaries were divided between 3 surveys to assess the quality of the produced text. The division was not random as we wanted each survey to have as many final results as possible. Table 4.1 shows the final results and how many times they happened in the games from the round 25 until the round 29. Those final results were divided

Final Result		Number of Games with the same final result
Home Team Goals	Away Team Goals	
0	0	3
1	0	5
0	1	10
2	0	4
1	1	2
1	2	2
3	0	1
0	3	2
3	1	2
1	3	1
2	2	4
3	2	3
4	1	2
5	1	2
2	5	1

Table 4.1: Distinct final results contained between the rounds 25 until 29 and how many times that final result occurred.

as depicted in Table 4.4. Fifteen evaluators from the *www.zerozero.pt* newsroom participated in the evaluation, such that each survey was evaluated by 5 different evaluators. The evaluators were asked to rate the match summaries along two criteria: intelligibility (Table 4.2) and fluidity (Table 4.3).

To compare how the users perceive a GameRecapper summary versus a human authored summary, another survey was made. The survey included ten randomly selected match summaries, from distinct games, from which five were generated by GameRecapper and the other five were the match summaries published online on *www.zerozero.pt*. The survey did not say if the match summary was generated by an algorithm or written by a journalist. Forty-six evaluators previously not involved in the project participated in the evaluation. The survey provided the gamesheet of the match and the evaluators were asked to rate the match summaries according to its accuracy/completeness and to its readiness to be published online (Figure 4.1).

Rate	Meaning
$\frac{1}{5}$	The sentences are not perceivable.
$\frac{2}{5}$	The sentences have major grammatical and/or lexical errors. Only with a lot of effort it is possible to deduce the meaning of the sentences.
$\frac{3}{5}$	The meaning of most sentences is clear, even though some parts are not as clear because of grammatical and/or lexical choices.
$\frac{4}{5}$	The meaning of all sentences is clear, but there are minor problems in some grammatical and/or lexical choices.
$\frac{5}{5}$	The meaning of all sentences is clear and all grammatical and/or lexical choices are appropriate.

Table 4.2: Meaning of each intelligibility rating (Source: [7]).

Rate	Meaning
$\frac{1}{5}$	The summary is unreadable.
$\frac{2}{5}$	The summary is difficult to read, but it is possible with some effort.
$\frac{3}{5}$	The summary is not too difficult to read, but there are some details that seem unnatural.
$\frac{4}{5}$	The summary is easy to read, even though there are some boring repetition of information.
$\frac{5}{5}$	The summary is extremely easy to read, it seems perfectly natural.

Table 4.3: Meaning of each fluidity rating (Source: [7]).

Survey 1	Survey 2	Survey 3
0-0	0-0	0-0
0-1	0-1	0-1
0-1	0-1	0-1
0-1	0-1	0-1
0-1	1-0	1-0
1-0	1-0	1-0
2-0	2-0	2-0
2-0	1-1	1-1
1-2	1-2	3-0
2-2	0-3	0-3
2-2	2-2	2-2
1-3	3-1	3-1
3-2	3-2	3-2
4-1	4-1	2-5
5-1	5-1	-

Table 4.4: Distribution of games between the surveys according to the final result

For accuracy/completeness the users rated within a 5 point rate scale the following question:

**Does the following text does a complete and accurate match summary of the game?**

- **Min (1/5):** No, the lack of relevant and key information makes this summary really incomplete.
- **Max (5/5):** Yes, the summary is very complete and accurate. All the key information of the game is provided by the summary.

For the readiness to be published online a simple yes or no question was provided:

**Do you think this match summary is ready to be published online?**

- **Yes:** Yes, this match summary does not need any kind of edition to be published.
- **No:** No, this match summary need to be edited by a journalist.

**Geração Automática de Notícias de Desporto**

Dada a ficha de jogo:  
<http://www.zerozero.pt/match.php?id=4516818>

E o seguinte sumário:

**SC Braga 2-0 União da Madeira**

Em Braga, a equipa local levou a melhor sobre o U. Madeira, vencendo por 2-0. Josué foi uma peça chave ao marcar 1 golo e juntando a isso 1 assistência.  
 Josué, aos 36, colocou a formação de Paulo Fonseca na frente do marcador.  
 A equipa dos arsenalistas fixaria o resultado final em 2-0 com um golo de Nikola Stojiljkovic (50').  
 Com o triunfo, a turma de Paulo Fonseca mantém o 4º lugar e passa a somar 50 pontos.  
 Por seu lado, a equipa de Luís Norton de Matos continua com os 25 pontos com que entrou para a 27.ª jornada e mantém o 15º lugar.

Na sua opinião, a notícia faz um sumário completo e correto do jogo? \*

	1	2	3	4	5	
Não, a falta de informação chave e relevante torna o sumário muito incompleto.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Sim, o sumário é muito completo. Toda a informação-chave é apresentada na notícia.

Na sua opinião, a notícia encontra-se preparada para ser lançada numa peça de jornalismo online? \*

☐ Não, ainda é necessário ser editada.

☐ Sim, não parece ser necessário qualquer tipo de edição.

Figure 4.1: Image of the GameRecapper summaries vs human-authored summaries survey.

