UNIVERSITY OF GOTHENBURG

Developing an Interactive Communicative Search Agent

LAUREN AKIF

Master of Communications Thesis

Report No. 2011:043
ISSN: 1651-4769

University of Gothenburg
Department of Applied Information Technology
Gothenburg, Sweden, June 2011

I would like to dedicate to my parents, Joseph Akif and Karen Akif,
all of whom have constantly supported me and guided me
throughout my studies both at home and abroad.

Acknowledgements

I would like to express my gratitude to Keisuke Takahashi for his aid in developing Latex documentation and for programming assistance, my classmates for encouraging me, and my advisors Alexander Almer and Claes Strannegard for their help and guidance.

Abstract

Developments in semantic web are changing how websites are written for the internet. The categorization principles are applied to closed systems such as a company's internal document database; however, these changes are not as quickly reflected in semantic search due to existing problems within the natural language programming field.

This project aims to develop a framework for a different type of search: one that combines linguistic principles and current search options and tailor returned results according to user input. The Python programming environment and Natural Language Toolkit libraries are the primary resources used to develop this program.

Once a basic working code is developed, search results are compared against an increasingly-popular alternative search tool. The results from these searches provide feedback on which areas the program needs to develop further.

Keywords: semantic search, human-machine interaction, question-answering, natural language programming, communications

Contents

List of Figures

Chapter 1

Introduction

Proper knowledge representation within machines is currently quite difficult to achieve within the AI field. Machines need to understand objects, their relations with others, properties, knowledge about knowledge, and other similar types of information in order to behave more intelligently. Currently, machines only rely on definitions and instructions written by humans. Not only does this make the machine reliant onto manual input, but without proper algorithms and logics put in place, it cannot learn new data on its own. They are unable to learn new information from other machines, making their databases increasingly outdated. In order to develop intelligent machines further, a new way of accessing and presenting data should be investigated.

1.1 Semantic Web and Implications

The 'Semantic Web', proposed by Tim Berners-Lee [1], is an 'upgrade' of current HTML. Semantic Web employs a more intelligent tagging system by telling the machine what the user means instead of telling it what to show. For example, with current HTML tagging, a user will bold a word by using the $< b >$ and $< /b >$ tags. With this, the machine will simply make text between those tags show up in bold. Often, when a user bolds a text, they actually want to show emphasis on that word. However, the machine will not recognize this distinction. By using a more sensible tag such as $< em >$ for 'emphasis', the machine will understand that

the user is placing importance on that phrase, much like how tone is used in speech to attract attention. Through the use of the Resource Description Framework, RDF, this is currently being made possible as users begin to implement the meta-tagging idea into a 'semantic desktop' as well as develop different indexes and ontologies.[2]

Current work is mainly focused on developing the Semantic Web itself or developing a semantic desktop. With projects such as SIMILE[3], SIOC[4], and NextBio[5], the use of a semantic framework is spreading rapidly into many different fields. Currently, there is a growing demand for medical and bioscience applications. With this new organization system, new types of relationships can be taken advantage of and accessed through new routes, rather than from a strict start-to-finish point of view. For example, a symptom such as a fever can be put into the system and a list of related symptoms and sources of these symptoms will be appear. By narrowing down the search with other symptoms, which are now acting as search parameters, correct information can be reached by way of relationship. This differs from current methods, where searching involved manual pruning through lists of articles by the user. Businesses are also using these systems internally as an aid in reorganization internalized company documents, databases, and other such tools.

Question-answering systems can also benefit from this development and become more useful when combined with work in natural language processing. First generation question-answering systems lacked linguistic input and focused on words as the smallest unit of meaning. This proved to be limiting, as seen in Simmons.[6] Later generations of question-answering systems developed framework that began to establish referential relationships as well as a basic semantic memory. However, with the rapid development of technology, the variables and and networks available have grown exponentially. Thus far, a system has not been developed that understands human input naturally. Semantic search tools have been attempted, but have either shut down or have not been very successful. Dbpedia[7] is realizing semantic web classification paradigms by reorganizing Wikipedia results according to entry's membership and relationships with other category classes. Wolfram Alpha[8] may be considered an exception to this, yet it is still being developed and it has been yet to see whether Wolfram's compu-

tational approach[9] will yield the same results as a semantic search would.

1.2 Current NLP Problems

The human mind can make an infinite number of connections between the meanings and knowledge stored within it. Language provides a reference point to these meanings by establishing a standard word for use within the specific language community. For example, when asked to imagine a cat, each person's image will most likely have similar characteristics, but will not be exactly the same. Assuming that the user is thinking of the animal, when asked to describe a cat, descriptors like "furry", "four legs", and "domesticated" are often called upon. However, these descriptors are not solely relevant to this particular animal. Bears are also furry, deer have four legs, and dogs are also domesticated. Without experiencing these meanings and having the proper means of making those connections, however, humans will not be able to describe what they are thinking.

This dilemma is currently present within machine intelligence. While a list of these descriptions may be presented, the machine is unable to establish and understand the variable relationships between each characteristic. Additionally, without pre-programmed expectations, the program cannot understand new information that has been unaccounted for. One can speculate that this issue is tied to a lack of understanding of how Language truly functions. As it is ethically wrong to simply cut open a brain and test to see which areas of the mind affect speech and which areas are activated during different speech activities, researchers are left to analyze and construct the working framework of Language from only its output (or lack of output, as seen in those with disabilities or injuries).

However, re-evaluating syntactic theory and applying those principles with other sentence- and context-bound information, in combination with other resources, may return new results in terms of the development of a more search-friendly agent.

1.3 Purpose

The purpose of this project is to develop a basic framework for a search agent that displays behavior similar to a human while taking advantage of semantic search methods. It should incorporate linguistic cues from user input into its search and return results that are tailored towards the type of information that an average user is looking for. Search behaviors and relevancy are enhanced by Google's search framework [10] along with a hierarchy of databases classified according to the type of question being asked.

1.4 Perspective

This project approaches developing an interactive search agent with a humanistic point of view. Machines and programs are useful aids and should not be feared. Users shouldn't feel intimidated by a communication agent, nor should users feel alienated when interacting with the agent. The program can do this by mimicking how humans communicate with other humans. For simplicity, the program will be referred to as Mordinn for the rest of the document.

Chapter 2

Theoretical Framework

In developing a search agent that utilizes Language like a human, human interactions should also be analyzed. The environment that humans interact in, the relationships between people, and social expectations all come into play when an interaction takes place.

2.1 Models: Human

According to Knapp and Vangelisti[11], the majority of human behavior is governed by inclusion, control and affection needs. The need to feel included, control over a situation, and positive social reception directly affect how people will interact with each other. These needs are often fulfilled through positive, engaging conversation and activities with other people. With recent growing trends of youth fulfilling these needs through a new medium of interaction- via online methods-, the ways in which people interact with others exponentially increases.[12] People who have traditionally found it difficult to converse in face-to-face conversations, whether it be due to an uncomfortable social environment or simply a low need for inclusion, can now satisfy these inclusion needs through online chats and blogging platforms without the stress of reading nonverbal cues or paying close attention to social conventions. People who are considered extroverted also benefit; now, they can chat with friends through texting protocols on cell phones and remain updated on their friends' activities by watching them

through various social networking sites.

As new generations continue to grow up with these new technologies, the ease and comfortability of using these technologies will continue to rise. In fact, people are becoming so comfortable in incorporating social networking into their everyday activities that the medical field is seriously considering developing tools to incorporate into medical practice that will take advantage of this new method of communication.[13] Interacting with a search agent designed to aid them without interrupting the flow of communication would be an extension of these activities and therefore fairly easy to incorporate into an already-established network of contacts and feeds. Before a search agent can seamlessly interact with a user, however, the nature of the relationship between a search agent and a user, the environment within which the search takes place, and each participant's goals must be established.

The nature and compatibility of relationships between people are fairly dependent on how these needs are met. Symmetrical relationships are those between participants that exhibit the same behaviors toward each other, while a complimentary relationship are based on mutual exchanges of complementary behavior.[14] In the case of the relationship between the search agent and the user, a complementary relationship is formed. The user is lacking information and requires new knowledge, while the search agent supplies it to the best of its ability. This relationship is understood by the user, as it understands the search agent's role as a tool.

This relationship can be further analyzed by establishing the activities that each participant takes to fulfill this role. The user turns to Mordinn when they need to access new information. The user needs answers that meet their requirements, be provided with information that is related to their inquiry, and be able to form new knowledge with that information. As a provider, Mordinn aims to provide this new knowledge as accurate and satisfying to the user as possible. In order to do this, Mordinn must be able to receive the user's question, break down the language into usable pieces, and present data that is pleasing to the user. Each participant will react according to whether this information presented meets or violates expectations.[15] The user will not generally interact with this program when needing someone to talk to, for example, and the program will

not even run unless the user activates it. This suggests that a particular type of communication protocol is needed to accomodate these interactions, and that the user understands this relationship inherently.

