Domain-Specific Semantic Role Labeling: Portability of Existing Resources

by

Nick Reese

B.A., Northern State University, 2012

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirement for the degree of

Master of Arts

Department of Linguistics

2015

This thesis entitled:

Domain-Specific Semantic Role Labeling: Portability of Existing Resources

written by Nicholas Reese

has been approved for the Department of Linguistics

_____

Dr. Martha Palmer, Chair of Committee

_____

Dr. Wayne Ward

_____

Dr. Mans Hulden

Date _____

The final copy of this thesis has been examined by the signatories, and we

Find that both the content and the form meet acceptable presentation standards

Of scholarly work in the above mentioned discipline.

Reese, Nicholas (M.A., Linguistics. Department of Linguistics)

Domain-Specific Semantic Role Labeling: Portability of Existing Resources

Thesis directed by Professor Martha Palmer

PropBank-style (Kingsbury and Palmer, 2002) semantic role labeling has good coverage in several general domains, from the Wall Street Journal Corpus (Palmer et al., 2005) to the medical domain (Albright et al., 2013). The purpose of this project is to explore the efficacy of this labeling schema in the science domain. The My Science Tutor project (Ward et al., 2011) has an abundance of domain-specific data available to evaluate the coverage, portability, and usability of PropBank in sub-domains from the the Full Option Science System (FOSS), such as *Energy and Electromagnetism*, and *Living Systems*. The labeler usability will be tested in an off-the-shelf state and compared with manual annotation of the same data, all of which will be taken directly from the My Science Tutor project. A mapping of Propbank- to Phoenix-style annotation will also be devised, and machine learning classifiers for automated output will be created and evaluated.

Acknowledgements

I would first like to thank Martha Palmer, my thesis advisor, for bringing not only coherence and direction to this project, but also for introducing me to Wayne Ward and the opportunity of working with My Science Tutor in the first place, which has given me the incredible circumstance of developing and cultivating my fascination with dialogue systems. Martha's deep-seated intuitions about lexical semantics, and the field of linguistics as a whole, have enlightened and inspired me and my work in the field. Wayne Ward's unique perspective originally created the overarching vision for this project, and his continuing advice, leadership, and intellect, not only as a boss at Boulder Language Technologies, but also as an informal instructor, will always be appreciated. This thesis could not have been possible without the guidance, experience, and insight provided to me by both Martha and Wayne. Mans Hulden's knowledge of Python and Machine Learning as it relates to linguistic endeavors has provided clarity in my understanding of computer science, mathematics, and linguistics, bringing cohesion where it was unquestionably needed. To all three of the people on my committee, Martha, Wayne, and Mans: thank you for all of your instruction, patience, and wisdom.

There are several people in the department who have given time and effort to ensure the completion of this project. Lyndsie Clark, thank you for your time in annotating the corpus of sentences that were given for the exploration of this project idea. Claire Bonial, your hard work when you were approached near the end of this project is more than appreciated. Kathryn Conger, the training you have provided me in PropBank annotation has proven invaluable, and your work providing the entire corpus with gold standard PropBank annotations for this project is also appreciated. Wei-Te Chen, your help with my interactions with the ClearNLP interface, and Shumin Wu, for providing help with setting up and completing ClearSRL annotations have helped develop many of the conclusions drawn here. Last but not least, a thank you to Skatje Myers, who

ran ClearSRL on the last 20 sentences, created and troubleshot all of the Jubilee tasks, and ensured fast turn-around on every request.

In my personal life, I have been lucky to have the most supportive and kindhearted of family and friends, and my girlfriend Junia has been a pillar, standing beside me no matter how stressed I am, or how hopeless a deadline may seem.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The ability to interact with machines has been an aspiration of humanity's since even before the advent of computers, and science fiction writers have been expounding on the possibilities for over 100 years now, ever since L. Frank Baum's *Land of Oz* series debuted in 1907. With the rise of computers and technology, this ambition has been coming closer and closer to realization. It is now within the ability of the average person to look to computers for suggestions, such as nearby and highly rated restaurants, or to ask for the answer to a complex math or science question. We rely on computers to stay connected, for entertainment purposes, and even for driving cars. However, it is now becoming increasingly common to turn to computers for an even more critical aspect of our life: education. These implementations can come in many differing forms, one of which is a tutorial system such as My Science Tutor (MyST) (Ward et al., 2011). Since the student gives spoken feedback, a large corpus of spoken dialogues between students and a virtual science tutor has been amassed. The MyST system extracts domain-specific semantic representations from the students' responses, and semantic grammars are used to extract segments into domain-specific frame elements which then are used to guide the tutoring session. One possible method for increasing the robustness and reducing the development cost for this process is to exploit resources such as PropBank (Kingsbury and Palmer, 2002), which are created with specifically to bolster computer understanding of language. The purpose of this project is to explore how well PropBank parses can be mapped onto the science domain semantic representations used in the MyST system, and to also investigate how well PropBank performs on disfluent input from spoken dialogues.

There are several different kinds of educational interactions which can include a computer.