## 4.2 Results and Discussion

### 4.2.1 Evaluation of GameRecapper's Output Text

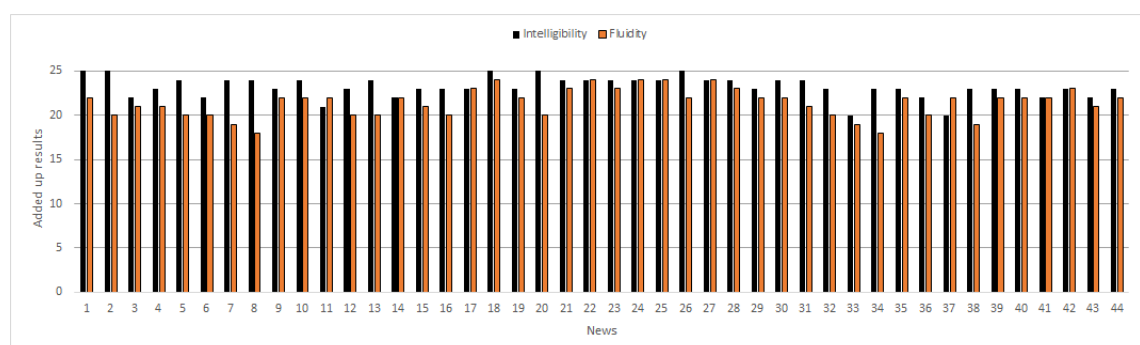


Figure 4.2: Results of text quality evaluation.

The added-up results are shown in Figure 4.2, giving a maximum grade of 25 for each of the 44 match summaries. The intelligibility score of the summaries averages  $\frac{4.645}{5}$ , which represents a score of 92.91%. As we can see in Table 4.5, five summaries have an average intelligibility rating of  $\frac{5}{5}$  (100%). There are actually only 9 summaries for which the average score is below  $\frac{4.5}{5}$  (90%), and there was not a summary that scored below  $\frac{4}{5}$  (80%). The fluidity average score of the summaries is 85.73%, quite lower than the intelligibility average score. None of the summaries hit the maximum fluidity average score. Despite that, only 5 summaries average a score below 80% and the minimum average score was 72%. The results suggest that GameRecapper makes few inappropriate grammatical and lexical choices, and the quality of the summaries is generally good. The superior score of intelligibility compared with the fluidity score might indicate that, even though the sentences are well-written, sometimes there are some unnatural details or boring repetitions of information. A deeper analysis will be presented below, so we can correlate the influence of multiple parameters on the intelligibility and fluidity average score.

	Intelligibility	Fluidity
Number of summaries that scored 100%	5	0
Number of summaries that scored lower than 90%	9	34
Number of summaries that scored lower than 80%	0	5
Minimum score of a summary (%)	80%	72%
Average Score (%)	92.91%	85.73%

Table 4.5: Additional information on the results of text quality evaluation.

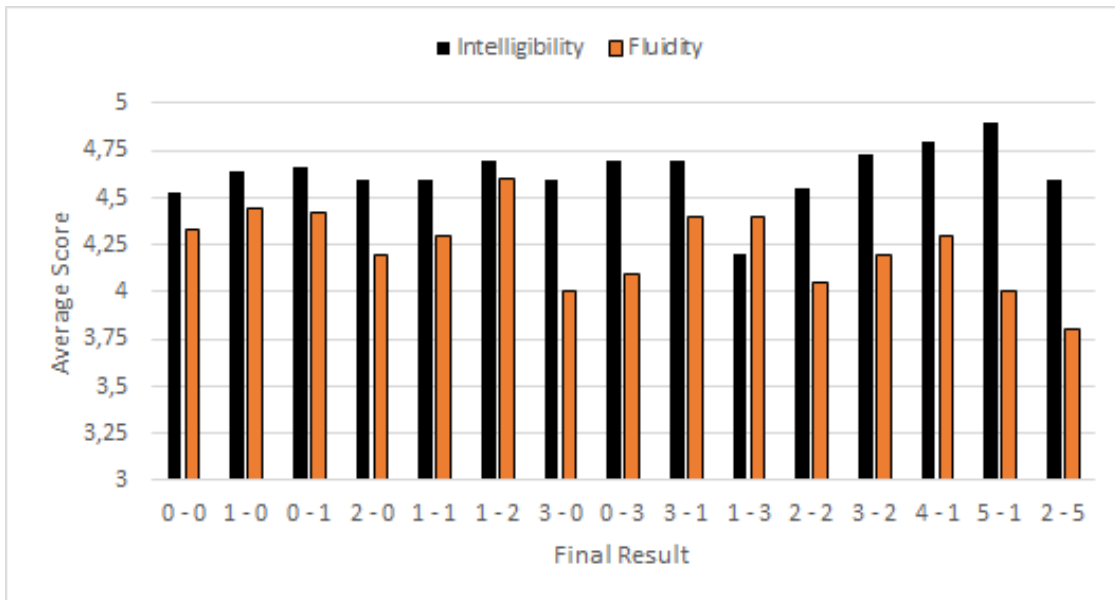


Figure 4.3: Intelligibility and Fluidity average score for each final result.

First, we checked the average score for every final result presented in Table 4.1. The results are shown in Figure 4.3. Despite not being crystal clear, it seems that an increasing of the number



of goals scored in a game has a negative impact on the fluidity of the text. The intelligibility of the summary seems to remain quite unchanged, except for the **1-3** final result that only has one sample. Therefore, two additional charts were made: one correlating the effect of the number of goals scored in a game with the intelligibility/fluidity average score (Figure 4.4) and another with the final outcome of the game (Figure 4.5).