Having established the framework in which the user and the program would generally interact, face-to-face communication strategies can now be analyzed to prepare Mordinn for proper interaction with the user.

2.2 Environment

When judging appropriate interpersonal interactions, the environment within which the conversation occurs should be considered. According to Gosling et al.[16], people respond to cues within the physical environment and incorporate them when people form impressions of others. Conversations taking place in a quiet study room in a library or within a corner at an active coffee shop will have two completely different effects on the interaction. This is due to several parameters that come into play when people react to environmental stimuli. Concepts of formality, warmth, and privacy, for example, affect how well one can relax in an environment and what topics may be considered safe to discuss openly.

Interactions between the search agent and its use generally will occur through a personal computer. Whether through a self-owned desktop or a public computer, there is still a semblance of privacy felt by the user as they work on the machine. Additionally, constraint perceptions can be relaxed, as the user controls when they open the program and can easily leave it if they so choose.[11] This leaves all control of the interaction to the user, allowing the user to feel more at ease. Additionally, computing platforms share similar characteristics between models. All computers utilize keyboards, monitors and computing mice to aid the user in using the machine. Also, the software that the computer runs on is generally one that's familiar to the user already. People generally prefer familiar places[17], and this can be applied to computers as well. For example, users of Windows will be more comfortable using different distributions of the Microsoft operating system and will prefer to use computers with that type of operating system then, say, a Macintosh computer. Therefore, the user will already be operating a computing environment that is predictable and familiar, thereby reducing

anxiety levels.

2.3 Interaction Limitations

Interactions between people are governed by communicative norms already established in society.[11] Starting from youth, people are taught how to greet people, how to manage conversation flow, and what to consider acceptable and unacceptable social behavior. These rules of engagement are developed and maintained in a variety of ways[18]: by conscious discussion and mutual agreement, through direct instruction of a more experienced individual, or through subtleties governed by meta-communication[11], for example.

Oftentimes, people are not as aware of communication norms until they are violated.[19] Different types of violations prompt different reactions, which can range anywhere from slight annoyance to great offense. To offset these violations, apologies and other methods of diffusing negative reactions are used, such as apologies and disclaiming in advance that a violation is about to occur. In the case of Mordinn, it will not know if the information presented to the user will be completely accurate and, therefore, will not know in advance if that will violate a user's expectations. To account for this, Mordinn will employ apologies[20] and use a credentialing strategy[21] to soften any frustration resulting from a result deemed unsatisfactory by the user.

There are also prescribed methods of starting an interaction with another person. These steps are often executed without consciously being aware of every step of interaction. However, programs only understand the code written by its creator. They are often not bound by conversation protocols because they are treated as tools. Additionally, a computer is not normally equipped with sensors that may be able to detect when a user wants to engage in a particular activity. Coupled with the inability to read non-textual cues, Mordinn is already at a great disadvantage in comparison to another human being.

To minimize these disadvantages, human-inspired responses and interaction strategies should be used as much as possible. In the case of Mordinn, the greeting is an especially important element to incorporate when designing it. How one greets another is a direct indication of what sort of relationship is established.

Traditionally, verbal responses are accompanied by physical cues such as a smile, a nod of the head, or handshakes. However, Mordinn can only understand input typed into the computer interface. A hand wave or other body language may be replaced by the user clicking on an icon programmed to execute Mordinn's file or by a keyboard shortcut. Approaching Mordinn would require manipulating hardware. This interaction is always initiated by the user; therefore, in keeping with human engagement protocol, Mordinn should greet the user once opened before asking if the user needs anything. Knapp and Vangelisti[11] categorizes these response types as verbal salutes and personal inquiry questions. By employing these strategies, the user will, hopefully, become more comfortable in engaging with Mordinn.

2.4 Linguistic Theory

2.4.1 Chomskian Principles

Machine comprehension of Language itself is considered an AI-complete issue, meaning that in order to solve one problem, one must also solve many other problems simultaneously. Fortunately, access to lexicon such as the Wordnet lexicon[22] and other linguistic components frees development time for this project and enables it to build upon their previous developments.

In order for a machine to understand and use Language, one must look at the source of Language: humans. The language phenomenon is a uniquely human ability, with a complex network of different areas of the brain working together to interpret, translate, and respond to the outside world. Theoretically, how Language is stored and used is controversial.

For this project, Chomsky-influenced principles are the assumed framework. This should benefit the development of the machine's natural language understanding and processing in several ways. According to his theories, all languages share some common rules, with some rules of grammar hardwired into the human brain.[23] Additionally, these common rules can be switched on or off, according to the Principles and Parameters theory.[24] By having a predetermined set of variables, a machine should be able to understand its input more efficiently and,

with the aid of propositional logic, should be able to understand relationships and begin to development knowledge independently of direct human input. For the scope of this project, developing tools that take into account the characteristics of English is the primary focus.

2.4.2 X-Bar Theory

To account for this, the construction of a new grammar that implemented the Principle of Modification [25] was attempted. The Principle of Modification states that if some phrase YP modifies some head X, then the YP must be a sister to X or is a projection of X. According to Travis 1984 [26], word order falls under the rules of predetermined parameters. For this project, we shall look at how specifiers, adjuncts, and complements are organized in English.

In layman's terms, X-bar parameters can be likened to a switch box. In the cases of specifiers, adjuncts, and complements, the position of the phrase head or the node representing the phrase head can come before or after the modifying phrase. In English, these parameters are realized as follows:

(a) Specifier: XP $\rightarrow (YP)X'$

(b) Adjunct: X$'$ $\rightarrow X(ZP)$ *or* $X' \rightarrow (ZP)X'$

(c) Complement: X$'$ $\rightarrow X(WP)$

In knowing where the head of the phrase should be within a larger phrase, we can see how different phrases are ranked in importance in comparison to the phrase head. For example, let's examine the noun phrase 'The large box of books from the bookstore' in Figure 2.1:

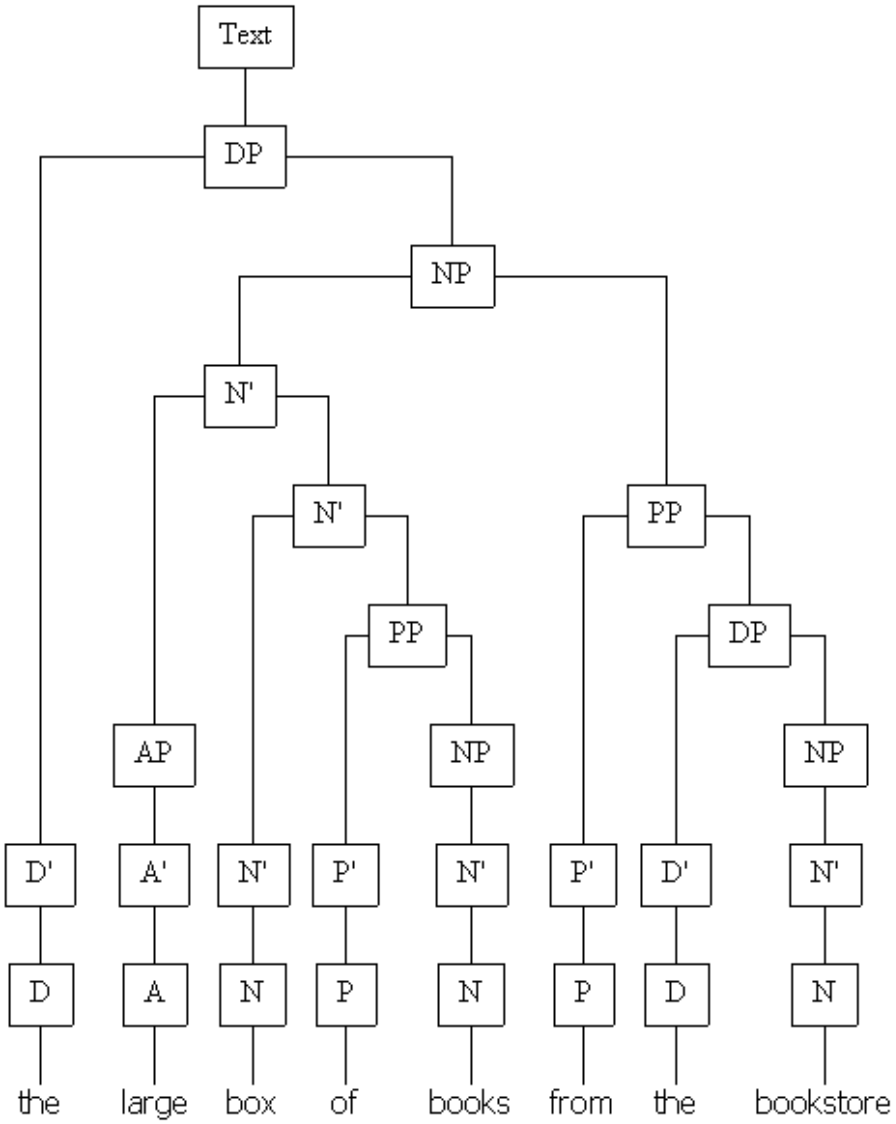

Figure 2.1: An X-bar tree of the noun phrase 'the large box of books from the bookstore'

As seen in its sentence tree, the prepositional phrase 'of books' is the complement, with the additional phrases as specifiers. By being able to recognize these traits, one can see that emphasis is placed upon the contents of the box, rather than the location where the contents of the box came from. As specifier traits are attached to the head node, the consequent information becomes less important. For English speakers, this may have origins in the fact that important pieces of information need to come first during communication, with extraneous information following. Having been given the key pieces of information at the beginning, English speakers may lose track of extra information as the sentence progresses.

