Answering Game Rulebook
Enquiries Through Natural Language Processing

ANTON BJÖÖRN
LUKAS UGGLA
Besvarande av regelboksfrågor genom språkteknologi

ANTON BJÖÖRN

LUKAS UGGLA
Abstract

The aim of this research project was to create a conversational interface for retrieving information from rulebooks. This conversational interface takes the shape of an assistant named OLGA (short for Open Legend Game Assistant) to whom you can give enquiries about the rules of any game loaded into the program. We tuned and designed the assistant around a specific type of board games called TRPGs (tabletop role playing games), hence the conversational interface is focused around game rulebooks. By giving the assistant the rules for a game in the form of a raw text document the program can extract key concepts and words from the rules which we call entities. The process of extracting entities and all other functions of the assistant were calibrated on the TRPG called Open Legend, hence the name Open Legend Game Assistant. When the user sends a query to the assistant it is first sent to the web service Dialogflow for interpretation. In Dialogflow we enter our extracted entities to assist the service in recognizing key words and concepts in the queries. Dialogflow then returns an object with information telling the assistant what the intent of the user’s query was and any additional information provided. The assistant then responds to the query. The standard response for a request for information about an entity is what we call a streak search. The assistant locates parts of the rules that contain the entity and sorts them by a relevance score, then the results are presented in order of relevance. When testing on people with no prior knowledge of the game it was concluded that the assistant indeed could be helpful in finding answers to rule questions in the limited amount of time provided. Generalization being one of our goals the program was also applied on another rule system in the TRPG genre, Pathfinder, applied on this system the assistant worked as intended without altering any algorithm.
Referat

## Contents

1 Introduction .................................................. 4  
1.1 Terminology .............................................. 4  
1.2 Problem Definition ....................................... 5  
1.3 Motivation ............................................... 5  
1.4 Approach ............................................... 6  

2 Background ................................................... 6  
2.1 Natural Language Processing ................................ 6  
2.2 Conversational Interfaces .................................. 7  
2.3 Third Party Tools .......................................... 7  
2.4 TRPGs ...................................................... 8  
2.5 The Games: Open Legend RPG and Pathfinder RPG ...... 8  

3 Method .......................................................... 9  
3.1 The General Structure of the Assistant .................. 9  
3.2 Entity Extraction ........................................... 10  
3.3 Stemming .................................................. 11  
3.4 The Standard Enquiry ..................................... 11  
3.5 Tuning of the Program .................................... 12  
3.6 Context Queries .......................................... 13  
3.7 Entity Usage Query ....................................... 13  

4 Evaluation ...................................................... 14  
4.1 Usefulness of the Assistant ............................... 14  
4.2 Design of the Questionnaire .............................. 15  
4.3 Usefulness Results ....................................... 15  
4.4 Application on the Pathfinder RPG ..................... 17  

5 Discussion ....................................................... 18  
5.1 Practicality for Users .................................... 18  
5.2 Generalized Application .................................. 19  
5.3 Improvements ............................................ 20  

6 Conclusion ..................................................... 21  

7 Appendix .......................................................... 23
1 Introduction

1.1 Terminology

**NLP - Natural Language Processing**
The field of handling natural language, language used between humans, with computers and turning it into something usable by the computer [13].

**NLI - Natural Language Interface**
An interface designed so that a human may interact with a program naturally using normal conversational language, whether through speech recognition or text input.

**NLTK - Natural Language Toolkit**
A library of modules for the Python programming language useful in dealing with natural language. Includes functions such as tokenization, sentence detection, word stemming, word frequency distributions and part of speech tagging, see section 2.3 [11].

**POS-tagging - Part of speech tagging**
Part of speech tagging is the process of assigning part of speech tags such as noun, verb or modal to words in a text. Done correctly by the computer this makes handling the text easier and may allow more advanced processing such as extracting meaning and information from the text.

**TTS - Text To Speech**
The process of converting written text to intelligible speech using a computer.

**Assistant**
In this report we refer to our conversational interface as ”the assistant”, that is the program which has the aim of answering questions about game rules.

**Query**
The natural language message entered as text into the program by the user to be sent to the assistant. The query is sent to Dialogflow which interprets meaning from it which is then relayed back to the program.

**Intent**
An instruction entered into Dialogflow telling the agent how to interpret a specific group of inputs which share the same general intent on the users behalf.