The most general forms of computer aided education include computer-assisted instruction (CAI) and e-learning. E-Learning is becoming increasingly common, and an example of such a system is taking a class online. CAI attempts to accomplish a much larger goal: encoding an entire teaching method and a set of classes onto a computer with which students can work their way through all of the same material in the same order (VanLehn, 2011). However, the state of the art in the field of human computer interaction in education is rapidly moving towards using computers to tutor students with virtual avatars. These implementations of computers in education, also known as intelligent tutoring systems (ITS), are a much larger area of interest for researchers and are distinguished from the other, more general forms of interaction, such as CAI and e-learning, in several ways. Intelligent tutoring systems are considered more advanced and better suited for this sort of work because of their "enhanced adaptability and power" (Steenbergen-Hu and Cooper, 2013), especially given their potential for being re-purposed for use in specific domains of inquiry, allowing higher performance in specialized subjects (Steenbergen-Hu and Cooper, 2013). The interactions can also have the ability to include an infinite number of possible variations for a single session, dependent upon the student and the system itself.

Another reason for the focus on this form of computer-assisted learning is that there is a large body of research which has repeatedly concluded that tutoring students leads to better educational outcomes (Bloom, 1984; Devin-Sheehan et al., 1976; Topping, 1996; Rohrbeck et al., 2003), whether the tutor is a human or a conversational agent. A meta-study performed by Cohen et al. (1982) of 65 independent tutoring programs concluded that tutoring had "definite and positive effects," and demonstrated statistically that tutored students "outperformed their peers on examinations, and they expressed more positive attitudes toward the subjects in which they were tutored." As early as the late 1980s, studies were being undertaken to determine the cost-effectiveness of bringing computers into the classroom to aid instruction (Levin et al., 1987; Niemiec and Walberg, 1987); a meta-analysis by Kulik and Kulik (1991) showed that findings from 254 studies prove that computer aid in instruction "produces positive effects on students," and students outside of the control group performed "0.30 standard deviations higher." While much of this research has focused on scientific and mathematical domains, other studies have shown a positive correlation of ITS with student learning outcomes outside of these subjects, for example, writing (Roscoe and McNamara, 2013), and communication and language learning (Lane et al., 2013; Seneff et al., 2007).

Almost 25 years later, the state of the art in human-computer interaction has drastically changed, but the results are the same. Ward et al. (2013) reported a change of 0.53 standard deviations when computer based tutoring was compared to a control group of untutored students, and a study by VanLehn (2011) found an effectiveness of $d = 0.79$ for human tutoring, and 0.76 for ITS systems, further reinforcing the idea that these computer-aided interactions can have extremely positive educational outcomes. The most important change in the field, however, has been the dramatic decrease in cost for smaller, portable computers with the power to handle complex interactions such as these.

1.1 Dialogue Systems

Intelligent tutoring systems are also called dialogue systems, and are sometimes also known as conversational agents (Jurafsky and Martin, 2009). Dialogue systems can be broken down into several components which work together to create the most realistic synthesis of an actual human tutoring session as possible (Steenbergen-Hu and Cooper, 2013; Ginzburg and Fernandez, 2013). These subsystems are automatic speech recognizers (ASR), a natural language understanding (NLU) component, a dialogue manager, and a text-to-speech system (Ginzburg and Fernandez, 2013). There are many challenges inherent in endeavors such as these, the most prevailing of which is the challenge of training both the ASR and the NLU component to understand natural speech: there are several distinct problems with natural speech, including disfluencies and elided constructions. Other problems include defining a hierarchy of dialogue moves (Becker, 2012; Ginzburg and Fernandez, 2013; Pilkington, 1999), and finding off-the-shelf natural language processing components that provide adequate performance for specific sub-domains (Yi et al., 2007a; Hwang et al., 2010; Choi and Palmer, 2013). Some of these issues do have solutions, but the exact methods are dependent upon the type of dialogue system being used, and the context in which the system is used (Ginzburg and Fernandez, 2013).

The My Science Tutor system (Ward et al., 2011) has documented results in tutoring students across several scientific fields (Ward et al., 2013). The approach of the system is a mixture of domain-specificity and specialization, along with sound tutoring techniques, which has also been proven to be an important factor for student educational outcomes (Cohen et al., 1982). The

original evaluation of the My Science Tutor system involved comparing three groups of students: human tutored, computer tutored, and non-tutored (control). The results were promising, and reinforced the idea that tutoring will benefit students regardless of whether the tutor was a human or virtual agent. Additionally, the difference in the improvement between the human- and computer-tutored groups was shown to be small enough for even minor improvement in human computer interaction to close the remaining gap (Ward et al., 2013). In addition to the fact that dialogue systems are now cost-effective, all of the research has continued to show the advantage of tutoring, and recent improvements in dialogue management theories and implementations have paved the way for more interesting and fulfilling tutoring outcomes from intelligent tutoring systems.