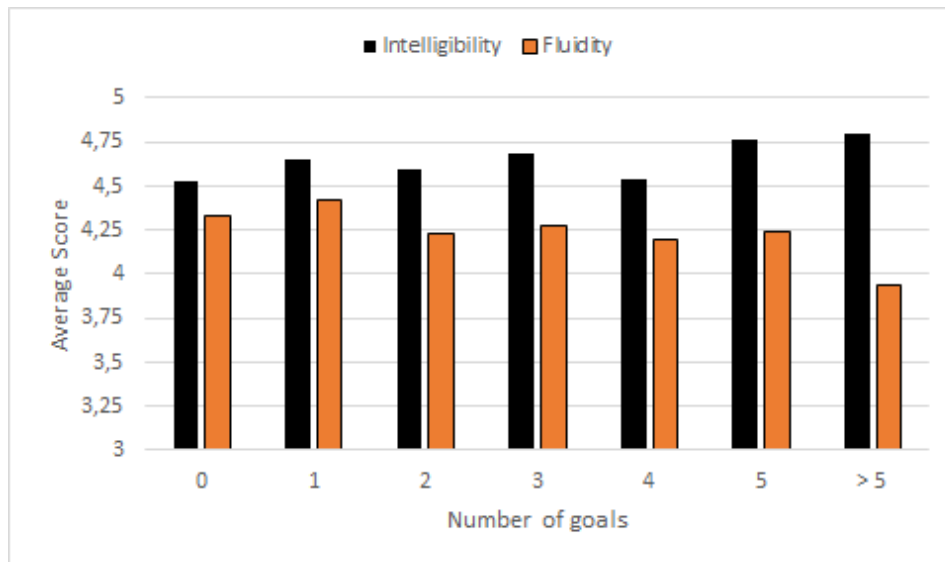


Figure 4.4: Intelligibility and Fluidity average score according to the number of goals scored in a game.

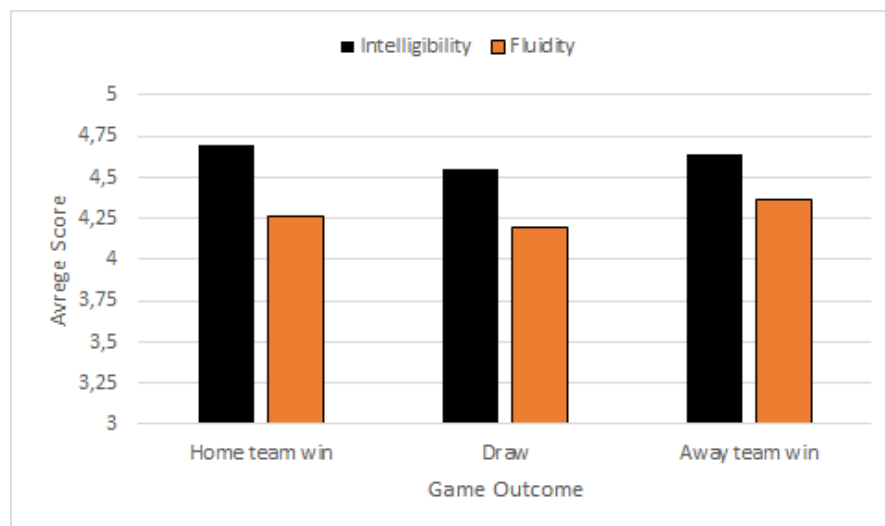


Figure 4.5: Intelligibility and Fluidity average score according to the final outcome of the game.

Figure 4.4 shows an interesting feature. An increasing number of goals seems to increase the Intelligibility score and to decrease the Fluidity score. The more goals are scored in a game, the more sentence templates for goal events are used. As GameRecapper does not perform sentence

aggregation and referring expression generation tasks, the more sentence templates used for goal events makes the possibility of repetition of information increase as well. Therefore, it seems reasonable that the fluidity average score tends to decrease with the number of goals scored. Repetition of information is boring and makes the text seem less natural and that is why journalists try to avoid it at all costs. This also suggests that not only the number of goals scored in a game have a negative impact on the fluidity of the text, but also the number of similar goal event sentence templates used. So, games that have a lot of goals with the same impact on the result might also decrease the fluidity of the text. With respect to Figure 4.5, it seems that the final outcome does not have an impact on the summaries' intelligibility and fluidity. To assess the possibility of the negative impact of similar sentence templates used for goal events a chart was made (Figure 4.6). The easiest way of grouping together the games with more similar sentence templates used is to check the intelligibility/fluidity score according to the goal difference between the winning team and the losing team. As expected, the fluidity of the summary seems to linearly decrease from the one goal difference games to the four goal difference games. A four goal difference game makes use of a lot of "increasing the lead" sentence templates which makes the summary have more similar information and consequently look less natural.

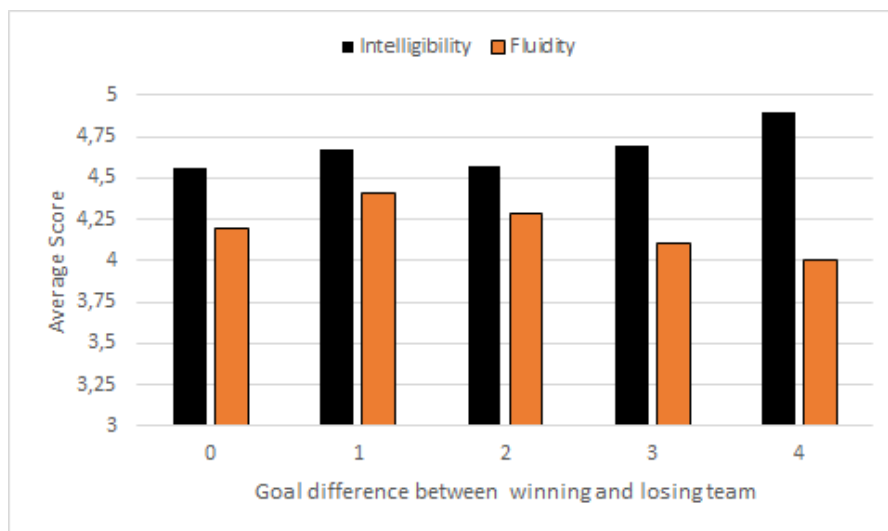


Figure 4.6: Intelligibility and fluidity average score according to the goal difference between the winning and losing teams.