**Entity**
A list of words and synonyms entered into Dialogflow to help it distinguish them in a query. In our case the most important entity is all of the relevant concepts from the game on which the assistant is being used.
JSON - Java Script Object Notation
A common notation for storing a programming objects information into a text file allowing for easy transfer and conversion of the file back into an object again.

Stemming
A stemmer is a function that cuts down words to their basic "stem" such as turning the word "authorities" into "author". This has the advantage of allowing a search for the word "authority" to match the word "authorities" but may also lead to undesired words such as "authors" to match.

Tokenization
A tokenizer or tokenization function breaks up a text into a list of "tokens". In this project we use NLTK’s tokenizers both to detect sentences and to split sentences into smaller parts such as words, punctuation and possessive suffixes ”’s”.

TRPG - Tabletop Roleplaying Game
See section 2.4.

OGL - Open Game License
Games published under the Open Game License grant consumers the right to modify, copy and redistribute some of the content within certain limits. This is why all the rules for the Pathfinder RPG are available for free online but rewritten as not to exactly copy the contents of the published books. So while rule explanations are different the mechanics of the game are identical [14].

1.2 Problem Definition
In this report our aim is to determine the usefulness of a general purpose conversational interface, made using free modern tools, in helping the user find answers to questions about rules in a game (more specifically TRPGs). We will investigate whether using this interface will prove more effective than manually navigating a website containing all the rules.

1.3 Motivation
The idea for this project comes from our own experience of playing TRPGs. These games usually want to maximize freedom for the players to do whatever they want, while still following the rules of the game. This leads to the need of having a large amount of rules to account for the vastly varying scenarios that might occur. Rulebooks therefore tend to be quite substantial collections of text and sorting through them for a specific rule can be very time consuming especially when one is new to the system. This is where we hope our research will help.
Instead of being forced to have at least one player at the table who has to memorize all the obscure rules, it would be practical to have a digital version of this player. An assistant that could save a lot of time otherwise spent looking through rulebooks.

1.4 Approach

Our first approach for creating a useful assistant was mapping the different rules and processes of the game into a conversational system. Our thinking was that many TRPGs use similar thinking and concepts such as hit points, dice rolls and character traits. The result of this mapping was promising, but we quickly realized that our way of thinking would fundamentally limit our scope and could not easily be expanded into other rule systems. Upon reevaluating our research we decided to change our approach and delve deeper into natural language processing to achieve a higher level of generalization resulting in a more exciting research project. Our approach is to only offer the assistant a raw text file of all the rules in a game. From this text file it extracts entities automatically allowing it to search for parts of the rules pertaining to those entities when prompted by the user.

Our entity and information extraction approaches in this project are based on conventional NLP techniques which we learned about from the official book on using the Python library NLTK [13]. We combine these conventional techniques with a conversational user interface powered by machine learning which is a comparably modern approach to creating a user interface. This part uses technology now owned and developed by Google which you can read about in section 2.3.

2 Background

2.1 Natural Language Processing

Research into making computers interpret natural language and gain information from it goes back more than 40 years [7]. Humans can easily understand and interpret what someone means when they talk or write in a language they are proficient with, but for computers it is far from a simple task. The technology where computers process natural language used to be closely related to AI research but in recent years it is more often classified as text mining and is used in many different contexts, not only Artificial Intelligence.

We’ve been able to do quite sophisticated NLP for years now. Despite this, using NLP to access information from large amounts of texts is not part of our everyday work. In a book on the area first published in 1992 Paul Schafran Jacobs already addressed this. It is stated there that often the strengths of the currently used methods are ignored, namely simple text search, Boolean queries
and keyword retrieval. They are easy to learn and if used correctly very powerful tools in finding information [5]. Most of us are very comfortable with these standard ways of searching for information. Online search engines might use more advanced methods but when we are presented with a large pdf, website or text file searching within it is usually done with these methods. In order to compete it is necessary for information extraction approaches using NLP to perform very well and to be intuitive to use; If a general purpose information extraction algorithm is not robust and functional enough it won’t see use.

One cannot ignore the very systematic and code-like nature of language. While there is ambiguity in most natural language, which requires a human understanding to interpret, a lot of text is also written in a very factual and structured way. Rulebooks are an example of the types of texts that often lack this ambiguity seeing as it is necessary that the rules can only be interpreted one way, suiting them well for NLP approaches.

