Describing golf shots using Natural Language Generation

An investigation of a suitable method for implementing a Natural Language Generation system with the aim of being a useful resource for golfers

LUDVIG JANSSON
ROBIN ENGSTRÖM

Degree project
Supervisor: Michael Minock
Examiner: Karl Meinke
Abstract

Natural Language Generation, NLG, is the translation of raw data into legible and understandable text. It is currently being used in a variety of ways ranging from weather forecasts to directions given by Google Maps. External NLG libraries are available for several programming languages, allowing more focus on the determining what would be included in the text generated.

This report aims to investigate a suitable method of building an NLG system to be used with data provided by the Protracer software with the intent to provide golfers with text-based feedback of their game. Possible methods have been narrowed down to three approaches. The first is simply outputting random feedback to the golfer with generic words and hope for the best. The second is using a machine learning technique that takes shot parameters along with a human interpretation of that shot. From this, the algorithm would acquire information about what builds up a specific shot. Finally, the last method is using shot parameters along with human interpretation to construct an algorithm with observed threshold values to determine the shot type.

Ultimately, the system constructed with the help of the report was effective in classifying and describing golf ball trajectories. The difference between human interpretation and that of the system was negligible as the line between the classifications of a golf shot is very thin.
Contents

1 Introduction ........................................... 1
   1.1 Problem statement ................................. 1

2 Scope .................................................... 3

3 Background ............................................. 5
   3.1 Golf Linguistics .................................... 5
   3.2 Protracer ........................................... 6
   3.3 Natural language generation ......................... 6
      3.3.1 Content determination and text planning .......... 6
      3.3.2 Sentence Planning .................................. 6
      3.3.3 Realization ......................................... 7

4 Method .................................................. 9

5 Research ............................................... 11
   5.1 State-of-the-art ..................................... 11
      5.1.1 SumTime-Mousam ................................. 11
   5.2 Possible ways of implementing the Content Determination stage of the NLG System (Research) ................................. 14
      5.3 Machine learning method based on human interpretation ................................. 15
         5.3.1 Pros and cons of this method ..................... 16
      5.4 Random classification and feedback .................. 16
         5.4.1 Pros and cons of this method ..................... 17
      5.5 Determine classifications based on human interpretation ................................. 17
         5.5.1 Pros and cons of this method ..................... 17
      5.6 Protracer software .................................... 18
      5.7 Choosing a framework ............................... 18
         5.7.1 SimpleNLG ........................................ 18

6 Building the System .................................. 21
   6.1 Knowledge Acquisition and Validation ................. 21
   6.2 Content determination and text planning ............... 23
   6.3 Sentence planning .................................... 24
Chapter 1

Introduction

The aim of this report is to research a suitable method for implementing a natural language generator so as to make comments on golf ball trajectories in the best possible way, as no such system is available on the market as of present. The purpose of the application is to give golf players information about their golf ball trajectory after recording the shot with hardware and software provided by Protracer. Due to the amounts of data contained in a golf ball trajectory the natural language generator will have to be able to generate complex sentences.

The objective is to develop an application that can achieve this using information about the golf ball trajectory generated by the Protracer Range system. The application is meant to work as an extension to the software, and a future potential use would be providing real-time feedback to the golf players practicing on the driving range. This could be useful to golfers by helping them improve their game.

1.1 Problem statement

The objective described above leads to the problem statement in this report:

How can a Natural Language Generation system be constructed with the intent to analyze data from the Protracer software, draw conclusions about the properties of a golf shot and return a Natural Language statement describing the ball trajectory?
Chapter 2

Scope

In order to preserve a well-defined scope of the project, some limitations and require-ments have been realized and established.

Ideally, the system should be able to give real-time feedback to the golfer. This was, however, deemed out of the scope of this project due to time limitations, but could be a future extension to the system.

For similar reasons, but also due to the fact that this is hard to achieve with the average golfer, the system will not show at which target a golfer was aiming. Instead, the system can show the nearest target and how close the shot was to this.

As the system does not know where the user was aiming, the system would not be able to tell if the shot was pushed (initial ball direction is to the right of where the golfer was aiming if the golfer is right handed) or pulled (same, but to the left for a right handed golfer).

Additionally, the system will calculate shot length based on distance travelled between the moment of impact and when the ball lands, in other words the carry, and the system will assume the player is right handed in the judgment of shot shape.

The outlined requirements for the system are:

- Output text needs to be coherent and grammatically correct English.
- Output needs to be as close to human interpretation of the golf shot as possible.
- System classification of golf shots needs to be based on human interpretation of a golf shot.
- The system needs to be fast and effective.
- The system must be something Protracer would find a use in.
Chapter 3

Background

This chapter aims to introduce the reader to both Protracer as well as what an NLG system is and for what it is used.