Reconstructing the grammar to reflect these trends should, theoretically, allow for the use of phrase heads and recognize sister and daughter phrases. With the implementation of the Wordnet tagger, descriptors and the phrases modifying the noun head would be identified quicker by bypassing complicated rules and allowing the program to focus on generating relational hierarchies. This step will be key when constructing a system that would allow the computer to apply semantic relationships between constituents of a sentence.

2.4.3 Theta Criterion

Language is composed of two main components: a computation component, and a lexical component. The computational component of Language stores the grammatical rules and constraints of a language as well as constructs sentences while filtering out ungrammatical input. The lexical portion of Language acts as a mental dictionary, where words and their properties are stored. The relationships between members of a sentence are established through the use of the Theta Criterion[27].

The Theta Criterion matches the types of arguments of a sentence to a theta grid. This relationship is established through the use of a theta role, which is the bundle of morphosyntactic relationships that are bound to an argument. These roles can be external, where the role is assigned to the subject, or can be internal, where roles are assigned to direct and/or indirect objects. According to the Theta Criterion, each argument can only be assigned to one theta role, and each theta

role can only be assigned to one argument.

Additionally, each lexical entry must contain within itself at least the following information:

1. the meaning of the word
2. the syntactic category of the word
3. pronunciation
4. exceptional information (for example, morphological irregularities)
5. theta grid (argument structure)

For this project, points two and five are especially important. Point two is important because the machine will be able to parse and extract information properly once the parts of speech are established. Additionally, point five is important because knowing the different types of argument structures will allow us to predict what types of input scenarios the system will encounter. For this project, points one, three, and four aren't as important because:

(a) meaning of the word will be left to the user,

(b) the input is currently text only, currently rendering phonetic information unnecessary, and

(c) as input is assumed to be grammatical, the user will already account for irregularities.

Point one will, with future development of the program, become a more important piece of the code; however, establishing a word's synsets, lemmas, and hypernyms and their relationship with these qualities of other words in conjunction with propositional logic are outside the scope of this current project and will be addressed in the next phase of development.

Yet another useful characteristic of the theta criterion is that the roles are assigned according to the node that governs it. According to Koopman et al 1991's VP-internal subject hypothesis, if it is assumed that subjects are assigned

by verbs, then it can also be said that theta roles are assigned entirely within the verb phrase.[28] This makes determining where arguments are placed within the use cases much easier, as the interrogative statements often fill an argument role and are immediately followed by a verb. For example, as seen in use case 1.10, the subject is satisfied by 'what', the verb 'is' immediately follows it, and the secondary argument 'invention' is found inside the predicate itself. This pattern was a common pattern across the different use cases used.

Chapter 3

Methodology

3.1 Tools

Before the program can be constructed, several variables must be considered. A major player in developing a more user-intuitive search program is the programming language with which the program will be written with. There are many different types of programming languages that can be used, with each language structuring rules differently and using different libraries. Currently, programming languages like C and Java are very popular amongst computer scientists. For this program, however, the programming language Python will be implemented. Python is an open-source language that has been developed for more scientific applications.[29] Unlike Java's case, many commonly-used definitions are already defined and stored within its libraries, whereas in Java's case everything must be developed from scratch. Python is also widely used within academics, scientific research, and other programming projects. With an active online community and a wealth of support, information, and tools provided under flexible creative commons licenses, Python is the best choice in developing this new tool.

Another benefit to using Python for this project is that it hosts the Natural Language Toolkit.[30] This series of libraries is developed specifically for work within the Natural Language Processing field. Many different types of corpora are already included within the library itself, making it useful when testing the many different functions written into the Python depositories. Additionally, it provides

classes for commonly-needed functions like parts-of-speech tagging, parsing functions, and chunking strings. As this project is centered on the use of language and creating a program that can manipulate user input more efficiently, these two tools are indispensable.

Having selected Python and NLTK as my main tools, the next step was to establish exactly what was needed from this program. By establishing the output that is needed, one can then determine the steps that need to be reached to get there. For this project, the program needed to return results that matched the type of information the user was asking about. This requires the machine to be able to understand user inquiry in terms of parts of speech, the relations between the language, and the context that the question is asked within.

3.2 Use Cases

To start with, input was restricted to interrogative statements. Though people do not always follow standard structure when asking for information, the machine needs to be able to parse standard question form before it can be modified to accept a greater number of variables. Basic interrogative statements with either one or two noun phrases were first used to determine whether the program could successfully navigate through a sentence. The first batch of questions used the interrogative 'what'. While the English language has six major wh- question words in use (there are more, but are not used in as much frequency as these), each have specific contexts in which they are used and are involved with different types of morphological categories. To start, 'what' was the first type of question to be explored. 'What' allows for simple questions with correct and incorrect answers. By testing the code with these use cases, basic parsing tools can be developed.

As seen in Appendix 1, simple interrogative statements like 'What is a pen?' were used. By using simple interrogative statements, the user can check for accuracy and recognize errors in the code that need to be fixed. When run through the terminal in Python, one can also call for information, create new definitions, and interact with the code live. In fact, much of the testing stage was done through the live use of a terminal due to the immediacy in determining

errors and in allowing to promptly test solutions.

Chapter 4

Experiment

4.1 Tagging

By chance, the first test case 'What is a pen?' was tagged inappropriately. When using the embedded tagging system, 'pen' was marked as an adjective. To determine accuracy, changes in plurality and phrases were made. When testing 'What are pens?' and 'What does a pen do?', 'pens' and 'pen' were correctly tagged as nouns. To determine where the error occurred, the source code of the tagger was examined.

A large portion of the tagging protocol ran the Punkt tokenizing system, which was developed using the methods of recognizing abbrevations in Kiss and Strunk 2006's 'Unsupervised Multilingual Sentence Boundary Detection.'[31] Kiss and Strunk define an abbreviation as a 'very strict collocation', and can be characterized by three properties:

1. The abbreviation and the period attached to it share a close bond,
2. Abbreviations have a tendency to be short, with the liklihood of being an abbreviation decreasing as the length of the abbreviation increases, and
3. Word-internal periods are often found within many abbreviations.

This was found to be 99.38% accurate when tested across newspaper corpora in eleven languages.

Given that 'pen' was short in length and was next to punctuation, it could have been mistagged as an abbreviation or some form of descriptor. However, as an English speaker, it can be seen that 'pen' is clearly a noun due to its position inside its phrase and the surrounding phrases. This information is tied directly into the syntactic rules of English, which the machine does not have. To help prevent mistagging and to help the machine understand a word's relation to the rest of the phrase, a chunk grammar should be implemented.

4.2 Automatic Parsing

The most widely-used chunk parser class in NLTK is the RegexpChunkParser class. This class breaks down a sentence into the phrases that compose it. Here, all phrases are marked as being on the same level as the other phrases. This disregards nesting, where some phrases only modify specific phrases. For example, when the phrase 'the large box of books from the bookstore' is parsed using this class, the output becomes as seen in Figure 4.1.

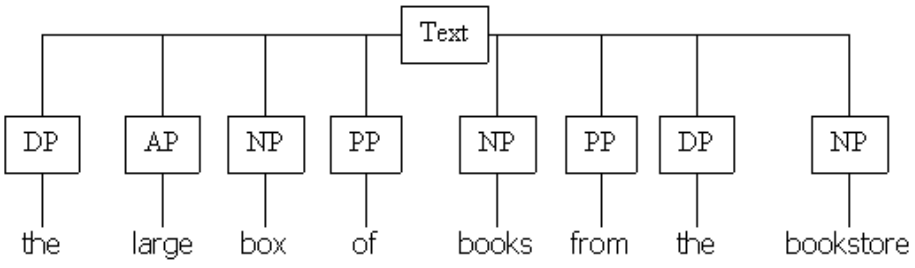

Figure 4.1: A traditional tree of the noun phrase 'the large box of books from the bookstore'.

While this method of organization may be proficient for simple phrases, it does not account for the complexities of larger phrases. For example, information from sentences like 'Yesterday, my younger sister that runs a flower shop went to the stylist's to get her hair cut.' are not organized on a flat structure. Phrases like 'younger' and 'that runs a flower shop' specifically modify the overarching

noun head 'sister' and are not considered as important to the subject as 'sister'. This is also the case when analyzing the predicate, as the verb phrase 'went to the flower shop' is more important to the grammaticality of the statement than is the reason for going, which was to get a haircut. The inability to recognize which phrases are considered more important than others hinders the program from recognizing the hierarchy of information that the user is presenting.