### 2.2 Conversational Interfaces

For a long period of time, the technology of being able to have a conversation with a machine was limited to science fiction. But since the first real conversational interfaces were introduced into many people’s daily lives in recent years these programs have become better and better and more widely used [8]. Google Assistant and Apple’s Siri are good examples of assistants to which a user can talk via their phone or computer using natural language. These virtual assistants can tell you jokes, set up meetings and answer factual questions. As our intelligent devices grow ever more powerful and accepted (and with the gradual introduction of the Internet of Things) conversational interfaces may become a big part of many peoples daily lives. Today we are at a point in time where quite sophisticated conversational technology is available for free which enables almost anyone to create a useful conversational interface for a specific need.

### 2.3 Third Party Tools

NLTK was released 2005 and today it is one of the most popular tools for handling text and natural language in Python [12]. NLTK not only has its own tools built in but grants the user easy access to common NLP tools and resources made by others such as stemmers, WordNet and over 50 corpora and lexical resources [11][4].

We have chosen to use NLTK for a number of reasons, including it being easy to use, free, and having built in tokenization, POS-tagging and stemmers. Besides all that NLTK has a great book on its use and NLP in general available on their website. The book is called Natural Language Processing with Python and was used throughout the project [13].
After some research into different alternatives for how to implement a conversational interface we decided to use Dialogflow. Well-functioning conversational interfaces use extremely advanced NLP and machine learning algorithms, but the Google owned service Dialogflow (formerly known as api.ai and Speaktoit) helps solve this problem so that the wheel doesn’t have to be reinvented (and this is one non-trivial wheel) [9]. Using machine learning and years of data Dialogflow helps the assistant to be easily communicated with in a natural way. See section 3 for implementation.

2.4 TRPGs

A TRPG (or Tabletop Roleplaying Game) is as the name would suggest a game where a group of people (usually 5-6) sit around a table and take on the roles of imaginary characters [6]. These characters then interact with the world which is described by a person called a 'Game Master' who acts like a storyteller and referee. The results of these interactions are determined by rules, which vary between rule systems. Different rule systems determine what you can and cannot do, and the results of different interactions. Roleplayers and the Game Master use dice to determine outcomes, and they usually follow the rule system quite thoroughly as long as a rule exists for the interaction at hand. Since many TRPGs want to maximize the freedom of choice for the players this can lead to a lot of rules for specific scenarios. Whether it is holding your breath, climbing a wall or simply attacking a monster there will likely be an applicable rule for most scenarios.

2.5 The Games: Open Legend RPG and Pathfinder RPG

Our assistant is built to work on game rules in general but for development purposes we picked the game Open Legend as a starting point. Open Legend is an open-source game available for free online, it is created and copyrighted by Seventh Sphere Entertainment.[1]

Compared to most board games, the rules of Open Legend are quite complex and very lengthy, but when comparing with many other TRPGs they are actually on the modest side. They are also well suited for tuning our assistant since the system is built to function in all genres of RPG, everything from ultra realism to science fiction and fantasy; the Open Legend rule system should be a good representation for most TRPGs. The TRPG Pathfinder on the other hand is in our experience one of the most rules heavy games around.

The Pathfinder RPG is published under the Open Game License (OGL)[3] (see 1.1) and therefore the rules are available for free online but they are structured in such a way there that they may not simply be turned into plain text without some amount of manual work. Therefore we chose to use a pdf of the Pathfinder Core Rulebook instead since it will tell us how our system performs on books.
while also being easy to turn into plain text. Pathfinder being a complicated system, the plain text file for the rules is actually more than eight times the size of the text file for the rules of Open Legend. We should expect Pathfinder to pose a bigger challenge for the assistant as it has a much greater amount of concepts and being written as a book rather than a web page it probably has lower information density (more superfluous sentences and longer explanations).

3 Method

3.1 The General Structure of the Assistant

![Flow chart describing the general structure of the program](image)

Booting up the assistant greets the user with the title "O.L.G.A", which stands for Open Legend Game Assistant, along with some basic information on how to use the program. Below the introductory message the assistant’s first reply appears, for example "OLGA: Hello", indicating that the connection to Dialogflow (see section 2.3) has been established and that the user may now enter a query in the form of text written in natural language.

Upon entering the query the program then sends it to Dialogflow for interpretation. In Dialogflow we have defined instructions for how to interpret the query in the form of intents and entities. Dialogflow uses these and returns a JSON object to the assistant serving as instructions for what action the program should take to solve the query.
Upon receiving the JSON object the assistant resolves the query appropriately, such as by searching for relevant information in the rules, giving more information about a previous query or ending the enquiry into a topic.