3.1 Golf Linguistics

Golf is a game with a variety of specific words to describe a specific shot. This section is meant to give the reader an introduction into the golf vocabulary this system may use.

**Carry**  The distance the ball has travelled before hitting the ground for the first time.

**Top/Bladed**  A shot with a topspin. Related to the trajectory shape of the golf ball as it is usually very low and rolls far.

**Hook**  A shot with a severe spin in the direction of the swing. For a right-handed golfer, this type of trajectory bends far to the left.

**Draw**  A shot with a minor to substantial spin in the direction of the swing. For a right-handed golfer, this means the bend is less to the left than a hook.

**Fade**  The direct opposite of a draw; for a right-handed golfer, this shot bends to the right.

**Slice**  The direct opposite of a hook; for a right-handed golfer, this shot bends severely to the right.

**Backspin**  How much backspin the ball has. This is directly related to the trajectory shape of the golf ball and how quickly it stops after landing.
3.2 Protracer

Protracer is a small company based in Sweden. Since its inception, the mission of the company has been to enhance the game of golf. The first aspect of this included that watching golf on TV was missing a fundamental piece; seeing the golf ball's trajectory[1]. Through analyzing the images produced by TV cameras at PGA competitions, a product was developed that would draw trajectories in real-time and add these to the TV images sent to viewers. The product has since grown exponentially, and as of 2014, it will be a standard at PGA competitions aired by NBC and Golf Channel [2].

As a result of the success with the TV product, Protracer has now switched most of their focus to adapting the system for driving ranges and for the average golfer. This new range system is already live on a few driving ranges across Europe and will soon be in North America as well. The system collects data from each shot and analyzes this before presenting it back to the golfer. This way, the software has the potential of helping golfers improve their game [1, 2].

3.3 Natural language generation

The purpose of Natural Language Generation (NLG) system is to translate raw data into legible text in a given language. This is achieved by a combination of artificial intelligence (AI), natural language processing techniques [8], and knowledge acquisition (KA) to build the AI [9]. Current applications of NLG systems include weather forecasting, as well as helping to write routine documents and portray information in an understandable way to someone not well included in the subject [3].

According to Ehud Reiter [3], nearly every NLG systems follow the same structure and perform three different tasks:

3.3.1 Content determination and text planning

These two are usually done simultaneously in an NLG system. Content Determination handles what kind of information will be portrayed, while Text Planning structures this information in a rhetorically coherent manner. This can be done in various different levels of sophistication, ranging from a hard-coded system of words and phrases to a powerful system utilizing AI to achieve a result. A popular approach is a middle-ground between these two extremes [3].

3.3.2 Sentence Planning

Sentence Planning, is different from Text Planning as the purpose of this is to add a flow in sentence structure, imitating human writing. This adds for example conjunctions and pronominalization to sentences to avoid awkwardness in the
3.3. NATURAL LANGUAGE GENERATION

structure. If sentence flow is not particularly important, Sentence Planning can be
demphasized and weight can be put towards the other two tasks instead [3].

3.3.3 Realization

Realization makes sure the output text follows the grammatical rules that belong
to the language being used. This includes for example proper punctuation, when
to use singular/plural, etc. There are several linguistic theorems that can be used
for a Realization module, but acceptable performance can still be achieved without
much effort. If only a specific number of sentence types are being generated, sentence
templates can be used to achieve good performance with minimum effort [3].
Chapter 4

Method

The project consisted of four major steps:

1. Analysing the information given by the Protracer software in order to determine what properties about the golf ball trajectory were retrievable. This was done in order to determine what type of information the application could give the user using NLG.

2. Analysing plausible approaches to the problem in order to determine which method fit the needs of the application best with given possible input of information. This was done through both a study of a current approach to gain inspiration and then discussing how to implement a Knowledge Acquisition method.

3. Building the system using knowledge gained from literature studies as well as results gathered from steps 1-2.

4. Analyzing results from steps 1-3.

With the method above, it was determined enough effort would be put on the two most demanding steps and through this a positive result would be achieved. These two steps were analyzing the information provided by Protracer and analyzing plausible approaches.

The input to the system was collected from a database with over 5000 golf shots from different positions at a driving range. All golf shots were thus real shots. This data was collected towards the end of the day on a Sunday to ensure a maximum number of test data, as the database is emptied over night.

Evaluation of the results was conducted by comparing output from the GolfBot application with golf shot classifications from golfers looking at the shots in the 3D grid application shown in figure 6.1. The figures represented as percent values were acquired by looking at the percent of the human specified properties that were consistent with the GolfBot generated properties. For instance, the human interpretation of a shot was "Hooked. It looks like he tried to hit it hard and got
CHAPTER 4. METHOD

some back spin out of it.", which had the properties **Hook**, **Hard** and **Back spin**. The GolfBot output of the same shot was "The shot was a substantial draw with substantial back spin. It was 158 m long.", which had the properties **Draw** and **Back spin**. This means that the GolfBot was 33\% accurate as it noticed the back spin, but considered it being a **Draw** instead of a **Hook**. It also missed that the shot was hard. Furthermore, the system and output was also checked against the requirements outlined above.