X-Bar theory allows for the embedded relationships between different phrases, developing a sort of hierarchy of importance when phrases are against each other. If a grammar can be successfully read by the program using this method of breaking down sentence pieces, traces and content such as tense, possessives, and plurality would be aligned in their appropriate spots. This would allow for the system to simply be rearranged when wanting to apply languages outside English. Additionally, by having a basic underlying structure established for all phrases, the process of finding the overarching phrase head and, therefore, the main object of the inquery, would be completely more efficiently.

However, this approach failed because the system could not recognize several changes. First, the program could not understand the new structure system. Instead of merely labeling the sentence as 'S', tense phrase 'TP' was tried. For theoretical reasons, this allows the system to be broken down further. However, the grammar could not recognize the name 'TP' as the starting point of the sentence and refused to compile. As a means of correcting this, the label 'S' was left and coding the head nodes as an intermediary layer was attempted. A sample of a self-contained grammar can be seen below:

grammar1 = nltk.parse_cfg("""
S $\rightarrow NPVP$
VP $\rightarrow VNP|VNPPP$
PP $\rightarrow PNP$
V $\rightarrow$ "saw"|"ate"|"$walked$"
NP $\rightarrow$ "$John$"|"$Mary$"|"Bob"$|DetN|DetNPP$
Det $\rightarrow$ "a"|"an"|"the"|"my"
N $\rightarrow$ "man"|"dog"|"cat"|"$telescope$"|"$park$"
P $\rightarrow$ "in"|"on"|"by"|"$with$"
""")

This example, given by Bird [34], is a closed system that can supplies the program with a basic grammar to parse input with. Taking the same phrase from Figure 4.2, a phrase parsed with this grammar would be separated as the following:

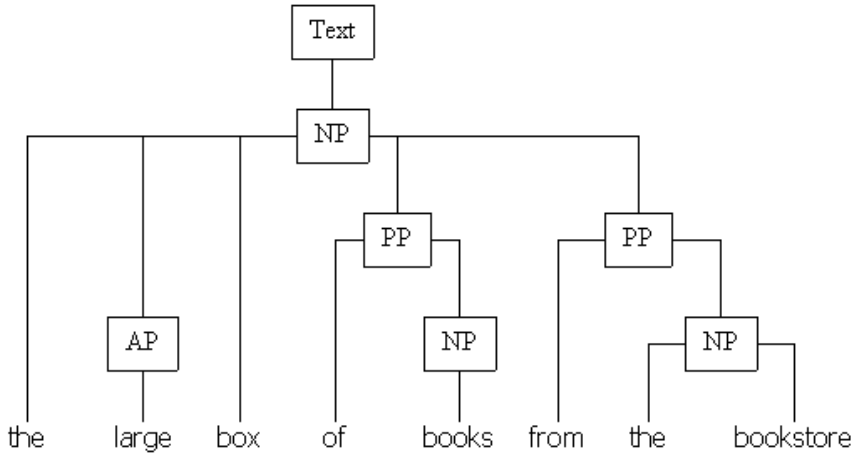

Figure 4.2: A traditional tree of the noun phrase the large box of books from the bookstore

To test the parameters of the new grammar, values were already defined within the grammar to ensure that the only variables that were being tested were the grammar rules. Here is a sample of how the grammar was edited:

grammar2 = nltk.parse_cfg(”””

CP $\rightarrow CTP|TP$

TP $\rightarrow DTTVP|DPVP$

DP $\rightarrow D1$

D1 $\rightarrow DNP$

NP $\rightarrow N1$

N1 $\rightarrow AdjPN1|N1PP|NPP|N$

)

This code attempts to restructure the traditional sentence decomposition by using X-bar theory as its basis. Head nodes are denoted with 1 instead of ' , as the computer could not recognize the use of the apostrophe. By breaking down a sentence by phrase, this should have given the machine a way to recognize phrase hierarchy. Unfortunately, this approach failed.

When trying to incorporate an intermediary layer into the grammar, the system could no longer identify what the rules wanted. One can suspect this may be due to its inability to account for a variable intermediate node as well as a system developed solely on traditional grammar parsing. Without the ability to use nodes and basic X-Bar notation, more complicated organization such as theta criterion, head movement, and theta roles cannot be implemented directly. Therefore, an alternative approach to reading input is needed.

4.3 Manual Parsing

A back-door method must be developed in order to achieve a result that behaves similar to what was originally designed. In order to do this, one must anticipate how the data will be presented and analyzed. Here, the Theta Criterion becomes more important when determining where different parts of speech are found within the sentence structure.

Currently, machines do not have the capability to process all of this information at a rate similar to a human. Calculations and judgments of this caliber would requirement an extensive knowledge network as well as constant access to a larger database that is frequently updated with new trends in language use. However, a substitute can be made. Online networks and databases can be accessed, therefore rendering these sources as part of the program's brain. By taking advantage of resources already posted online, a system can be written to access that information and return proper results, much like how people access a catalogue.

4.4 Current Code

4.4.1 Interrogative Statements

Currently, Mordinn is restricted to handling four types of interrogative statements: those involving 'who', 'what', 'where' and 'how'.

'When' and 'why' were not developed for this particular project due to the complexity of the output that the user is expecting. For example, 'why' questions are often asking for explanations. Being able to explain 'why' requires the respondent to understand the concepts being asked, the relationships involved between the different concepts presented within the question, and the reasons behind their relationships. 'Where' is also complicated in that it involves a timeline. Dates can be written either with numbers or longhand, while relational terms like 'yesterday' and 'tomorrow' are only relevant at the moment being asked. Developing a system that takes these relationships into consideration is too complicated to be completed within this particular project, and will be attempted once the code has progressed further.

4.4.2 Input

Once the program is loaded, the user is greeted by a request for input. The prompt for this input is customizable, and can say anything the programmer would like it to say. To help ease the user into becoming familiar with using this program, I have the program greeting the user with 'Hello' followed by the prompt 'What questions do you have for me today'. This makes the interaction more natural as well as acts as preps the user for a response that the machine is more capable of handling.

Once the user gives input, the data must be separated and tagged appropriately before the program can begin deciphering what kind of content the user is looking for. The input is first broken down into individual parts through a tokenize function and saved as a separate definition, allowing it to scan over the data by word. For example, if use case 1.4 was tokenized, a definition question could contain data as ['What', 'is', 'the', 'color', 'of', 'a', 'banana', '?'].

From here, the program then tags each token. The tags are based off data

from the Wordnet Corpus. Tagging these words allows for the program to then be able to analyze the data according to its part of speech. This provides a way to extract grammatical information from the word pertinent to the Theta Criterion without the aid of an independently-developed grammar.

Currently, the program is expecting the input to begin with an interrogative and begins by determining which interrogative is used. If 'Who', 'What', 'Where', and 'How' are used, then it continues forward. Otherwise, it returns a message apologizing for its inability to handle the input.

4.4.3 What

The code for the 'what' interrogative statement was first to be developed. This was because the simplest questions can be asked. Questions like 'What is a koala' and 'What color are dandelions' are asking for a definition, a characteristic of an object, or other such simple inquery. The sentence structure is also simple, allowing for a basic code to be developed for extraction.

When deciphering the input, the noun phrases must be extracted and separated. Nouns and the adjectives are important for searching for the correct answer because they provide the boundaries that the answer must lie within. As seen earlier through the discussion of X-bar theory, the head of a noun phrase has higher priority than the descriptors attached to it within its boundaries. In addition to the assumption that arguments are governed by verbs, the remaining arguments will follow after the verb phrase.

Nouns and adjectives are extracted and stored separately in their own definition. They are appended to a blank definition in the order that the program cycles through the input (which is from start to finish). Once this input is extracted, the program looks to see if the user is asking for a definition. If the user is looking for a simple definition, then the length of the definition storing the nouns and adjectives should only be one. If it is one, then the system immediately searches for a definition using a Google definition and returns several different definitions.

If there is more than one entry within the definition, then Mordinn prepares the data to be searched. The entries inside the definition are reversed, then joined by $\acute{+}'$ and placed into a new definition. The entries are reversed in order to rank

the words from largest category to the smallest category. When communicating in English, the most important pieces of information come first, with the information following it decreasing in importance. By reversing it, the search terms are then reordered, allowing the search to look for the most specific information as possible. This allows Mordinn to work from a rudimentary skeleton that attempts to emulate human behavior.

The terms are then entered into Google search. Before the program returns information, it first searches for a Wikipedia entry. Wikipedia has quickly become a first source for people to become more familiar with unknown topics. Since Google returns search results according to how often people visit sites with those search terms, the likelihood that a Wikipedia entry regarding the topic asked about will be returned is high. This search specific can be tailored to what the user would like it for, and rank search results from another source higher than others. If a Wikipedia result is not found, then the first link is designated as the main link to open.