Figure 2: The greeting screen when launching OLGA

3.2 Entity Extraction

In order to allow the assistant to work on any rule system and on large amounts of text it is necessary that the extraction of entities is not done manually. Therefore we developed a method for extracting important keywords and concepts automatically from the text.
The basics of our entity extraction is to pick out the words that occur with the highest frequency while filtering out short words and common non-informative words such as "every", "would" and "using". We also pick the most frequently occurring bigrams and trigrams, once again filtering out non-informative words to avoid a common collocation such as "every time". This generally produces entities that are more informative and unique to the text used.

(Natural Language Processing with Python, Chapter 1 [13])

3.3 Stemming

When searching through text for a word or collocation in a broader sense than just looking for a particular array of characters a stemmer can be very useful. A stemmer cuts down words to their basic "stem" such as turning the word "authorities" into "author" which is desirable since we want words like "attacker" and "attacking" to match an enquiry on how to perform "attacks" in the game. NLTK has a selection of stemmers built into it which we tried out looking for a more aggressive stemmer. An aggressive stemmer cuts words down more allowing for more possible matches which was desirable in order to make sure we didn’t miss any matches. We decided to use NLTK’s ported version of the Snowball stemmers developed by Dr Martin Porter [10].

3.4 The Standard Enquiry

A standard enquiry is triggered by phrases such as "What is X" or "Tell me about X" where "X" is one or more words recognized by Dialogflow as an entity from the game. If the entity is left out completely the assistant will ask the user what the enquiry is about after which the user may simply answer with only the entity.

Once a correct query is registered the program responds with what we have named a "streak search". The streak search looks for sentences containing the relevant entity, ignoring capitalization and stemming the words in order to match different word forms. Upon a match the sentence is saved and the streak search proceeds to check the following sentence. Each following sentence that matches is connected to the previous, forming a "streak" of matching sentences. Each match raises the score of these collections of sentences. As the streak grows longer how much the score is raised for each match increases linearly. Thus longer collections of matching sentences score higher. Afterwards all found streaks are sorted according to score and the highest scoring streak is presented to the user. At this point the user is asked if they need more information. If the answer is yes the assistant first shows the entirety of the highest scoring streak, if it was very long, and additional requests for more information beyond that presents the next highest scoring streak and so on. For an example of exactly how this would look see figure 4 in the Appendix.
When searching for a collection of words such as the bigram "minor action" or the trigram "total party level" it is less likely that the entity will appear in many sentences in a row. Therefore the streak search function is modified to allow for searching ahead a few sentences. How far ahead the streak search looks depends on the length of the current streak, a longer streak lets the search continue further while a single matching sentence will only look two sentences ahead. When no more sentences are found any sentences at the end of the streak with a score of zero are removed. The results are then presented to the users just as when searching for a single word.

In a streak search for a collocation we also add to the score of a sentence for each part of the collocation that appears. This boosts the score of relevant sentences even if they did not match the entire collocation.

There is one more factor that affects the score that a matching sentence grants to a streak. We found that different forms of the word "you" are especially important in game rules since it almost always refers to the reader or the player. Therefore if a sentence contains any form of the word "you" that sentence's score is raised by a flat amount. Basically sentences containing different forms of "you" usually treat what the player(s) may or may not do i.e rules for the game.

3.5 Tuning of the Program

In our streak search and entity extraction there are many parameters that tune the performance of the program.

Some examples in the streak search are:
- how much score is granted from a matching sentence in a streak search,
- how much streak score increases with the length of the streak,
- how much the score is raised by the presence of the word "you",
- how many sentences ahead the search continues when searching for a collocation.

In the entity extraction the following parameters are tuned:
- how many of the most frequent words/collocations are extracted,
- the minimum length a word needs to be for extraction,
- what POS-tags should be excluded.

The program was tuned for extracting the best results using the TRPG Open Legend. Once tuned to work well on this game our hypothesis was that it would work well for all similar games without need for retuning. For the streak search we came up with typical queries and manually located the part of the rules text that would answer the query best. We then tweaked the parameters so that this part would score the highest. After repeating this process on many queries we found a set of parameters that worked well and solved most queries in a satisfactory manner.
For the entity extraction tuning was performed by comparing the set of entities extracted before and after each tuning. By comparing the sets we could see what words had been filtered out. We kept making the parameters stricter until entities that were remotely relevant started being filtered. Since Dialogflow can handle a large amount of entities we decided to rather go for a too large list of entities since we want to make sure no important concepts or keywords were lost. The only disadvantage to extracting a large amount of entities is the risk of words being identified as entities incorrectly. However since short words and common non-informative words have already been filtered out extracting a large amount of entities proved to work well.