The literature used was found primarily through a literature search. The book *Building Natural Language Generation Systems*, written by Reiter & Dale, was introduced by the supervisor of this project, and the website of the University of Aberdeen was searched for more sources. In order to find sources outside the University of Aberdeen, a literature search online was conducted.
Chapter 5

Research

5.1 State-of-the-art

This chapter of the report aims to investigate a state-of-the-art of NLG systems and see how it translates raw computer data to text in order to be used in everyday life. The purpose of this is to draw inspiration to the construction of GolfBot.

5.1.1 SumTime-Mousam

SumTime-Mousam is a system developed by Dr. Ehud Reiter and Dr. S.G. Somaya-julu at the University of Aberdeen, Scotland, and Ian Davy from Weathernews Ltd., Aberdeen. The system is part of the larger University of Aberdeen project Sum-Time. This name is derived from “Generating Summaries of Time Series Data”, which is the broad purpose of the project. More specifically, SumTime-Mousam is an NLG system generating a weather forecasts based on Numerical Weather Prediction (NWP) data for offshore oil rig applications [6].

General overview

The input to SumTime-Mousam is a table of weather data from an NWP model where each row marks three hours in a day for a maximum of 72 hours and each column contains different forecast parameters for those three hours. There are about 40 parameters in a full data set and these include wind direction and speed for near surface (10 meters) and height of a turbine (50 m) [6].

SumTime-Mousam interprets this data and outputs text organized into forecast elements such as wind, air temp, and wave, each of which describing basic weather parameters. For example, if SumTime-Mousam received the data in figure 5.1 as input, a forecast for the wind element would have been returned as output. An important aspect of the system is that its forecasts can be configured in level of detail and style by the end user. This is done by saving user preferences in control data tables in external files [6].
SUM TIME-MOUSAM follows the structure of an NLG system outlined in the background in a very structured manner:

Document Planning

First, the data needs to be reduced and the important datapoints need to be selected. In order to find what datapoints are “important”, an extensive analysis is performed on the data set. This analysis is known as segmentation, and is the process of aligning linear segments to an input data series while keeping the maximum

<table>
<thead>
<tr>
<th>Date and Time</th>
<th>Wind Dir</th>
<th>Wind Speed 10m</th>
<th>Wind Speed 50m</th>
<th>Gust 10m</th>
<th>Gust 50m</th>
</tr>
</thead>
<tbody>
<tr>
<td>12-06-2002 02:00:00</td>
<td>SW</td>
<td>9.0</td>
<td>11.0</td>
<td>11.0</td>
<td>14.0</td>
</tr>
<tr>
<td>12-06-2002 06:00:00</td>
<td>W</td>
<td>10.0</td>
<td>12.0</td>
<td>12.0</td>
<td>16.0</td>
</tr>
<tr>
<td>12-06-2002 09:00:00</td>
<td>W</td>
<td>11.0</td>
<td>14.0</td>
<td>14.0</td>
<td>17.0</td>
</tr>
<tr>
<td>12-06-2002 12:00:00</td>
<td>WSW</td>
<td>10.0</td>
<td>12.0</td>
<td>12.0</td>
<td>16.0</td>
</tr>
<tr>
<td>12-06-2002 15:00:00</td>
<td>SW</td>
<td>7.0</td>
<td>9.0</td>
<td>9.0</td>
<td>11.0</td>
</tr>
<tr>
<td>12-06-2002 18:00:00</td>
<td>SSW</td>
<td>8.0</td>
<td>10.0</td>
<td>10.0</td>
<td>12.0</td>
</tr>
<tr>
<td>12-06-2002 21:00:00</td>
<td>S</td>
<td>9.0</td>
<td>11.0</td>
<td>11.0</td>
<td>14.0</td>
</tr>
<tr>
<td>13-06-2002 00:00:00</td>
<td>S</td>
<td>12.0</td>
<td>15.0</td>
<td>15.0</td>
<td>19.0</td>
</tr>
</tbody>
</table>

**Figure 5.1.** Portion of the input to SumTime-Mousam, showing the weather of 12th June 2002.

**Figure 5.2.** Text generated by SumTime-Mousam. The wind element has been generated by the data in figure 5.1.
error introduced lower than a threshold value predefined by the user. SumTimeMousam does its segmentation through a modified version of the bottom-up algorithm, discussed by Eamonn Keogh at the University of California-Riverside. Most of the modifications to the algorithm were done in order to maintain the user configurability as discussed above [6].