In conclusion, the results indicate that GameRecapper produces an acceptable text output. The intelligibility and fluidity of the summaries is fairly high (92.91% for intelligibility and 85.73% for fluidity). We also verified that the intelligibility average score remained quite unchanged despite of the number of goals, final outcome and goal difference between winning and losing team of the game. In relation to the fluidity average score, we observed a certain correlation between the fluidity of a text and the number of goals scored in a game as well as for the goal difference between the winning team and the losing team. The size of GameRecapper's summaries is proportional to the number of goals scored in a game (GameRecapper writes a sentence for each goal event) which

makes the summary of high-scoring games more propitious to repetitions of information. The fact that the games with bigger goal differences between the winning and losing teams scored lower in fluidity is due to game summaries with bigger goal difference make use of more similar sentence templates. The lack of a sentence aggregation and a referring expression generation tasks makes the text seem less natural because it is more prone to repetition of information.

#### 4.2.2 GameRecapper Summary versus Human-Authored Summary

The results of the evaluation of GameRecapper summaries versus the summaries written by journalists are depicted in Table 4.6.

	GameRecapper Summary				
	Game 1	Game 2	Game 3	Game 4	Game 5
Number of words	218	125	174	115	173
Completeness	86.38%	80.00%	87.24%	68.94%	79.58%
Readiness	89.13%	65.22%	78.26%	67.39%	65.22%

---

	Human-Authored Summary				
	Game 6	Game 7	Game 8	Game 9	Game 10
Number of words	125	195	310	441	185
Completeness	82.98%	85.96%	91.91%	91.91%	85.53%
Readiness	67.39%	82.61%	93.48%	71.74%	91.30%

Table 4.6: Scores of GameRecapper and human-authored summaries on completeness and readiness to be published online.

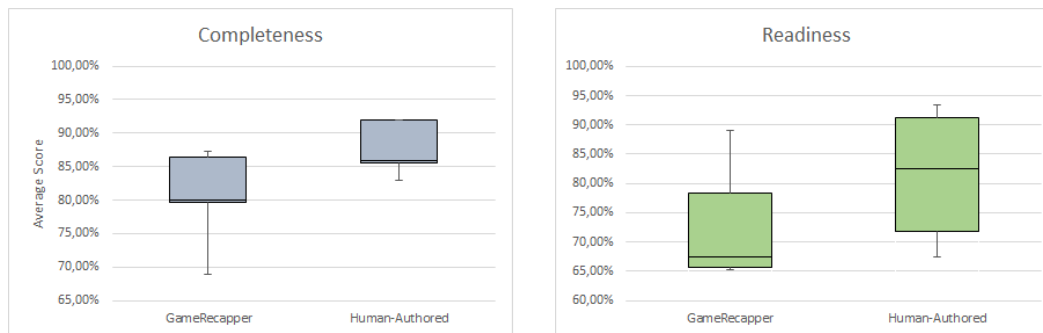
The completeness of the GameRecapper summaries averages  $\frac{4.021}{5}$ , which represents a score of 80.42% against the  $\frac{4.38}{5}$  (87.66%) average score of the summaries written by journalists (Table 4.7). With respect to the summaries' readiness to be published online, GameRecapper has an average score of 73.04% and human-authored summaries of 81.30%. These results are fairly surprising since GameRecapper summaries are being evaluated against news pieces that were actually released online. Despite the evaluators recognized the human-authored summaries as more complete/accurate and more ready to be uploaded online, the difference between them is considerably low.

	GameRecapper Summaries	Human-Authored Summaries
Completeness	80.42%	87.66%
Readiness	73.04%	81.30%

Table 4.7: Average scores of GameRecapper and human-authored summaries

In order to have a better perception of the results, two box plot charts comparing the GameRecapper and the human-authored summaries were created (Figure 4.7). Box plot is a graphic that

identifies where 50% of the most probable values are located, the median and the extreme values of a data set. Figure 4.7a show us that the average scores of three summaries of GameRecapper are within a 6.38% interval and the best average score is 0.85% higher than the second best. The worst average score is 10.7% lower than the second worst. Checking the gamesheet of the summary which has the lowest average score, we verified that the game had a sent-off and GameRecapper does not write about sent-off events. This assessment means that the evaluators think that a sent-off should be referenced in the summary. All the completeness average scores of the summaries written by journalists are within a 8.51% interval which means the degree of dispersion is low.



(a) Completeness - GameRecapper vs Human-Authored.

(b) Readiness - GameRecapper vs Human-Authored.

Figure 4.7: Distribution of the average scores of the summaries according to the completeness and readiness criteria.

With respect to Figure 4.7b, we can analyze that the readiness of GameRecapper summaries is being benefited by the readiness of Game 1 which has an average score 10,87% higher than Game 3 (the second best readiness average score). The latter also scores 10,87% higher than Game 4 (the third best). The three summaries with the lowest average scores are within a 2.17% interval which means the readiness average score of GameRecapper (the 73.04% in Table 4.7) is not a quite accurate value for the readiness of GameRecapper. As for human-authored summaries, the readiness box plot indicates that the degree of dispersion is quite high since the difference between the two quartiles (the second and fourth best average score) is almost 20%. An interesting fact is, despite Game 9 scoring 91.91% in the completeness criteria, only 71.74% think it is ready to be released online. Game 9 is by far the longest summary with a total of 441 words. To verify the impact of the length of a summary in the completeness and readiness of a summary, two more charts were made (Figure 4.8, 4.9). We can check that the length of a summary seems to increase the completeness and readiness average score. The only exception is in human-authored summaries with more than 200 words decrease 5% in relation to human-authored summaries with a length between 150 and 200 words. This decrease is due to Game 9 average score in readiness since Game 8 averages 93.48% in that criteria. Checking the news piece of Game 9 we can check that despite the length of the summary, the news piece does not have much detail about the goals. Below, we present the description of the first two goal events of the human-authored summary and the description of GameRecapper for the same goals in the generated text presented in the survey

to assess the quality of the text output:

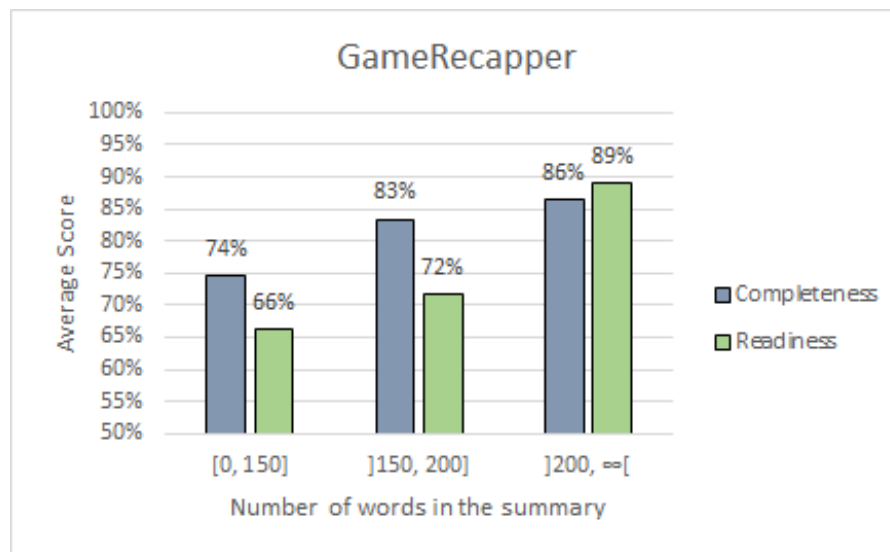


Figure 4.8: Impact of the length of a summary in the completeness and readiness criteria (GameRecapper summaries).

**Human-authored summary** - "Jardel (10') e Jonas (24') fizeram os dois primeiros golos do Benfica."

**Translation** - "Jardel (10') and Jonas (24') scored the first two goals of Benfica."

**GameRecapper summary** - "Os homens de Rui Vitória chegaram ao golo aos 11 minutos, numa finalização do defesa Jardel, após passe de Nico Gaitán. O segundo golo da equipa da casa chegaria aos 24 minutos, com Jonas a fazer o 2-0, depois de uma assistência de Nico Gaitán."

**Translation** - "Rui Vitoria's team scored the first goal at the 11th minute, in a finish by the defender Jardel, after a pass by Nico Gaitán. The second goal of the home team was reached at the minute 24, with Jonas making the 2-0, after Nico Gaitán assist."

As we can assess, despite being a long summary, this human-authored summary is not really descriptive which makes the readiness criteria average score decrease. Regardless of being a complete summary, the evaluators think that with the summary may be improved with some edition. To sum up, the GameRecapper summaries versus human-authored summaries evaluation let us know that the evaluators think that the GameRecapper generated output makes a complete and accurate summary of the game when the game has no sent-offs. Except for the low completeness average score of Game 4, all the other summaries have an average score above 80% and the average score of all summaries is just 7.24% below the summaries published online at the [www.zerozero.pt](http://www.zerozero.pt) website. Despite this, three of the GameRecapper summaries scored between 60% and 70% for

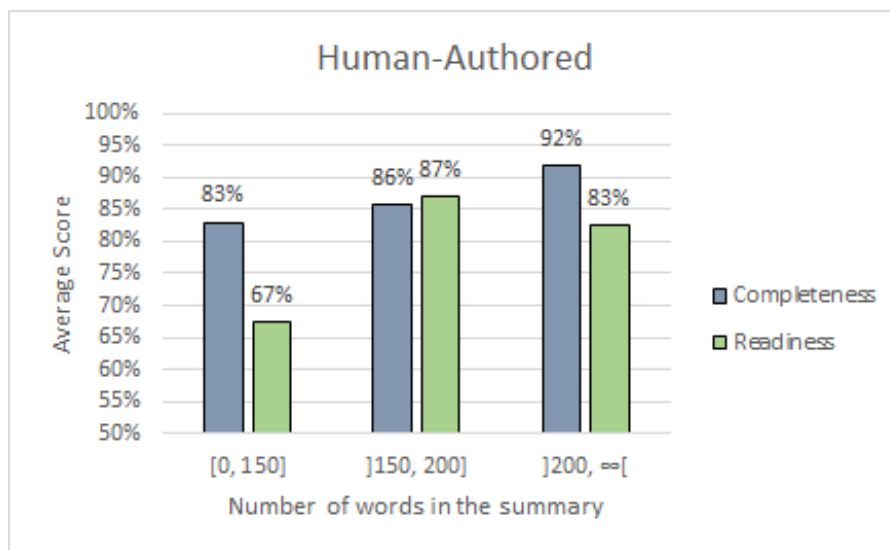


Figure 4.9: Impact of the length of a summary in the completeness and readiness criteria (Human-authored summaries).

the readiness to be released online. This suggest that GameRecapper generated output should be reviewed and edited before being a valid news piece option.

## Chapter 5

# Conclusion

### 5.1 Summary

The main purpose of this dissertation was to implement automatic generation of sports news pieces. Today, there is a need to report on a lot of football matches, and, many of them, are played at the same time. A significant amount of human resources and working time would be needed for journalists to watch every match they have to make a report. Due to the abundant availability of information that is stored in databases, journalists are able to make a report based just on that information. However, if this process was automated, it would save a lot of working time of journalists that they could spend it on in-depth reporting for example.