3.6 Context Queries

When browsing the rulebook via the assistant the user could encounter an extract from the rules that is too short, or maybe they simply want to keep reading. In other cases the extract doesn’t make sense without the previous sentences. In these situations allowing the user to query for more context is important.

We designed two types of context queries to meet these needs. The first is triggered by simply asking the assistant "what comes after/before" the currently presented text. This query allows the user to scroll back and forth through the text at will.

The second context method can be used to gain context anywhere in the rulebook. Each sentence that the assistant presents is given a number (1,2,3,...) telling the user that it is for example sentence number 2048 in the rulebook. These numbers are always printed at the start of each sentence. This query is triggered by asking for context around a sentence. If a specific sentence number is not provided the assistant will ask around which sentence the user requires context. The user is then presented with a block of sentences around their chosen sentence. The assistant then asks if even more context is needed to which the user can reply "yes" repeatedly to expand the amount of context presented around the chosen sentence.

3.7 Entity Usage Query

After examining the structure of many sentences in the rulebook we found a simple formula for answering a specific question. When the user asks "What can I do with X?" or "What can X do?", where "X" is an entity from the rules, the answer usually occurs in a sentence containing a modal word. Some of the most common modal words in English are "can", "may", "will" and "must". An example would be the user asking "What can I do with Energy?" and the answer being found in sentences like "Energy may be used to cast spells that deal damage to enemies.".
We designed an intent in Dialogflow for these queries and constructed a simple search algorithm. The algorithm looks for sentences which contains a modal word and then splits it in half at the position of the first modal word found. If the first half of the sentence contains the entity which the user queried for then the sentence is a match and presented to the user. Using this query all matching sentences are presented even though they were not adjacent to each other. Thus the user receives a list of sentences of which many will be informative about the uses of the entity.

While this query is available in the assistant it was not used too extensively since it is very specific. When the program was evaluated using a questionnaire (see section 4.1) the testers were instructed to use the standard enquiry instead because it is broad and more functional. In order to allow users to issue more specific queries we feel that many more specialized algorithms would need to be developed to handle them differently.

4 Evaluation

4.1 Usefulness of the Assistant

In order to evaluate the usefulness of our assistant we designed a questionnaire with rule questions about the Open Legend RPG. Since the normal way of looking up the rules is by using the Open Legend website [2] we wanted to compare that to our program. We therefore found testers with no knowledge of Open Legend, and often no knowledge of TRPGs in general, and had half of them attempt the questionnaire using the website and the other half using our assistant.

The people using the program were first given instructions on how to use the program. This was important since most people are already familiar with how to navigate a web page but not our assistant, a factor which we wanted to mitigate at least slightly. After that both groups were given a test question to answer using their respective resource. Answering the test question gave both groups time to familiarize themselves with the website or program and to the structure of the rest of the questionnaire. Each question in the questionnaire has four alternatives, only one of which is correct, and is a question about the rules in Open Legend.

After answering the test question the tester was allowed to turn over the questionnaire revealing 5 questions in the same style as the one they just answered. They were then given 10 minutes to answer as many of them as possible. Before turning over the questionnaire they were instructed not to guess randomly but to base their answer on something in the rule’s text. They were also instructed that they may answer the questions in any order.
When the ten minutes had passed or the tester believed they had completed all the questions the time of completion was noted down along with the amount of correct and wrong answers.

When performing the questionnaire tests we made sure to closely observe the testers progress to gather the less numerical but very important data on how they used the program/website. We wanted to see what prevented some questions from being answered, what went wrong and what techniques the testers applied etc.

4.2 Design of the Questionnaire

When designing the questionnaire we made sure to design questions that were not known to be answerable by the assistant. All rules are of course answerable using the website but since the performance of our program varies a lot depending on what concept one asks for information about it was important that the questions were designed without the program in mind.

Each question in the questionnaire was given four alternatives. This served to help the testers know what information found in the rules was actually relevant and to simulate that the tester has some knowledge about Open Legend, since the alternatives themselves could be used as leads to answering the question. Often when pondering a question about rules in a TRPG you know it must be one of a few alternatives so giving alternatives fits in this way as well.

The questionnaire can be found in its entirety in the appendix (section 7).