4.4.4 How

Parsing 'how' interrogative statements was done similarly with 'what' statements except that a) Ehow.com was the main database used for the search, and b) entire verb phrases were used instead of only adjectives and nouns. When asking 'how' questions, one is generally trying to learn the means of accomplishing some task or come to an understanding of the mechanisms behind the behavior of others. By using Ehow.com as the base for the search, the odds of finding instructions on how to carry out an activity are raised considerably. An interesting challenge is raised, however, when trying to search for the proper links to return.

Unlike the instructions written for 'what' interrogative statements, where a library was imported to assist in locating the search results within the html, the code must navigate through the raw source code of the search results and extract the links directly. This is generally very troublesome, as not only html code not uniformly written across webpages, but there are also embedded links that contain urls for advertisements, links to other sections of the host website, and images. In order to determine if there was a trend in where the host site

places the desired links, the location of the links from the actual results had to be determined. When comparing the results for five different use cases, a trend was discovered where the actual search results began at the 21st link. From this, the code had a place to start its extraction of urls.

4.4.5 Where

Deconstructing 'where' questions became more complicated. Unlike 'what' questions, 'where' questions may have only one noun phrase that may or may not be enough information to return an adequate response. Additionally, a user may ask 'Where is the library' without telling the program where the user is situated in real life or where they would like the search to be confined. Before anything can be searched, the program needs to be able to know if there is enough data presented to look and, if not, ask the user for more information before searching.

The code goes through a series of loops. The first round of looping separates the tokenized input by its part-of-speech. This initial loop is conducted in order to determine what type of information the user gave the program. Its main goal is to pinpoint any prepositions within the input. Prepositions like 'in', 'at', 'on', and 'under' designate location. These phrases most often are tied with an argument and separate the argument from other possibilities with similar characteristics. For example, when considering two balls- 'the ball under the table' and 'the ball inside the box'- the receiver understands that the difference between the two is their location. It can be said that prepositional phrases help place restrictions on what is true about the argument by distinguishing it from others with similar qualities, and therefore can also be seen as modifiers of arguments.

When conversing with another person, there is, oftentimes, understood information that neither person need to outwardly confirm. If two friends meet on a school campus and one asks 'Where's the library?', the listener makes the assumption that their friend is referring to the school library and continues conversation with that assumption. A program, however, does not have access to that background information without having it physically stored into its program.

To account for this, when separating the input's parts of speech, it is also looking for indicators of vague questioning. Vague questions such as those demon-

strated by the previous example are often short, using a single noun phrase. After the input is separated, the program looks for indicators of this type of question. If the second word within the program is 'is', 'are', or 'was', it then looks for for any prepositions. If there are prepositions found after this verb ('are' and 'was' are the plural and past tense forms of 'is', respectively), then the computer moves forward, assuming that the information tied to the prepositions will make the search query more specific. In the case that prepositions are not present, it then looks to see if there are any proper nouns. Proper nouns like Tucson, Gothenburg, and even Sweden are the names for specific locations, and if they are present the program assumes the search query is specific enough.

In the case that there are no proper names, the program interacts with the user and, once informing them that the question is a little vague for it, asks for a more specific location. This interaction enables the program to obtain the necessary information it needs to provide an accurate answer with as little interruption as possible. Assuming that the user cooperates with the request, the program will then append the new user input to a raw list of search terms and incorporate it into its search for the appropriate answer.

For this particular search, certain types of results are ranked higher. The highest ranked type of result is one that from maps.google.com, followed by tripadvisor.com, then wikipedia.org. These three sites were specially as they provide maps and other means of designating the location of their inquery. Tripadvisor may seem like an odd site to include; however, when asking where the best burger in Tucson is, Tripadvisor.com returns results that users themselves have given. Therefore, not only is the user given a map or directions, but there is also additional user feedback attached to the sought-after information. Depending on the search results, one of these will, most often, be chosen as the designated link to be returned.

4.4.6 Who

When creating the code for 'who' questions, three types of inquiries were considered: questions about a public figure, questions about a private figure, and questions about the actions of a person. Each of these questions would use dif-

ference sources for information, especially when searching a public figure versus a private figure. The primary difference here, then, is to determine whether a) the user is asking about a person specifically, and b) if this person is a public figure or a private individual.

If asking about a specific person, that person in question is generally listed first within the predicate. Questions like 'Who is Naoto Kan?' and 'Who is Prince William married to?' all refer to the person immediately after the verb. People's names are considered proper nouns, and by determining whether the word immediately following the initial verb is a proper noun, the system can then ask the user if the person in question is a public figure or a private individual. If the person is, indeed, a public figure, Wikipedia.org entries are listed as the favored search result, as oftentimes the Wikipedia entry for public individuals is found within the first several urls.

If the person is a private individual, however, the results favor results from different social media sites. Given the current abilities to access social media sites from cellphones and smart-phones, along with many sites incorporating other social media sites within their own, private users are more likely to use these services to connect to others. Currently, five sites are ranked from most favored to less favored: Facebook.com, Linkedin.com, Twitter.com, Tumblr.com, and Myspace.com. These sites are currently very popular in terms of social media.[32] By favoring these sites, users can also search for screen names, when the person's real name is not known and still return fairly accurate results.

In the case that the user is inquiring after the actions or status of a person, the Wikipedia.org results are also preferred over other results. This, predictably, works better for public figures than for individuals. Currently in development is the Friend of a Friend network.[33] Also known as FOAF, this network works with the development of the semantic by establishing networks of relationships between people. While it is unknown whether this will become a very popular option, it would be an interesting addition in the future.

4.4.7 Output

Once found, Mordinn will open up the designated link immediately through the user's default browser while providing a list of up to nine alternative links. While the computer is opening the main link, the program presents alternative links and informs the user that some of them may also help in case the original link is not a completely accurate return. The designated link is opened through the browser automatically to decrease the amount of time and labor that the user must commit before finding the answer that is needed. Oftentimes, internet users start from the first link and progress forward as their questions remain unanswered. The intermediary steps of visiting a search engine, manually parsing through results, and then opening a link are done by the machine instead. The user, then, only needs to open additional links if the user is unsatisfied with the initial response.

Chapter 5

Results

5.1 Setting

To test the accuracy of the search agent, the use cases where run live in the program as well as entered into Wolfram—Alpha. Wolfram—Alpha was chosen as the competing search engine because of its designed. Based off the commercial software Mathematica, Wolfram—Alpha steps away from semantic search and searches for results computationally. By comparing results against Wolfram—Alpha, we can also if different kinds of results are favored in either engine as well as determine whether linguistic information is important when determining a result.

As seen in Appendix A, there are 40 use cases total that were tested. Each use case was tested in Mordin during the development stage to ensure proper parsing and information extraction. There are ten different questions for each type of question. When entered into the engines, the results were given one of three judgements: 0 for inaccurate, 1 for insufficient, and 2 for accurate. Inaccurate answers were returns that did not answer the question at all or were completely off-topic. Insufficient answers were those that either had the answer on the first page indirectly or had better returns in alternative links. Accurate answers were returns that gave the information the user was requesting. To account for possible ambiguous searches, where the answer could be judged either accurate or inaccurate depending on the type of information wanted, the answer

was judged inaccurate. Judgements were also made on a more reserved basis to try and account for possible interpretations from other users.

5.2 What

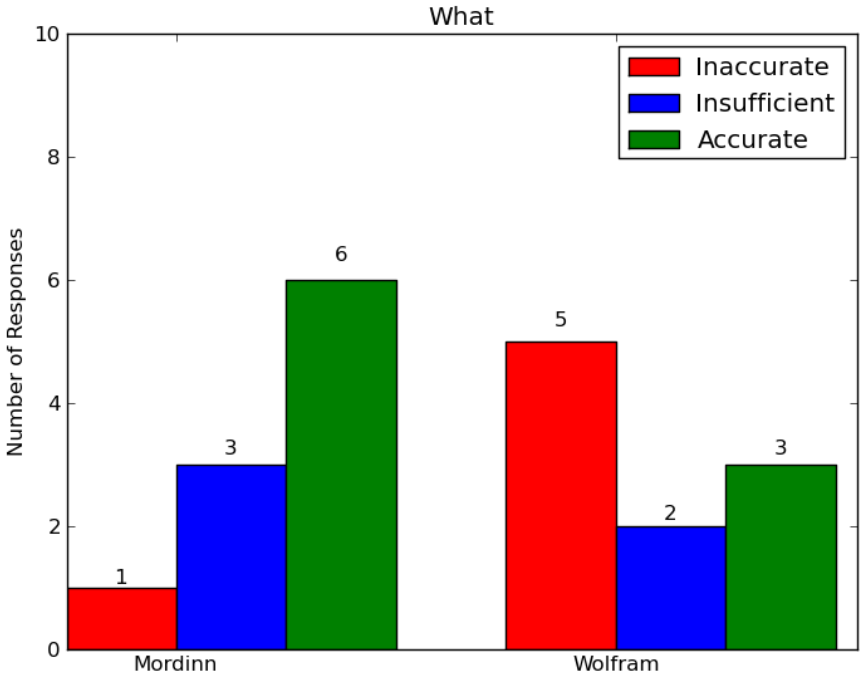

Figure 5.1: Accuracy of What-initial user input.