With this in mind, in this dissertation, we presented the GameRecapper, a template-based system that generates Portuguese summaries of football games from structured input data, i.e. a gamesheet of the *www.zerozero.pt* website. After reviewing the literature, we implemented the system based on the text generation module of a previous NLG system with the same domain and communicative goal, the GoalGetter system. GameRecapper has a Generation module which is the basic algorithm that creates the news. The Generation Module makes use of domain data, linguistic functions, grammatical functions and a collection of sentence templates. Domain data provides additional information about the teams in order to achieve more variation in the output text. Linguistic functions translate some numerical data into words (e.g. getting a week day from a numerical date representation) and grammatical functions ensure the coherence of the text. The collection of sentence templates was manually built from an initial corpus written by actual journalists from a newsroom. Each template contains open slots for variable information. These sentence templates were divided into groups according to goal events of the game and to the characteristics of the game. GameRecapper's ability of knowing the impact of a goal event on the result allowed us to achieve a significant amount of variation on the generated summaries.

As stated in Subsection 2.5, there is not a well-defined method when it comes for evaluating a system. In any NLG system, it is as important to evaluate the quality of the output text as how successful we were on achieving our communicative goal. With that in mind, in the evaluation of GameRecapper, our focus was to evaluate the quality of the produced text and to compare how

users perceive a GameRecapper summary versus a human authored summary. The results showed that GameRecapper is able to produce a grammatically correct and easy to read summary, with an average intelligibility score of 92.91% and an average fluidity score of 85.73%. We came to the conclusion that the average fluidity score linearly decreases with the goal difference between the winning and the losing team, because of the larger amount of use of similar sentence templates. A larger amount of use of similar sentence templates makes the summary more propitious to repetitions of information which make the text seem less natural. These results emphasize the importance of a sentence aggregation and a referring expression generation task since these tasks can avoid repetitions of information or can make a text seem more natural.

According to the evaluation of GameRecapper vs human-authored summaries, we came to the conclusion that GameRecapper is able to make an accurate and complete summary of the game, when the game has no sent-offs. The total completeness average score was 80.42%. The results also suggest that even though the summary is not ready for being uploaded online (73.04%), it might be a good starting point draft for a journalist.

## 5.2 Future Work

The first and easier option to improve the output quality of GameRecapper is by adding additional major events. As we could notice in our evaluation, the lack of information besides goal events, may produce an incomplete summary. The addition of sent-offs, penaltis awarded, penaltis missed will make an immediate impact on the completeness of the generated new.

Another possibility for future work, could be trying to provide the input data with more statistics or even information of a live match commentary. This will provide the system with more options of content selection when the match has few or no major game events.

Fluidity in the summary can be increased by providing to the generation module a sentence aggregation or a referring generation expression task. Specially, in the cases where there are larger goal differences, or players with a significant amount of impact in the game statistics, the ability of aggregating two sentences with the same protagonists.

In spite of most of the module of GameRecapper being language-dependent, it would be interesting trying to apply it in a different language. The algorithm for creating the news is language-independent and apart from the addition of new sentence templates, it would not require major transformations in the system to port it to another language.



# References

- [1] E. Reiter and R. Dale. Building applied natural language generation systems. *Natural Language Engineering*, 3:57–87, 3 1997.
- [2] E. Reiter, R. Dale, and Z. Feng. *Building natural language generation systems*, volume 33. MIT Press, 2000.
- [3] M. E. Vicente, C. Barros, F. Agulló, F. S. Peregrino, and E. Lloret. La generacion de lenguaje natural: análisis del estado actual. *Computación y Sistemas*, 19(4), 2015.
- [4] E. Krahmer, S. Van Erk, and A. Verleg. Graph-based generation of referring expressions. *Computational Linguistics*, 29(1):53–72, 2003.
- [5] M. Theune, E. Klabbers, J. R. de Pijper, E. Krahmer, and J. Odijk. From data to speech: a general approach. *Natural Language Engineering*, 7:47–86, 3 2001.
- [6] F. M. Hasan. *Automatic generation of multilingual sports summaries*. PhD thesis, Applied Science: School of Computing Science, 2011.
- [7] N. Bouayad-Agha, G. Casamayor, S. Mille, and L. Wanner. Perspective-oriented generation of football match summaries: Old tasks, new challenges. *ACM Transactions on Speech and Language Processing (TSLP)*, 9(2):3, 2012.
- [8] 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.
- [9] A. van Dalen. The algorithms behind the headlines. *Journalism Practice*, 6(5-6):648–658, 2012.
- [10] R. Matsumoto, H. Nakayama, T. Harada, and Y. Kuniyoshi. Journalist robot: robot system making news articles from real world. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 1234–1241, Oct 2007.
- [11] K. S. Jones. Natural language processing: A historical review. volume 9 of *Linguistica Computazionale*, pages 3–16. 2001.
- [12] C. Clerwall. Enter the robot journalist. *Journalism Practice*, 8(5):519–531, 2014.
- [13] W. R. Swartout. A digitalis therapy advisor with explanations. Technical report, Cambridge, MA, USA, 1977.
- [14] R. Kittredge, A. Polguère, and E. Goldberg. Synthesizing weather forecasts from formatted data. In *Proceedings of the 11th Coference on Computational Linguistics, COLING '86*, pages 563–565, Stroudsburg, PA, USA, 1986. Association for Computational Linguistics.