4.3 Usefulness Results

The results from the questionnaires are presented in two tables below, table 1 is from the testers using the assistant and table 2 is from those using the website.

<table>
<thead>
<tr>
<th>Tester nr</th>
<th>Time [s]</th>
<th>Correct Answers</th>
<th>Incorrect Answers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>600</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>600</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>600</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>600</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>600</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

Average: 2.4  Average: 0

Table 1: Questionnaire results from testers using the assistant

Unfortunately we were not able to find and gather results from a large amount of testers. The size of our group of testers will definitely limit the certainty of anything the results may indicate.
<table>
<thead>
<tr>
<th>Tester nr</th>
<th>Time [s]</th>
<th>Correct Answers</th>
<th>Incorrect Answers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>600</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>600</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>600</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>600</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>479</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Average: 2.4</td>
<td>Average: 0.2</td>
</tr>
</tbody>
</table>

Table 2: Questionnaire results from testers using the website

Figure 3: The amount of correct answers received to the 5 different questions on the questionnaire for each test group

When performing the tests we noted that many people got the same questions right, question 2 & 3. The program clearly performed well on those two questions which indicates that the results are very dependant on the exact questions used in the questionnaire. A greater number of questions would perhaps reduce this uncertainty.

When using the program many testers were close to finding the answer to a question had they only asked the assistant for more information about their current enquiry. Instead they often opted to change enquiry in order to perform a wider search for the answer. This resulted in fewer testers finding the answer to questions for which the answer was not given in the top streak given by the assistant.
If the answer was given inside a larger amount of text, which was the case for question 4, it also seemed to make fewer testers find it.

When navigating the website testers who were good at using the web browser’s ”find” feature, which searches the current page for a string of text, found answers more quickly and performed better.

The biggest problem for finding information on the website seemed to be that the rules are split into 9 different chapters. It was difficult for the testers to know which chapter contained the information they sought since the names of the chapters did not indicate that the information would be found there.

Website tester nr 5 is a potential statistical outlier since this tester had extensive experience with browsing TRPG rules on a web page.

We noted that tester 1 & 3 from the website group and testers 1, 3 & 5 from the assistant group had particularly little experience with RPG games in general (not just board games).

The ”Entity Usage Query”, see section 3.7, did not see use in the questionnaire evaluation. It would likely have been useful in answering question 3. ”What can a character’s Presence attribute be used for?” since it would present only two sentences, one of which contains the correct answer. The standard enquiry provides the same sentence but along with more text making it easier to miss. It should have been a help for answering question 2 ”Which of the following can a player do using a Minor Action?” as well but due to the uses of a Minor Action being presented without a modal word and in a list the algorithm fails to pick them up. Since both question 2 & 3 were among the questions most commonly answered correctly (figure 3) the query seems to not have been necessary.

4.4 Application on the Pathfinder RPG

The application of the assistant on another game will be evaluated on three points: Entity extraction, Enquiry satisfaction and General performance.

Entity Extraction

The set of entities extracted seem to be very complete. We could not identify any important concepts or keywords to have been left out by the algorithm. However since the assistant is set to pick out all the collocations that occur more than a certain number of times and since it was tuned to a much smaller text a great amount of collocations were extracted. It seems these extra collocations contain a lot (at least 50%) of useless combinations that a user would never use.
Enquiry Satisfaction
It would seem that the completeness of the set of entities allows for an enquiry into almost any topic the user is likely to ask for. The assistant successfully picks out relevant parts of the rulebook containing relevant information for many enquiries. Just like for Open Legend the assistant does fail for some enquiries and requires the user to ask for more information or more context several times before producing a desirable extract.

General Performance
The program is slower when run on the Pathfinder Core Rulebook, likely due to it being more than 8 times larger. For the entity extraction which is done once this does not matter, and the boot up of the program only has to be performed once when you want to use the assistant. However since each enquiry needs to search through the rules it has been slowed down so much that the usability of the program suffers.

5 Discussion
5.1 Practicality for Users
At a first glance the results from table 1 and 2 seem to indicate quite a similar average performance. The assistant however has a lower spread of results (all testers had either 2 or 3 correct answers) while the website got correct answers ranging from all (5) to none (0). All of this should of course be considered keeping in mind that the number of testers was small.

We would like to point to the fact that the highest performing tester and the only one to complete all questions within the given 10 minutes had a lot of experience looking up rules for TRPGs on a website, having played the Pathfinder RPG, see section 2.5. Looking at the data without considering this potential statistical outlier would make the results indicate a higher performance from our program compared to the website.