Figure 5.1 demonstrates that in the case for 'What'-initial questions, Mordinn has a higher rate of accurate returns than Wolfram-Alpha of at least 30%. Wolfram performed better when statistical data was the preferred response, whereas Mordinn performed better at returning information concerning definitions or relationships between arguments.

Although Wolfram-Alpha has a higher rate of inaccurate responses, its responses to questions such as use case 1.05 are much more accurate when the user needs data for statistical reasons. If the user's purpose of asking about gas prices

in a specific area was to plan where to replenish their vehicle's gasoline tank at the minimum cost that day or week, then the results generated by Mordinn are much more suited. However, if the user wanted to determine average gas prices over a longer period of time, the graphs and numerical data provided by Wolfram are much more relevant. A further exploration could be done, where search queries are reworded, thereby slightly changing the semantic meaning of the question without changing the overall content. Similarly, when searching for information pertaining to use case 1.07, Mordinn's results return a more detailed account of John F. Kennedy's assassination, while Wolfram returned the exact date and additional calculated days since the event.

When searching for information that relied on relationships between the different arguments of the inquiry, Mordinn returned better results than Wolfram. Use case 1.08 is a good example of this. Wolfram completely failed to return any sort of information in regards to what a microscope or what microscopy is. Instead, it returned technical information about the mathematical symbol 'difference between', a topic which is completely irrelevant to the question. Mordinn had a more difficult time returning correct data immediately; however, a description of microscopy with an embedded link to microscopes was the first result given to the user. This sort of question is a good example of the current incapabilities of natural language processing, as programs are unable to 'learn' from raw texts.

Both failed when specialized data was requested. Use case 1.06 was the only case where both search engines failed to return accurate answers. The term 'magic number' was searched, anticipating its definition according to the chemical application. However, Mordinn returned the definition as used in programming fields, while Wolfram returned baseball statistics. This demonstrated that the program was lacking background, or contextual, information that the user was employing nonverbally. Had the user asked this of a chemistry professor, the professor would have understood that the asking person wanted the chemistry definition. It was rather surprising that Wolfram did not return the chemistry definition, as Wolfram is specially useful when making calculations or when references data for mathematical or scientific applications. Further tests asking for specialized definitions may be done in the future to help develop a better parameter for contextual information.

5.3 How

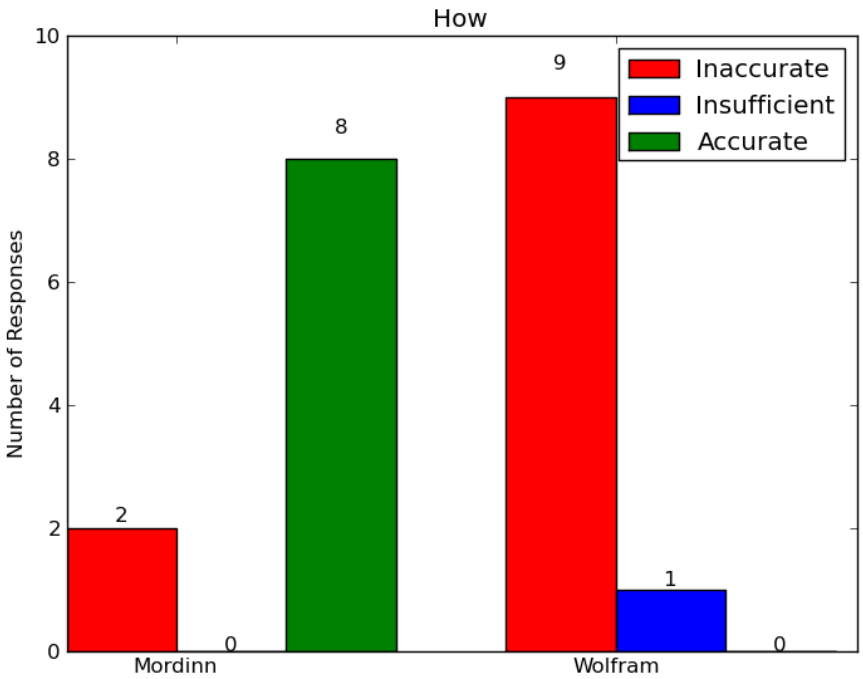

Figure 5.2: Accuracy of How-initial user input.

Figure 5.2 demonstrates that in the case for 'How'-initial questions, Mordinn is significantly better at returning information than Wolfram-Alpha.

'How' questions specifically rely on the relationships between the arguments. As seen in most of the use cases, the user is asking for instructions or ways to carry out an action. In essence, this could be narrowed down to a basic pattern of: 'How can I perform X in Y?'. To be fair, Mordinn was specifically set up to run in an engine that handles these types of requests, and therefore was expected to pass. However, Wolfram claims that Wolfram- Alpha can arrive at the same results while approaching the search computationally.[9] With the exception of use case 2.07, where a numerical was needed, Wolfram continually returned incorrect and often unrelated results. For example, use case 2.02 asked for a method of opening a can; instead, Wolfram returned data pertaining to the Andean Community of

Nations.

An exception to this would be when a numerical was sought after, as in the case of 2.07. Mordinn is currently unable to handle such a request and returned an error. Wolfram, instead, answered the question indirectly. Wolfram provided a large taxonomy of a domestic dog and, after expanding the taxonomy tables several times, an answer was given indirectly through the use of an image. Interestingly, Wolfram gave information about the volume of eyes and other specific details but could not simply list that a domestic dog has two eyes. Obtaining this answer required being able to see images as well, presenting an additional issue to consider when developing Mordinn further.

5.4 Where

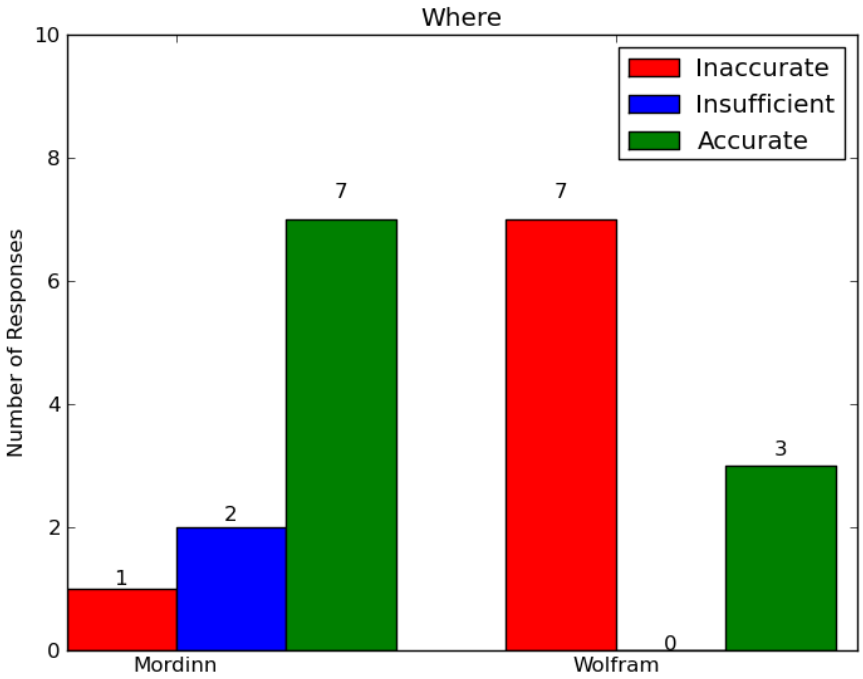

Figure 5.3: Accuracy of Where-initial user input.

Figure 5.3 demonstrates that in the case of 'Where'-initial questions, Mordinn is better equipped to handle location questions when the user is looking for personalized responses, while Wolfram failed those inquiries. When asking about famous or large-scale locations, Wolfram was completely accurate and returned place name and latitudinal and longitudinal coordinates to the location in question.

Use cases like 3.01, 3.03, and 3.04 were tested specifically to test for contextual clues. These examples asked for information that required a search restriction before returning results. Mordinn recognized that these use cases did not have enough specific information for it to search properly, and initiated an interaction with the user, requesting more specific information. Assuming that the user cooperates and enters relevant data, Mordinn incorporated the new parameter and returned results within that search limit. For each case that needed to be specified, the city Tucson was used due to first-hand experience of the area. These results all returned accurate locations and often included maps of the requested locations. Wolfram completely ignored linguistic data encoded in the terms and returned results that were completely off-topic. This demonstrated that lexical information does play a part in disambiguating content and in determining what the user is looking for.