- [15] A. Ramos-Soto, A. Bugarín, and S. Barro. On the role of linguistic descriptions of data in the building of natural language generation systems. *Fuzzy Sets and Systems*, 285:31 – 51, 2016. Special Issue on Linguistic Description of Time Series.
- [16] J. Coch. Evaluating and comparing three text-production techniques. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, pages 249–254, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics.
- [17] E. Reiter, S. Sripada, J. Hunter, J. Yu, and I. Davy. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167(1):137–169, 2005.
- [18] C. Sauper and R. Barzilay. Automatically generating wikipedia articles: A structure-aware approach. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 208–216. Association for Computational Linguistics, 2009.
- [19] S. Williams and E. Reiter. Generating basic skills reports for low-skilled readers. *Natural Language Engineering*, 14(04):495–525, 2008.
- [20] F. Portet, E. Reiter, A. Gatt, J. Hunter, S. Sripada, Y. Freer, and C. Sykes. Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence*, 173(7):789–816, 2009.
- [21] J. Yu, E. Reiter, J. Hunter, and C. Mellish. Choosing the content of textual summaries of large time-series data sets. *Natural Language Engineering*, 13(01):25–49, 2007.
- [22] K. Kukich. Design of a knowledge-based report generator. In *Proceedings of the 21st annual meeting on Association for Computational Linguistics*, pages 145–150. Association for Computational Linguistics, 1983.
- [23] J. Robin and K. McKeown. Empirically designing and evaluating a new revision-based model for summary generation. *Artificial Intelligence*, 85(1):135–179, 1996.
- [24] S. Mille and L. Wanner. Multilingual summarization in practice: the case of patent claims. In *Proceedings of the 12th European association of machine translation conference*, pages 120–129, 2008.
- [25] E. Reiter, R. Turner, N. Alm, R. Black, M. Dempster, and A. Waller. Using nlg to help language-impaired users tell stories and participate in social dialogues. In *Proceedings of the 12th European Workshop on Natural Language Generation*, pages 1–8. Association for Computational Linguistics, 2009.
- [26] L. Ferres, A. Parush, S. Roberts, and G. Lindgaard. Helping people with visual impairments gain access to graphical information through natural language: The igrph system. In *Computers Helping People with Special Needs*, pages 1122–1130. Springer, 2006.
- [27] E. Reiter, R. Robertson, and L. M. Osman. Lessons from a failure: Generating tailored smoking cessation letters. *Artificial Intelligence*, 144(1):41–58, 2003.
- [28] E. Cambria and B. White. Jumping nlp curves: a review of natural language processing research [review article]. *Computational Intelligence Magazine, IEEE*, 9(2):48–57, 2014.

- [29] M. Liu, R. A Calvo, and V. Rus. G-asks: An intelligent automatic question generation system for academic writing support. *Dialogue and Discourse: Special Issue on Question Generation*, 3(2):101–124, 2012.
- [30] M. O. Dzikovska, A. Isard, P. Bell, J. D. Moore, N. Steinhauser, and G. Campbell. Beetle ii: an adaptable tutorial dialogue system. In *Proceedings of the SIGDIAL 2011 Conference*, pages 338–340. Association for Computational Linguistics, 2011.
- [31] A. Fiedler. Natural language proof explanation. In *Mechanizing Mathematical Reasoning*, pages 342–363. Springer, 2005.
- [32] N. Rose Lim-Cheng, G.I Isidro G Fabia, Marco Emil G Quebral, and Miguelito T Yu. Shed: An online diet counselling system. 2014.
- [33] M. Johnston, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker, and P. Maloor. Match: An architecture for multimodal dialogue systems. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 376–383. Association for Computational Linguistics, 2002.
- [34] W. C. Mann and S. A. Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281, 1988.
- [35] M. Halliday, C. M. Matthiessen, and C. Matthiessen. *An introduction to functional grammar*. Routledge, 2014.
- [36] A. K. Joshi and Y. Schabes. Tree-adjointing grammars. In *Handbook of formal languages*, pages 69–123. Springer, 1997.
- [37] AK Zholkovskii and IA Mel’chuk. On a possible method and instrument for semantic synthesis. *Nauchno-tekhnicheskaya informatsiya*, (6), 1965.
- [38] B. J. Grosz and C. L. Sidner. Attention, intentions, and the structure of discourse. *Computational linguistics*, 12(3):175–204, 1986.
- [39] B. J. Grosz, S. Weinstein, and A. K. Joshi. Centering: A framework for modeling the local coherence of discourse. *Computational linguistics*, 21(2):203–225, 1995.
- [40] Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics, 1996.
- [41] P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.
- [42] R. Bod. Using an annotated corpus as a stochastic grammar. In *Proceedings of the sixth conference on European chapter of the Association for Computational Linguistics*, pages 37–44. Association for Computational Linguistics, 1993.
- [43] J. A. Bilmes and K. Kirchhoff. Factored language models and generalized parallel backoff. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers-Volume 2*, pages 4–6. Association for Computational Linguistics, 2003.