Looking at the average amount of correct answers in 10 minutes might not be the most interesting result though. In figure 3 we can see the amount of correct answers to each question in the questionnaire separated by test group. Here we can see that the website’s performance seems to be more even while the assistant performs exceptionally well on questions 2 and 3. In fact every tester who used the assistant got those questions right.

What questions 2 and 3 have in common is that both ask for the use of a concept of the rules but also that the queries ”tell me about Minor Action” or ”what is Presence” makes the assistant present the correct answer in the first hit of the streak search; the tester did not have to dig deeper into the enquiry to find the answer. Our problem for the other questions then might be that
the answer was not in the highest scoring streak received from queries about
the main concept in the question statement. We have confirmed that indeed
this is the case. For example, for question 1 "tell me about hit points" does
not instantly present the right answer in the highest scoring hit, the user needs
to type "yes" for more information twice. One explanation to this is that the
standard enquiry answers the question "What is X?" quite well and question
2 and 3 are the most similar to that question. To present the correct answers
to questions 1, 4 and 5 in the top hit we would probably need different algorithms.

An important thing to note is that we as the creators of the assistant are very
efficient in using it to find information. This is partly due to our knowledge
about the game system it is being applied on but mostly because we are famil-
iar with how the program works. Since people have a lot of experience using web
browsers we made sure to instruct people properly in how to use the program.
We roughly described how the program works and let them read the introd-
tory instructions of the program, see figure 2. With just about 2 minutes of
instruction followed by answering the test question, which both groups did, the
program performed as well and arguably a lot better for some questions than
the web browser, a tool which people were very familiar with.

The language of TRPG rulebooks can be quite confusing to new players. Some
of the testers had no previous experience of TRPGs at all but despite this they
were able to find answers to questions about a game system they knew nothing
about when using the program. Using the program testers 1, 3 and 5 got 3, 2
and 3 correct answers respectively. The most inexperienced testers in the web-
site group were testers 1 and 3 who got 0 and 1 correct answers respectively.
Here we must note that our data is too small to indicate anything with confi-
dence but this would indicate that the program is especially useful to new and
inexperienced players.

5.2 Generalized Application

The nature of our approach being to simply work from a text file greatly widens
the range of rule systems that the assistant can be applied to. Since no part of
our program except the working name OLGA actually binds us to any particular
system the results of our Pathfinder implementation (see section 4.4) is not very
surprising. Since our questionnaire-results (section 4.3) also seem to indicate
the assistant is useful for people not familiar with the rule system or in fact
TRPGs at all, one could assume that it would be very useful for looking up
particular rules when testing out a new system, without having to familiarize
one self with a new book. There was of course the problem of speed when
running the assistant on very big systems, this is addressed in section 5.3 below.
5.3 Improvements

There are a number of additional functions that could potentially have been useful for the assistant but for which there was not enough time to properly implement. For example a syntax like "How do I X my/the/a Y?" could allow you to first streak search for Y but then filter out streaks not containing X. This would have solved question 1 in the questionnaire easily with "How do I raise my hit points?". There are many possible similar syntaxes that would allow the user to pose more specific questions allowing for further refining of the information presented by the assistant. Given the nature of the program being a conversational interface additional functionality could be added without clogging any visual interface, so one could argue that in this case more would be better.

During our testing we noticed a difference in how we, the developers, used the program and how the testers who were just introduced to it did. We’d say that due to the conversational nature of the interface it helps to know exactly how the language is interpreted and what intents exist in Dialogflow. A way to make the program more accessible would be to add further conversational fluidity so that the assistant tries to help the user to find the correct function. This could be realized by improving the use of Dialogflow.

NLP was used to analyze the rule texts, but not everything in rulebooks are written with natural language. One thing that potentially could be handled completely separately is tables and lists. As it stands tables and lists are treated almost the same as normal text, but if they could be categorized and handled in a completely different way one could redesign some of the functionality around it to for example present table results more clearly and treat list headers (for example if there is a list of traits to choose from, the name of the trait would be a header) with more significance.