When asking for larger-scale or more famous locations, such as Sweden's capital or the host for Woodstock, Wolfram gave very accurate results. When searching for the location of the Swedish capital, Wolfram returned the city name and its exact latitudinal and longitudinal coordinates. Mordinn returned results that provided more information about the capital as well as the coordinates. For users that only need a specific coordinate or name of a location, Wolfram currently stands as the better engine. In cases where extended information is wanted, Mordinn returns better results.

5.5 Who

Figure 5.4 demonstrates that in the case of 'Who'-initial questions, both cases were able to return accurate results at least half of the time.

In regards to asking about individuals, Mordinn returned correct information

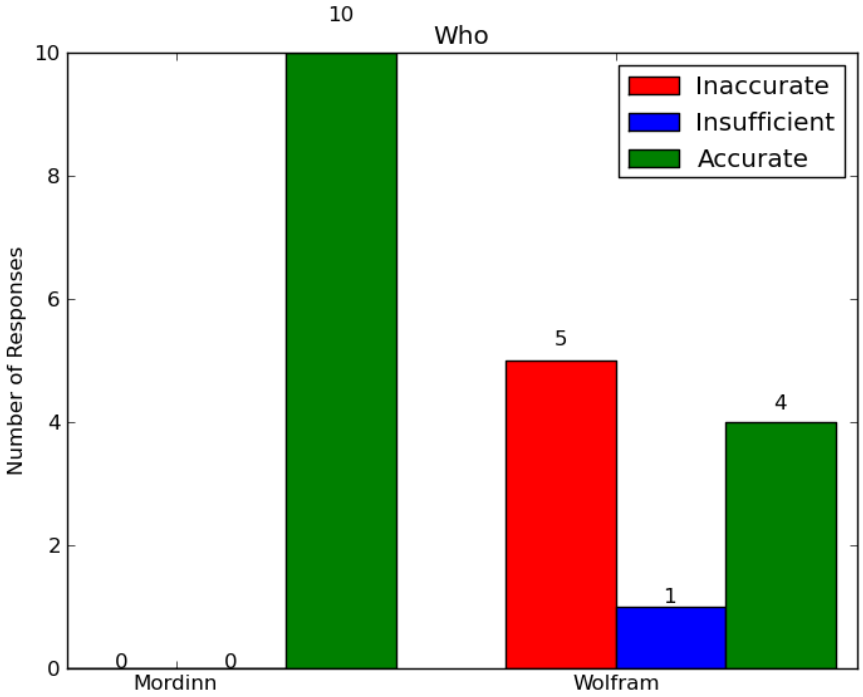

Figure 5.4: Accuracy of Who-initial user input.

more often than Wolfram. In cases where public individuals were involved, both engines returned correct results. Mordinn, however, provided more detailed information about the individual than Wolfram. In case 4.01, though, Wolfram returned a result defining 'filming' instead of any information in regards to the film. Wolfram's database was not reliable when testing current figures, though it was surprising to see it return partially correct information when asked about the death of Dumbledore, a fictional character from a globally-known novel series.

Mordinn outperformed Wolfram when searches for private individuals were asked about. Mordinn took advantage of the popularity of several social networking websites and favored those results over others. Wolfram, on the other hand, does not access those sorts of sites for data. This function becomes useful when a user needs to look up a person by their user name, not a proper name. However, the spelling correction written within the Google search engine can affect the return of appropriate sites.

Neither engine can currently account for persons that may share the same name. For instances like this, a network like Friend of a Friend[34] would become particularly handy, as it would take advantage of semantic web protocol and write in the relationships between people. Unfortunately, this is currently unusable due to it still being in the development stage.

5.6 Total

In total Figure 5.5 , Mordinn gave an average accurate return of 77.5%, while Wolfram-Alpha gave an average accurate return of 25%. Wolfram-Alpha gave a larger percentage of inaccurate results, 45%, while Mordinn returned an average of 10%.

During the search phase, Wolfram's correct results favored results that involved statistical and mathematical information. Cases such as 2.07 and 4.03 demonstrated that when a taxonomy of information is required, Wolfram returns accurate and detailed results. In cases where answers needed to incorporate relationships between arguments or required linguistic knowledge of the input, Wolfram failed.

In the other hand, Mordinn returned better results when search inquiries

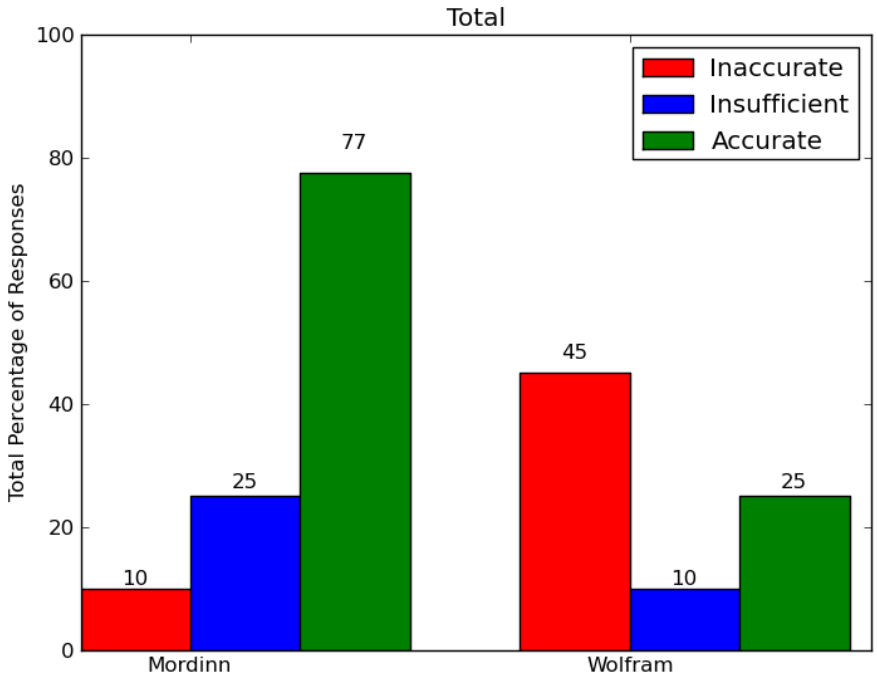

Figure 5.5: Accuracy of Total-initial user input.

relied on relational and contextual background. Mordinn's ability to initial communications with the user and ask for more specific information not only helps increase Mordinn's performance, but also allows for the user to remain involved in the search process while letting Mordinn do the search work.

Chapter 6

Discussion

The errors and inaccurate results returned by both Mordinn and Wolfram-Alpha gave good insight into how language plays a part in the communication process. Errors involving context, relations between arguments, and dealing with 'fluid' data- data that can be considered opinion pieces or are subjected to change quickly- were particularly insightful.

Context played a large role in returning accurate results. Inaccurate results, as demonstrated through use case 1.06, demonstrated the need for more context. Currently, machines do not understand when a word has several meanings, and without additional input from the user, the program has a lower probability of returning the needed answer. This phenomenon was explored through the 'Where' cases, where Mordinn searched through the entire phrase looking for additional information before asking for more specific locations. This function helped Mordinn function better against Wolfram-Alpha in this type of question.

However, the contextual knowledge that humans often draw upon is from the surrounding environments and from established relationships between other people. Machines currently do not have the capability to read from their environments and storage of previous searches and interactions is often limited. An alternative means of handling various forms of ambiguity must be developed. Further development of this aspect of the program can set out to minimize lexical ambiguity by exploring a word's relationship to its lemma, hypernyms, and synsets as well as how relationships between groups of these characteristics are formed and maintained.

Argument relations also proved to be an important player in generating meaning from a question. As seen in most of the 'How' use cases, Wolfram-Alpha was unable to access lexical information that humans use with ease on a daily basis. This could have led to its inability to reliably return related information, as seen with case 2.02 where the question about can-opening returned information about the Andean Community of Nations. Taking advantage of user-created content and databases, rather than official compilations of technical data, can help the search task when relationships and instructions are called upon.

Additionally, incorporating argument relations will help prevent the program from making assumptions. As seen with use case 2.03 Wolfram-Alpha assumed that 'journal' was used as a unit of measurement and returned information about its use as a measurement. If linguistic context had been considered, it would have recognized the adjective modifying 'journal' as a noun, as well as recognized that the noun phrase was governed by a verb head. Mordinn attempts to account for this context; however, more complex logics need to be incorporated in order to truly begin to function on a linguistic level. Future development for Mordinn could include moving away from specialized data sources and incorporate a wider range of databases.

Finally, navigating 'fluid' data is troublesome. This kind of data is constantly evolving. When trying to search for private persons, it becomes especially difficult when a) the private individual does not use a popular networking site, b) when the private individual shares a name with others, and c) when the private individual's user name is used by several different persons. However, when looking for current events or popular information, Mordinn had a fairly successful return of accurate information. Future development of private individuals will be dependent on how people continue to interact with online resources as well as how different networking sites cooperate with each other and larger search engines such as Google.