Once the segmentation is complete, the “Wind Speed 10m” column in figure 5.1 produces one segment, binding 10 knots at 6 a.m. to 12 knots at 12 a.m. This means it interpreted this change in wind speed as the important part of this column. The “Wind Direction” column produces two segments: one binding W at 6 a.m. to SW at 3 p.m., and one binding SW at 3 p.m. to S at 12 a.m., realizing two major changes in wind direction and determining these as important [6].

These segments are in turn summarized into tuples containing time, lower wind speed bound, upper wind speed bound, wind direction, and modifiers. The wind speed bounds predictions are computed using the user preferences and the modifiers include gusts, showers, and change in weather. In order to maintain the level of information in the segments, the tuples are built up by the union of the segments. This means the sample data produces three tuples, which is also the output of the document planning stage:

\[(0600, 8, 13, W, \text{nil}), (1500, 8, 13, \text{SW}, \text{nil}), (2400, 10, 15, S, \text{nil})\] [6].

**Sentence Planning**

Sentence planning performs two tasks: building up a lexical structure, and asserting what will be included in the final output through a series of flags. For the purpose of making lexical choices, the tuples are arranged into high-level descriptive phrases. These phrases are in turn built up by lower-level phrase components, shown in the middle column of the table below [6].

<table>
<thead>
<tr>
<th>Tuple</th>
<th>Wind Phrase Components</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0600, 8, 13, W, nil)</td>
<td>Dir phrase 1: W; Speed Phrase 1: 8-13; Time Phrase 1: by early morning</td>
<td>Time phrase is elided</td>
</tr>
<tr>
<td>(1500, 8, 13, SW, nil)</td>
<td>Dir phrase 2: SW; Speed phrase 2: 8-13; Time Phrase 2: by mid afternoon; Verb Phrase: backing</td>
<td>Speed phrase is elided</td>
</tr>
<tr>
<td>(2400, 10, 15, S, nil)</td>
<td>Dir phrase 3: S; Speed Phrase 3: 10-15; Time Phrase 3: by midnight; Verb Phrase: backing</td>
<td>Verb phrase is elided</td>
</tr>
</tbody>
</table>

**Figure 5.3.** Table showing the process of the sentence planning stage performed on the tuples generated in the document planning stage.

The flags follow a series of rules in order to build a coherent sentence structure.
These rules include:

- Ignoring the time phrase of the first tuple as this is the beginning of the forecast
- Ignoring the a phrase component if it is the same as the previous one
- Ignoring an entire wind phrase if both the speed and the direction components are ignored

These flags are illustrated in the third column in figure 5.3 [6].

Realization

Input to this stage is the lexicalized phrases created in the sentence planning stage and the flags that correspond to them. This stage is responsible of organizing the complete sentence in a coherent manner following grammatical rules and guidelines. Thus, the output of the realization stage, and the system as a whole, for the example data from the Wind Speed 10 m column in figure 5.1 is:

\[ W \text{ 8-13 backing SW by mid afternoon and S 10-15 by midnight} \]

How it is used

After SunTime-Mousam had been created, the company Weathernews based in Aberdeen used the software to create an initial weather forecast for a year as an evaluation period. They used it as part of an iterative process where the forecaster first edited the NWP data in their internal editing tool, creating data 1 in figure 5.4, ran it through SunTime-Mousam to create forecast text 1 in figure 5.4. The forecaster then edited data 1 where needed to create data 2 and ran it through SunTime-Mousam again to create forecast text 2. Then, the forecaster edits this output manually and creates forecast text 3, which is delivered to the end-user [6].

A later evaluation conducted in 2005 showed that forecast users preferred the texts generated by SunTime-Mousam over text generated by humans. This might, according to Reiter et al., have been the first time text generated by NLG was preferred and credit is given to the better word choice in forecasts generated by SunTime-Mousam [7].

5.2 Possible ways of implementing the Content Determination stage of the NLG System (Research)

It was established early on that the hardest part of implementing the NLG system would be implementing the Content Determination stage described by Dr. Reiter
5.3. Machine learning method based on human interpretation

The basic idea behind this method is simple. A number of golfers with a wide range of skill level would take a survey in which a high number of shots have been individually plotted in a graph with a table showing the properties of the shot.
They would for example say “This shot is a draw with a 80 meter carry. The height is very low, so this could indicate a topped shot.”, or similar.

These interpretations would then together with their shot parameters be divided into two sets: a training set containing most of the shots and a testing set. Using the training set, an algorithm using a machine learning method would classify what makes up a slice or a hook, and what makes a shot topped or not.

Thus, after the machine learning algorithm was done with realizing what classifies specific shots, the input to the algorithm would be individual shots and the output would be classifications. As not all of the shots investigated earlier by golfers would have been used in the machine learning algorithm, the set of shots that had not been used (the testing set) would be the testing data which would be compared to the output from the GolfBot system in order to maintain a closed set of data. From these results, conclusions would then be drawn. To achieve this, the supervised learning approach was deemed fitting as it builds up an inferred function to classify data from a data set with already realized classifications [5].