When applying the program on the very extensive rules of Pathfinder, we noted the program worked very slowly. If we want to further widen the generalization we therefore need to make the assistant quicker at answering questions. Besides normal optimization in the code, we have identified two potential ways of speeding up the answers. The first is that when extracting entities we could go through and save the sentence numbers where the entity can be found. Then when searching for a certain entity the process would be much faster as we could use only those sentences. A more extreme version of this would be to pre-generate the sentences that would be replied if a certain entity is asked about. This would greatly increase the response time but this pre-generation would likely take quite some time in itself, so perhaps the amount of entities would have to be decreased since it will be done for each entity.
6 Conclusion

TRPG rulebooks are complex, and often a gaming group needs at least one player that is very proficient with the rule system to be able to answer questions as the game goes on, or the game will deteriorate into flipping through endless tomes instead of playing. We set out to investigate if we could create a digital assistant that could fulfill the role of the player that knows all the rules at the table. An assistant you could ask questions just in the same way as you would ask a human. Using natural language processing and Dialogflow we were able to create a program that from a plain text file containing the rules can automatically extract entities and create a rules assistant. Looking back we conclude that we have achieved our goal, and while it certainly has areas of improvement, our results indicate that the assistant already is useful for answering game rulebook enquiries. This method is potentially also useful for other situations where information extraction from a text is an important part of a conversational interface, since no intrinsic part our our project actually depends specifically on board games. We hope to see further development in programs like this in the near future. For now we will continue to use and improve our own assistant until a superior, perhaps completely general purpose, assistant comes along.

References


7 Appendix

Figure 4: An example of how OLGA (the assistant) works

OLGA: Hi!
What can I do?
OLGA: I am a game rulebook assistant. You can ask me to elaborate on concepts from the game rules by giving me enquiries.

Examples of enquiries:
- What can I do with a major action?
- What are the basic game attributes?

Upon receiving information, I might ask you if you need more. If you do, I will give you other relevant parts of the rules. Alternatively, you can ask for more context by for example writing:

- What comes first, they?
- Why didn't develop?

You can also ask for context around a sentence:

"Give me more context around 1:34."

Tell me about Fortitude
OLGA: Let's see... Looking up "Fortitude" for you...
OLGA: Fortitude
Fortitude governs your ability to withstand physical punishment.
Fortitude has no other functions including dealing off diseases, preventing malnutrition, and weathering long stretches without rest.

In most cases, you'll be making Fortitude rolls as the GM is prompting you to make a roll. These are essentially yes or no choices. Your Fortitude is used to pass or fail a series of tasks that represent a group of actions through the process.

OLGA: Have that answers it, or do you need more information?

yes
OLGA: 270 KNP POINTS > 2 x (FORTITUDE + PRESENCE + WILL) = 30.
OLGA: Need more?

no
OLGA: (Okay) What else...
Website or OLGA: _______ Date: _______

Participant's name: _______________________

If you are using OLGA please read the instructions presented to you on the screen. After that regardless whether you are using the website or OLGA familiarize yourself with the program/website by answering this test question:

**Test**

When are you allowed to make an opportunity attack?

[ ] Whenever an enemy tries to flee
[ ] If you have a weapon equipped when an enemy flees
[ ] If you can attack the enemy when they move away from you
[ ] If you have a melee weapon equipped and an enemy moves away from you

You will now be given 10 minutes to answer the 5 questions on the other side of this sheet. There is only one correct answer to each question, if you have no answer please do not guess by ticking a box randomly. You do not need to answer them in a specific order, if you find that you are stuck on a question you may move on to the next one and return later.

Good Luck!

Figure 5: Questionnaire, Page 1/2
1. Which attribute scores raise a character's max hit points (hp)?
   [ ] Toughness, Strength & Constitution
   [ ] Fortitude, Presence & Will
   [ ] Protection, Fortitude & Might
   [ ] Might, Protection & Will

2. Which of the following can a player do using a Minor Action?
   [ ] Move their character
   [ ] Assist an ally
   [ ] Attack an enemy
   [ ] Draw a weapon

3. What can a character's Presence attribute be used for?
   [ ] To persuade people and see through lies
   [ ] Cast magical spells
   [ ] Inspire allies and scare away enemies
   [ ] Retain focus on a difficult task while being distracted

4. A player is attacking an enemy using a longsword. He/she rolls an Agility roll to hit the target. To hit this attack roll needs to exceed the opponents...
   [ ] Armor Class
   [ ] Agility
   [ ] Guard
   [ ] Toughness

5. Which of these statements is true about how the combat turn order in Open Legend works?
   [ ] Each player performs a set of different actions when it is their turn
   [ ] The oldest player takes his/her turn first.
   [ ] Each player performs one action when it is their turn
   [ ] Players may perform two actions each round, one on their turn and one on a friend's turn

TIME (s):__________  CORRECT:____  WRONG:____