Chapter 7

Conclusion

The aim of this project was to establish a basic framework for an interactive search agent that carried out much of the search process for the user. Before the start of the construction process, goals, interaction limitations, and a linguistic theoretical approach had to be established before actual construction could begin. By analyzing how people interact with each other on a daily basis, a sense of what is unobtrusive and friendly was determined. In treating the search agent as a human with major communication blocks, such as a lack of environmental context or the limited ability to interpret input, methods of overcoming these obstacles became easier. With something as simple as a casual "hello", the search agent is able to ease the user into responding into a more human fashion. Initiating further interaction and asking for extra information in a way a human may by apologizing and asking for more specific information allows the program to obtain further data without being obtrusive.

From there, developing a basic framework became a problem. Much of the trouble in developing the framework came from applying a different theory than what a lot of current NLP applications are based off. Having failed to construct a complete grammar from scratch, an alternative method of catching parts of speech was created. Once able to pull out wanted pieces of lexical information, resorting user input into data the program can use to then search with became easier. Accessing databases online became a problem, mainly due to messy HTML tagging, thus proving that cleanly-written semantic tags will assist in the data-mining process. Once able to access online, results were then able to be returned

to the user.

Testing the use cases for actual content against a search engine that favored statistical input demonstrated that in many cases, lexical information was important when disambiguating user input. Further development in dealing with ambiguous content, handling context-bound requests, and determining argument relations in relation to overall content will help fine-tune the program. Additionally, further insight into human communication procedures can be gathered through the errors found in the testing process.

Appdx A: Use Cases

Refference Number	Question Type	Inquery
1.01	WHAT	is a pen
1.02	WHAT	are pens
1.03	WHAT	do pens do
1.04	WHAT	is the color of bananas
1.05	WHAT	is the cost of gas in Tucson
1.06	WHAT	is the magic number
1.07	WHAT	the year JFK was assassinated
1.08	WHAT	is the difference between microscope and microscopy
1.09	WHAT	instruments are used in jazz
1.10	WHAT	is the most famous invention of Thomas Edison

Table 1: Use Cases:What

Refference Number	Question Type	Inquery
2.01	HOW	do I roast a chicken
2.02	HOW	do I open a can without a can opener
2.03	HOW	do I write a reflective journal
2.04	HOW	do I change gears on a bike
2.05	HOW	can I get married in Hawaii
2.06	HOW	do I buy music from the Itunes store
2.07	HOW	many eyes does a dog have
2.08	HOW	can I increase my server speed
2.09	HOW	can I make a fire without a lighter
2.10	HOW	can I predict an earthquake

Table 2: Use Cases:How

*Name changed for publication in order to maintain privacy

Refference Number	Question Type	Inquery
3.01	WHERE	can I buy pizza [specifier: Tucson]
3.02	WHERE	is the capital of Sweden
3.03	WHERE	is the nearest library [specifier: Tucson]
3.04	WHERE	is the best burger [specifier: Tucson]
3.05	WHERE	does the president of the United States live
3.06	WHERE	was Prince William's wedding
3.07	WHERE	can I buy a portable playstation
3.08	WHERE	is Da Vinci's Mona Lisa showcased
3.09	WHERE	will the 2012 Olympics be held
3.10	WHERE	did Woodstock take place

Table 3: Use Cases:Where

Refference Number	Question Type	Inquery
4.01	WHO	is filming The Hobbit
4.02	WHO	is Naoto Kan
4.03	WHO	shot Abraham Lincoln
4.04	WHO	is albinwonderland
4.05	WHO	is Monica West*
4.06	WHO	was the 4th president of the United States
4.07	WHO	first discovered the Americas
4.08	WHO	wrote To Kill A Mockingbird
4.09	WHO	killed Dumbledore
4.10	WHO	is the richest celebrity in 2011

Table 4: Use Cases:Who

Bibliography

[1] Berners-Lee, Tim, *Mobile Web*. 3GSM Barcelona, 22nd February 2007.

[2] Stefano Mazzocchi, "Toward the semantic web.". *http://www.betaversion.org/ stefano/*

[3] "SIMILE Project." SIMILE Project. Web. 1 Apr. 2011. *http://simile.mit.edu/*

[4] John G. Breslin, Uldis Bojrs, Alexandre Passant, Sergio Fernndez, Stefan Decker. "SIOC: Content Exchange and Semantic Interoperability Between Social Networks. W3C Workshop on the Future of Social Networking". 15-16 January 2009, Barcelona, Spain.

[5] "NextBio." NextBio. Web. 13 Apr. 2011. *http://www.nextbio.com/b/nextbio.nb*

[6] Robert F. Simmons, "Natural Language Question-Answering Systems: 1969". Communications of the ACM 13:1, 15-30

[7] "Dbpedia." Dbpedia. Web 24th Apr. 2011. *http://dbpedia.org/About.*

[8] "About Wolfram—Alpha: Making the World's Knowledge Computable." Wolfram—Alpha: Computational Knowledge Engine. Web. 1 May 2011. *http://www.wolframalpha.com/about.html.*

[9] Stephen Wolfram. *A new kind of science*. Champaign, IL: Wolfram Media, 2002.

[10] Luiz Andre Barroso, Jeffrey Dean, Urez Holzle "Web Search for a Planet: The Google Cluster Architecture". IEEE Computer Society 13:1, 15-30

[11] Knapp, Mark L.. *Interpersonal communication and human relationships.* Boston: Allyn and Bacon. 6th edition. 2009

[12] Kaveri subrahmanyam, Stephanie M. Reich, Natalia Waechter, Guadalupe Espinoza "Online and offline social networks: Use of social networking sites by emerging adults". Journal of Applied Developmental Psychology 29 (2008), 420-433

[13] Carleen Hawn, "Take Two Aspirin and Tweet Me In The Morning: How Twitter, Facebook, and Other Social Media Are Reshaping Health Care". Health Affairs 28:2 (2009): 361-368

[14] D.C. Dryer and L.M. Horowitz. "When Do Opposites Attract?: Interpersonal Complementary Versus Similarity". Journal of Personality and Social Psychology 72 (1997): 592-603

[15] H. Garfinkel. "Studies of the Routine Groundes of Everyday Activities". Social Problems (1964) 11:225-250

[16] S.D.Gosling, S.J. Ko, T. Mannarelli, and M.E. Morris. "A Room with a Cue: Personality Judgement Based on Offices and Bedrooms". Journal of Personality and Social Psychology 82 (2002): 379-398

[17] U. Rutterfield and G.C. Cupchik. "Perceptions of Interior Spaces". Journal of Environmental Psychology 16 (1996): 349-360

[18] J.M. Wiemann and M.L. Knapp. "Turn-Taking in Conversations". Journal of Communication 25 (1975) 75-92

[19] R. Hopper. "The Taken-for-Granted". Human Communication Research 7 (1981): 195-211

[20] N. Tavuchis. Mea Culpa: A Sociology of Apology and Reconciliation. (Stanford, CA: Stanford University Press, 1991)

[21] J. Hewitt and R. Stokes. "Disclaimers". American Sociological Review 40 (1975)1-11.

[22] Claudia Leacock, Martin Chodorow, George A. Miller. "Using Corpus Statistics and WordNet Relations for Sense Identification". Association for Computational Linguistics (1998) 147- 165

[23] Noam Chomsky. *Aspects of the Theory of Syntax.* MIT Press, 1965.

[24] Noam Chomsky and H. Lasnik. "Principles and Parameters Theory". in Syntax: An International Handbook of Contemporary

[25] Andrew Carnie. *Syntax: A Generative Introduction.* Blackwell Publishing. 2nd Edition, 2006.

[26] Lisa Travis. "Parameters and Effects of Word Order Derivation". PhD. dissertation, MIT.

[27] Noam Chomsky. *Lectures on Government and Binding: The Pisa Lectures.* Holland: Foris Publications, 1981.

[28] Hilda Koopman and Dominique Sportiche, The position of subjects. Lingua. 85:1, 211-258.

[29] "Python Programming Language Official Website." Python Programming Language Official Website. Web. 2nd March 2011. *http://www.python.org/*

[30] "Natural Language Toolkit." Natural Language Toolkit.Web. 2nd March 2011. *http://www.nltk.org/*

[31] Tibor Kiss and Jan Strunk. "Unsupervised multilingual sentence boundary detection." Computational Linguistics (2006) 32:4, 485-525.

[32] Paul DiMaggio, Eszter Hargittai, W. Russell Neuman, and John P. Robinson. "Social Implications of the Internet". Annual Review of Sociology (2001) 27, 307-36

[33] "The Friend of a Friend (FOAF) project — FOAF project." The Friend of a Friend (FOAF) project — FOAF project. Web. 22nd February 2011. *http://www.foaf-project.org/.*

[34] Steven Bird, Ewan Klein, Edward Loper. *Natural Language Processing with Python.* O'Reilly Media. 2009.