### 5.3.1 Pros and cons of this method

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Efficient way of building up data</td>
<td>Time consuming</td>
</tr>
<tr>
<td>Closed set of data for training and testing</td>
<td>Advanced</td>
</tr>
<tr>
<td>Could be very accurate if training set is big enough</td>
<td>Could not work well if training set is not big enough</td>
</tr>
<tr>
<td>Scientific approach with survey</td>
<td></td>
</tr>
</tbody>
</table>

Even though this method initially seemed very fitting for the project, it was realized that knowledge in the machine learning field was lacking. As this project is about NLG and not machine learning, this approach was deemed out of the project’s scope.

### 5.4 Random classification and feedback

Another approach is simply outputting random classifications from the content determination and use these in the NLG to provide the golfers with legible sentences. As a result, this approach would be substantially easier than the machine learning method above.

The full NLG system would have the same input and output as the machine learning approach above and golfers would be consulted in the same way to build up a testing set. No training set would be needed as there are no classifications to be made since they are all random, but comparison to the testing set would be the same.
5.5. DETERMINE CLASSIFICATIONS BASED ON HUMAN INTERPRETATION

5.4.1 Pros and cons of this method

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to implement</td>
<td>Could be highly inaccurate</td>
</tr>
<tr>
<td>Could still be compared to a testing set, preserving a somewhat scientific method</td>
<td>Not very scientific</td>
</tr>
<tr>
<td></td>
<td>An all-round bad approach</td>
</tr>
</tbody>
</table>

This approach was deleted from the candidates of methods early on as it is neither very scientific or accurate. Even though there are only four or five classifications to be made, this could still lead to a substantial fail rate when compared to the test set.

5.5 Determine classifications based on human interpretation

After the two approaches above had been discussed and discarded, it was decided to use some of the factors in the machine learning approach. The idea behind this approach is to keep the scientific method by keeping a database over shots that have been interpreted by golfers and described in their own words. This data would then be divided into two subsets; the training set and the testing set, just like above. From the testing set shot data, conclusions would be drawn concerning what in the shot data could be used to determine classifications for the shot.

Based on these observations, the content determination stage of the NLG system would then be built. Finally, the NLG system would be tested against the human interpretation of the testing set shots in order to establish results, thus maintaining a closed set of learning and testing data. A similarity to the machine learning approach is also kept as it utilizes the same scientific method while trying to find the middle ground described by Dr. Reiter [3].

5.5.1 Pros and cons of this method

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easier to implement than machine learning</td>
<td>Might not be as accurate as the machine learning approach</td>
</tr>
<tr>
<td>Scientific approach</td>
<td>Might need a big training set to be accurate</td>
</tr>
<tr>
<td>Closed set of data for training and testing</td>
<td>Could be accurate</td>
</tr>
</tbody>
</table>
This was determined to be the best approach of the three as it maintains a scientific approach, might be accurate, is not out of the scope of the project, and is not as time consuming.

5.6 Protracer software

The Protracer system is able to provide the coordinates of the golf ball when traveling through the air. This information is collected through a software analyzing images from two cameras positioned at a driving range. The software utilizes these images to triangulate the positions of multiple golf balls and then sends this information to a physics engine that determines if the observed positions belong to a golf shot, and if so which golf shot and from where. From this it is then possible to plot the golf ball trajectory in an x, y, z grid. It is also possible to calculate certain properties of the trajectory by performing calculations on the coordinates. Some of the possible properties are:

- Carry
- Height
- Ball speed
- Launch angle
- Horizontal spin
- A rough estimate on vertical spin

5.7 Choosing a framework

As a lot of help in building NLG systems can be found using a framework, this section aims to describe the framework used by GolfBot.

5.7.1 SimpleNLG

SimpleNLG is a realisation engine for English to be used in NLG systems written in Java by Albert Gatt and Ehud Reiter at the Department of Computing Science at the University of Aberdeen, Scotland. One of its main purposes is to allow more research to be allocated to the Document and Sentence Planning stages instead of worrying about actually outputting the data as legible text.

Born out of experience in building large scale NLG systems, SimpleNLG has evolved into a robust and easy-to-use API covering a significant portion of English morphology and syntax while still letting users maintain a high control over the realisation. The engine lets the user determine the content of the sentences through a series of classes. These classes define sentence components and the purpose of
5.7. CHOOSING A FRAMEWORK

these is to let the user control every step in the generation process, from incoherent words to full sentence structure. Building a syntactic structure in SimpleNLG is a matter of setting properties to these classes [4].

The building of the syntactic structure and outputting this as text is done in four steps [4]:

1. Initializing the basic parts of the text.

2. Setting properties to these parts, as provided by the API and illustrated in the bottom half of figure 5.

3. Combining the parts into larger structures, again using the API.

4. Passing this larger structure on to the realiser itself, which goes through the parts of the structure, applying the rules and properties defined to these, and returns the successfully realized string.