- [44] I. Langkilde and K. Knight. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 704–710. Association for Computational Linguistics, 1998.
- [45] H. Alshawi, S. Bangalore, and S. Douglas. Learning dependency translation models as collections of finite-state head transducers. *Computational Linguistics*, 26(1):45–60, 2000.
- [46] M. White, R. A. J. Clark, and J. D. Moore. Generating tailored, comparative descriptions with contextually appropriate intonation. *Computational Linguistics*, 36(2):159–201, 2010.
- [47] M. White. Openccg realizer manual. *Documentation of the OpenCCG Realizer*, 2012.
- [48] M. S. B. Galindo. ¿ qué es la generación de lenguaje natural? una visión general sobre el proceso de generación. *Inteligencia artificial: Revista Iberoamericana de Inteligencia Artificial*, 11(34):105–128, 2007.
- [49] D. D. McDonald and L. Bolc. *Natural language generation systems*. Springer Science & Business Media, 2012.
- [50] K. R. McKeown. Discourse strategies for generating natural-language text. *Artif. Intell.*, 27(1):1–41, September 1985.
- [51] H. Dalianis. Aggregation as a subtask of text and sentence planning. In *Proceedings of Florida AI Research Symposium, FLAIRS-96*, pages 1–5, 1996.
- [52] M. A. Walker, O. Rambow, and M. Rogati. Spot: A trainable sentence planner. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics, 2001.
- [53] H. Cheng and C. Mellish. Capturing the interaction between aggregation and text planning in two generation systems. In *Proceedings of the first international conference on Natural language generation-Volume 14*, pages 186–193. Association for Computational Linguistics, 2000.
- [54] M. Theune, F. Hielkema, and P. Hendriks. Performing aggregation and ellipsis using discourse structures. *Research on Language and Computation*, 4(4):353–375, 2006.
- [55] P. Edmonds and G. Hirst. Near-synonymy and lexical choice. *Computational linguistics*, 28(2):105–144, 2002.
- [56] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
- [57] I. M. R. De Bleeker. Towards an optimal lexicalization in a natural-sounding portable natural language generator for dialog systems. In *Proceedings of the ACL Student Research Workshop*, pages 61–66. Association for Computational Linguistics, 2005.
- [58] S. Mille, A. Burga, and L. Wanner. Ancoraupf: A multi-level annotation of spanish. In *Proceedings of DepLing*, 2013.
- [59] K. Garoufi and A. Koller. Automated planning for situated natural language generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1573–1582. Association for Computational Linguistics, 2010.

- [60] R. Dale and E. Reiter. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive science*, 19(2):233–263, 1995.
- [61] E. Krahmer and M. Theune. Efficient context-sensitive generation of referring expressions. *Information sharing: Reference and presupposition in language generation and interpretation*, 143:223–263, 2002.
- [62] S. Wann, M. Dras, R. Dale, and C. Paris. Improving grammaticality in statistical sentence generation: Introducing a dependency spanning tree algorithm with an argument satisfaction model. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 852–860. Association for Computational Linguistics, 2009.
- [63] P. Resnik and J. Lin. 11 evaluation of nlp systems. *The handbook of computational linguistics and natural language processing*, 57:271, 2010.
- [64] J. C. Pereira, A. Teixeira, and J. Sousa Pinto. Towards a hybrid nlg system for data2text in portuguese. In *Information Systems and Technologies (CISTI), 2015 10th Iberian Conference on*, pages 1–6. IEEE, 2015.
- [65] E. Reiter. Natural language generation. *The Handbook of Computational Linguistics and Natural Language Processing*, pages 574–598, 2010.
- [66] A. S. Law, Y. Freer, J. Hunter, R. H. Logie, N. McIntosh, and J. Quinn. A comparison of graphical and textual presentations of time series data to support medical decision making in the neonatal intensive care unit. *Journal of Clinical Monitoring and Computing*, 19(3):183–194.
- [67] A. S. Lennox, L. M. Osman, E. Reiter, R. Robertson, J. Friend, I. McCann, D. Skatun, P. T. Donnan, et al. Cost effectiveness of computer tailored and non-tailored smoking cessation letters in general practice: randomised controlled trial. *British Medical Journal - BMJ*, 322(7299):1396, 2001.
- [68] A. Belz. That’s nice... what can you do with it? *Computational Linguistics*, 35(1):111–118, 2009.
- [69] K. Papineni, S. Roukos, T. Ward, and W.J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.
- [70] M. Denkowski and A. Lavie. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 85–91. Association for Computational Linguistics, 2011.
- [71] H. Van der Kaa and E. Krahmer. Journalist versus news consumer: The perceived credibility of machine written news. In *Proceedings of the Computation+Journalism conference*, 2014.
- [72] A. Belz and E. Reiter. Comparing automatic and human evaluation of nlg systems. In *European Chapter of the Association for Computational Linguistics*, 2006.
- [73] NLG Systems wiki. <http://www.nlg-wiki.org/>. Accessed: 2016-02-01.
- [74] S. Bird, E. Klein, and E Loper. *Natural Language Processing with Python*. O’Reilly Media, 2009.

- [75] S. Banerjee and A. Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, volume 29, pages 65–72, 2005.
- [76] A. Gatt and E. Reiter. Simplenlg: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation*, pages 90–93. Association for Computational Linguistics, 2009.
- [77] D. Galanis and I. Androutsopoulos. Generating multilingual descriptions from linguistically annotated owl ontologies: the naturalowl system. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*, pages 143–146. Association for Computational Linguistics, 2007.
- [78] M. Steedman and J. Baldridge. Combinatory categorial grammar. *Non-Transformational Syntax: Formal and Explicit Models of Grammar*. Wiley-Blackwell, 2011.
- [79] A. Graefe. Guide to automated journalism. Tow Center for Digital Journalism - A Tow/Knight Guide, 2016.
- [80] P. Corford. A leap forward in quarterly earnings stories. [https://blog.ap.org/announcements/a-leap-forward-in-quarterly-earnings-stories?utm\\_source=insights&utm\\_medium=blog&utm\\_campaign=automation-research](https://blog.ap.org/announcements/a-leap-forward-in-quarterly-earnings-stories?utm_source=insights&utm_medium=blog&utm_campaign=automation-research), June 2014. Accessed: 2015-12-03.
- [81] K. Tanaka-Ishii, K. Hasida, and I. Noda. Reactive content selection in the generation of real-time soccer commentary. In *Proceedings of the 17th international conference on Computational linguistics-Volume 2*, pages 1282–1288. Association for Computational Linguistics, 1998.