The API does not care if the properties are partly made up of canned strings, as defined by Reiter & Dale (2000), or not. A canned string is a text whose character sequence has already been defined, but still needs some processing by the realizer. An example of this is the dog barked. Note the lack of capitalization and punctuation, the reason behind being the text is not realized and can thus be used as-is in SimpleNLG to build up a potentially longer sentence [8]. A mix of canned strings is useful in a context where a deterministic fashion is available, while certain other applications may require a more flexible mapping. Examples when a mix is not preferable is when output is more dependent on semantic features and context [4].

```
Phrase s1 =
    new SPhraseSpec('leave');
s1.setTense(PAST);
s1.setObject(
    new NPhraseSpec('the', 'house'));
Phrase s2 =
    new StringPhraseSpec('the boys');
s1.setSubject(s2);
s1.setInterrogative:YES_NO;
(Did the boys leave home?)
s1.setInterrogative:WHERE, OBJECT;
(Where did the boys leave?)
```

Figure 5.5. A simple example of building a sentence using SimpleNLG.
Illustrated in Figure 5.5 is a simple example in which SimpleNLG is used to construct the sentence “The boys left the house.” The example starts with initializing the main verb “leave” and setting its tense to past. Then, an \texttt{NPPhraseSpec} is initialized and attached to the verb. This \texttt{PhraseSpec} subclass is thus being used to show \texttt{what} was left. Then, a \texttt{subject} is attached to the verb showing \texttt{who} left the house. To further illustrate how this sentence can be modified, interrogative types are shown that can be used to turn the sentence into a question. It is through this example easy to see that using SimpleNLG is fundamentally adding features to words, and through this creating full sentences and texts [4].

SimpleNLG has been used in several different projects at the University of Aberdeen, for example the BabyTalk project [10]. According to Gatt & Reiter (2009), the engine has been widely used for three main tasks:

- Front-end in NLG systems where realization is not the main investigative purpose.
- NLG component in the user interface of larger systems.
- Teaching tool in advanced university studies on Natural Language Processing.

This, according to the authors, reflects the simplicity and wide array of applications of SimpleNLG [4].
Chapter 6

Building the System

6.1 Knowledge Acquisition and Validation

Before the building an NLG system could be started, what kind of properties that make a golf shot be classified as a slice, draw, fade, etc, had to be determined. According to Reiter, et. al (2003), this is known as Knowledge Acquisition (KA) and there are broadly speaking two main types of techniques for this:

1. Working with experts through interviews and other structured fashions.
2. Working with data sets of correct solutions.

Of the two, the latter is the most common one in Natural Language Processing and is the one that was chosen for this project [9].

The KA was structured so that golfers were shown a golf shot drawn in a grid with some basic values showing length, height, ball speed, etc, on a computer screen. To draw the shot, a Python script was created that pulled the information from the database. They were then asked to describe this shot in their own words and focusing on length, trajectory shape (slice, draw, etc), and backspin versus topspin. They were also asked what kind of shot data was most important for them to always get feedback on. The data was then divided into the training set and the learning set, as mentioned above. From the training set data, the following conclusions could be drawn after setting up a 95% confidence interval:

- Slice is considered the mirror image of hook and vice versa.
- Draw is considered the mirror image of fade and vice versa.
- It is very hard to define a shot as straight as there usually is some kind of shape on the trajectory.
- Trajectories with late apexes were considered to have a high amount of backspin.
• Trajectories that had a symmetrical shape if straightened out were considered to have topspin rather than backspin.

• A draw is 0.6° to 7.6° to the left of its initial direction, and a fade is in the same range but to the right.

• A hook is everything that is more than 7.6° to the left of its initial direction, and a slice is thus everything more than 7.6° to the right.

• A topped shot is a shot with a low maximum height, symmetric shape, and short carry. The launch angle did not really make a difference in this matter.

• Golfers always want to know how long the shot was.

![Graph of a shot shown to golfers.](image)

**Figure 6.1.** The graph of a shot shown to golfers.

Reiter et al (2003) also describe ways of validating knowledge. Again, these are divided into two main types:

1. Having the knowledge and system checked by experts.

2. Having a data set of known inputs and outputs for comparison.
6.2. CONTENT DETERMINATION AND TEXT PLANNING

Knowledge can thus be acquired through establishing a data set, and it can also be validated through using a data set. Furthermore, knowledge can also be validated through experiments testing its intended function on users and see if the result is desirable. It is, however, important to distinguish the knowledge acquisition from the validation in order to maintain integrity. Thus, the same data set used for the KA cannot be used to validate the system afterwards. This is why the data was divided into two sets earlier in the process [9].

6.2 Content determination and text planning

The content determination is the largest stage of this NLG system. The purpose is to analyze the input data and find what in it is valuable for the user to receive back. For example, a long shot (decided to be >190 meters for amateurs) that has a high ball speed can be interpreted as “a long and hard-hit drive”, and a very low shot with a short carry is instead portrayed as “a topped shot”.

```java
/**
 * This determines if a shot has been hit hard.
 * @return String array with the speed value and a parameter saying if this shot was hard or not.
 */
private String[] determineHardHit(){
    int tempSpeed = (int) Math.ceil((getShotSpeed() + 2)/2);
    float tempLength = getShotLength();
    String[] res = new String[2];
    res[0] = String.valueOf(tempSpeed);
    if(tempLength > 190){
        if(tempSpeed > 108.21){
            res[1] = "yes";
        } else {
            res[1] = "no";
        }
    } else {
        res[1] = "no";
    }
    return res;
}
```

Figure 6.2. Snippet determining if the shot was a hard hit or not.

As illustrated by figure 6.3, input to the content determination is shot data about a specific golf shot straight from the Protracer database. In order to find which data is valuable, the content determination goes through a series of functions to calculate values based on the input data. These values portray length, ball speed, launch angle, spin axis, etc, and together build up the parameters for the shot. Length is always portrayed back to the user as this is what golfers found most important while the rest of the values help determining the shot shape and
thus classifying shots into categories. These classifications are deemed valuable as they provide the golfer with a lot of feedback on the shot and are thus portrayed back. Shown in figure 6.2 is also how the importance is passed on by setting a flag to “yes” or “no” in the String array it returns, much like the SumTime-Mousam system. Once all of this information has been calculated, determined, and retrieved, the content determination is finished and terminates.

6.3 Sentence planning

The sentence planner of GolfBot is simple. It is using the data from the content determination stage and making this into snippets of coherent text. For example, if the content determination decide a shot was a “draw”, the sentence planning stage turns this into an object containing the information “the spin suggests that you made a draw”.

This stage builds up sentences from smaller NLG elements (which may also in turn be even smaller NLG elements) called comments. If the comment is itself built up by smaller NLG elements, these are put together using the logic introduced by the CoordinatedPhraseElement logic in SimpleNLG which gives the user the possibility to merge two NLG elements into one.

Input to the function is a String array where the first position contains the speed value and the second position the importance flag determined by the content determination stage. If the flag is set to “yes”, then the value added to a sentence element putting the speed value in a context. Output from this specific function is a DocumentElement object which when realized by the realizer will turn this into “The height is x meters” where x is the height value. The sentence planner as a whole outputs a series of SimpleNLG fragments which will be used as input to the realizer.

6.4 Realizer

The realizer stage of this system is the simplest one, thanks to the extensive SimpleNLG library. This is the stage where the fragments created by the sentence planner are put together in a coherent and grammatically correct manner. Words like “and” are added to join the fragments together and create a flow in the text. Then, when everything is ready for output, the complete structure is run through SimpleNLG’s built-in realizer which converts everything to text.
6.4. REALIZER

![Diagram of the GolfBot NLG system using SimpleNLG]

**Figure 6.3.** GolfBot NLG system using SimpleNLG.

```java
/**
 * Generates a comment regarding the speed of the shot.
 * @param speedDesc A sentence describing the ball speed
 * @return A comment about the speed of the shot.
 */
public CoordinatedPhraseElement speedComment(String[] speedDesc){
    if(speedDesc[0].equals("no")){
        return null;
    } else {
        $PhraseSpec s1 = nlgFactory.createClause();
        s1.setObject(nlgFactory.createNomPhrase("a", "hard hit"));
        $PhraseSpec s2 = nlgFactory.createClause();
        s2.setSubject("a speed");
        s2.setPreModifier("of");
        s2.setObject(nlgFactory.createNomPhrase(speedDesc[0] + " mph"));
        CoordinatedPhraseElement coordPhrase = nlgFactory.createCoordinatedPhrase(s1, s2);
        coordPhrase.setConjunction("with");
        return coordPhrase;
    }
}
```

**Figure 6.4.** Short snippet demonstrating building a comment in the system.
Figure 6.5. The realizer of this NLG system. The top method binds the sentences together and the bottom one realizes the text.
Chapter 7

Results

After inputting the results from the Knowledge Acquisition process in the system, the system was calibrated to test it against the testing set established in the KA process. The testing set was 17 shots out of the 50 shots golfers were asked to describe. It was found that most shots in the testing set were classified with at least 50% accuracy compared to the corresponding human interpretation of the shot. Thus, at least 50% of the classifications that could be done with the shot data were an exact match between the system’s interpretation and of the human interpretation. On an average, the system was 66% accurate in this matter with some interpretations being exact matches. Shown in table 7.1 is a portion of the results acquired from GolfBot next to the corresponding human interpretation.

As time was not permitting, testing based on whether golfers thought the system was accurate enough and/or useful could not be conducted. Despite this, the CEO of Protracer was interested in using the system and saw potential in using it [2].
<table>
<thead>
<tr>
<th>Human interpretation</th>
<th>GolfBot interpretation</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classic slice. He hit it hard and got a lot of back spin.</td>
<td>The shot was a strong fade with substantial back spin. It was a hard hit with a speed of 140 mph and 195 m long.</td>
<td>GolfBot agreed on it being a hard hit with a lot of backspin. It did not agree on the shot shape.</td>
</tr>
<tr>
<td>Very beautiful draw. A lot of back spin.</td>
<td>The shot was a draw with substantial back spin. It was 146 m long.</td>
<td>GolfBot realized a hard hit and the fact that the shot had a lot of back spin. It did not classify the shot as a slice, but rather a strong fade.</td>
</tr>
<tr>
<td>A slight hook. A good hit and noticeable back spin.</td>
<td>The shot was a strong draw with substantial back spin. It was 158 m long.</td>
<td>GolfBot agreed on the back spin, but not on the shot shape.</td>
</tr>
<tr>
<td>A strong fade</td>
<td>The shot was a fade with substantial back spin. It was high with a height of 24 m and 170 m long</td>
<td>GolfBot realized the shot was a fade, but did not agree on how strong this fade was. It also determined it had a lot of back spin and was high, something the golfer did not comment on.</td>
</tr>
<tr>
<td>A hook. The apex is late, but it does not appear to have a lot of back spin</td>
<td>The shot was a hook with some back spin. It was 143 m long.</td>
<td>GolfBot agreed on both the shot shape and amount of back spin. A perfect match.</td>
</tr>
<tr>
<td>Hooked. It looks like he tried to hit it hard and got some back spin out of it</td>
<td>The shot was a substantial draw with substantial back spin. It was 158 m long.</td>
<td>GolfBot did not agree on the shot shape or the fact that it looked like a hard hit. It agreed on amount of back spin.</td>
</tr>
</tbody>
</table>

Table 7.1. A portion of the human shot interpretations in the testing set next to the GolfBot interpretation
Chapter 8

Discussion and Conclusion

8.1 Discussion

Throughout the project, focus has been maintained on keeping the academical sources relevant and of high quality. The research at the University of Aberdeen within the NLG field was early on suggested by our supervisor. This research, primarily lead by Dr. Ehud Reiter, is highly regarded in academical computer science circles was deemed applicable to the project. Many publications by Dr. Reiter and others were found, each providing a vital piece in the research. Unfortunately, not many sources were found without Dr. Reiter being a contributor. As a result, it is hard to regard these sources as absolute truth due to the low spread of authors. Thanks to the internationally recognized research within NLG by Dr. Reiter and they were still deemed of enough quality to be used. The SimpleNLG system developed at the University of Aberdeen has been used as it was helpful at the same time as it maintained the NLG structure outlined by Dr. Reiter.

The research was primarily to deduct what was important to portray back to users and threshold values to classify the shots and thus helped developing a functional NLG system. A scientific method verified by sources found was applied to the research in order to keep this relevant and applicable. One drawback to the method that was that it was time-demanding to give feedback on 50 golf shots. A result of this was that only a few golfers were willing to help. This undoubtedly affected the output of the NLG system as not as much data as preferred could be used to make the deductions about the shot. However, the interpretations provided were similar enough to still make deductions and the result was over expectations. The data provided by Protracer was also a big factor in achieving the result as it was extensive (>5000 shots to choose from) and easy to use.

The results presented in the previous chapter can be misleading as the classifications made by the system is based on the noted threshold values. The reality is different as golfers may interpret the same shot differently. Some may say a shot is a "substantial fade", while others may interpret it as a "slight slice". The line between these two is very thin. This was also portrayed in the results. Most of
the classification differences between the system’s interpretation and that by golfers were over this line, which decimated the result value shown in the previous chapter. This was as a result deemed close enough as most golfers would probably agree with the line being very thin. At the same time, however, this is speculative as not enough golfers willing to take the time to interpret 50 shots were found. This is an important aspect as the results may have been different had the golfer pool been bigger.

8.2 Conclusion

Based on the results presented earlier and the discussion in the previous chapter, the project was deemed successful. The classification, and as a result the output, is close to the human interpretation, structured in a grammatically correct and coherent way, based on observed data. GolfBot is also fast in its classification and output, and the CEO of Protracer has expressed interest in the system. Thus, even though further study on the robustness and usefulness of the system is needed, the system fulfills the requirements outlined in the Scope chapter.

Ultimately, the conclusion to be drawn is that the NLG system described in this report is suitable and applicable to the problem and that GolfCoach may be used in the future to further enhance the Protracer system.
Bibliography