The MyST system is built around a few simple principles, and addresses the "critical importance of scientific discourse in K-12 education" after it was reported by the 2009 National Assessment of Educational Progress (NAEP) that the percentage of students who were proficient in science was 34% in fourth graders, 30% in 8th graders, and 21% in 12th graders (Ward et al., 2013). The amount of students reaching an advanced science level is much lower: Ward et al. (2013) cites the 2005 NAEP statistics which give a level of 1%-2%, and the 2011 statistics have not improved much: 2%. The underlying goal of the MyST system is to establish an environment in which students are allowed to have a natural conversation with a virtual tutor based on multimedia activities such as interactive visuals and animations. A broad level summarization of MyST goals includes question generation, generating explanations, and knowledge co-construction (Ward et al., 2013).

Question generation concerns asking questions authentic to the system, focusing on underlying ideas that will lead to a higher level of understanding and information retention for students. The system requires students to generate explanations by asking open-ended questions that motivate students to construct novel descriptions of phenomenon. These open-ended questions begin broadly and narrow as the line of questioning in a sub-topic (there may be several sub-topics per session) progresses towards the correct or goal answer. This leads to knowledge co-construction by giving students an opportunity to evaluate their own thoughts as they are presented with either slight direction or an explanation from another individual, dependent upon the quality of the student answer. The MyST conventions call the greater interaction between the sys-

tem and the student a *multimedia dialogue*, because students "simultaneously listen to and think about Marni's questions while viewing illustrations and animations or interacting with a simulation" (Ward et al., 2013). A final goal of the MyST system is to strive to entirely close the gap between human- and computer-tutored student's test score improvements.

1.2 Parsing Systems

One of the major questions that must be answered when designing these systems revolves around what is done with the input after it is processed by the automatic speech recognizer (ASR). When the student speaks, the ASR receives that input and generates the string of words that it best corresponds to (Ginzburg and Fernandez, 2013). There have been several approaches proposed for extracting the strings of words from the spoken input data. The MyST system uses a method called Viterbi Beam Search, wherein there are two models: a language model and an acoustic model. The language model assigns a probability to a string of words, and the acoustic model assigns probabilities based upon feature extraction from the processed acoustic data. The ASR system chooses the words which attain the highest probability and considers them the correct ones (Bolaños et al., 2001; Ward et al., 2013). This is a fairly widespread implementation of an ASR, and is used in many state-of-the-art systems. The main goal of an ASR system is to transform an acoustic signal into words; however, it does not attach any meaning to those words when it is finished (Renals and Hain, 2013).

After the ASR is finished with the input, it is handed to the semantic parser. Nederhof and Satta (2013) define parsing as "the process of automatically analyzing a given sentence, viewed as a sequence of words, in order to determine its possible underlying semantic structure." There are many approaches to obtaining a proper semantic parse, and many of the most advanced systems utilize a technique which melds statistical information from a corpus with linguistic representations of a given language (Palmer et al., 2005; Nederhof and Satta, 2013). The linguistic representations used in a given system are an important factor in the extraction of the features in a given sentence, and are probably one of the most important aspects of a dialogue system outside of the dialogue manager itself. Computational models of semantics such as these are usually created by utilizing a wealth of linguistic information, and include large corpora, syntactic infor-

mation, and semantic relation information; this endeavor as a whole is most widely known as constructing semantic representations, and makes use of lexical semantics. There have been many approaches to computational lexical semantics, including the Proposition Bank (Kingsbury and Palmer, 2002), WordNet (Fellbaum, 1998), VerbNet (Kipper et al., 2000; Kipper-Schuler, 2005), and FrameNet (Lowe et al., 1997). Every one of these resources has its strengths and weaknesses, and an overview of each will be presented in Chapter 3. The main objective of all of these projects is to address a problem that has continually plagued accurate information extraction in natural language processing: correct predicate-argument structure assignment (Kingsbury and Palmer, 2002). The consensus on how to create this type of system is to spend large amounts of time and money annotating corpora with specific annotations in order to allow accurate prediction of predicate-argument structure.

The Proposition Bank, more commonly known as PropBank, has approached the challenge by developing an annotation schema consisting of argument labels corresponding to patterns associated with verb classes. The theory behind the classes is that of Levin (1993), whose work follows the

> fundamental assumption that the syntactic frames are a direct reflection of the underlying semantics; the sets of syntactic frames associated with a particular Levin class reflect underlying semantic components that constrain allowable arguments (Kingsbury and Palmer, 2002)

A crucial aspect of PropBank is that verb arguments will be the same regardless of their location in a sentence. For example, in these sentence alternations:

1. (John) broke (the window)
2. (The window) was broken by (John)

It is important that the logical structure always remains break(John, window) where "break" is a verb which takes an agent and a patient (i.e. break(agentive argument, patient-like argument). PropBank has decided to avoid much of the controversy surrounding the words "agent" and "patient" by assigning the roles Argument 0 (Arg0) and Argument 1 (Arg1), where Arg0 is the entity

which is more proto-typically agent-like, and Arg1 is the proto-typically patient-like argument. Within this schema, the two sentences above would have the same arguments, regardless of sentential position: Arg0 is "John" and Arg1 is "the window." There are also several other argument tags for modifiers such as temporals, directionals, manner, extent, and many more, providing rich semantic annotation for semantic relations.

My Science Tutor uses a system known as the Phoenix Spoken Language System (Ward, 1994) to derive semantic interpretations of a sentence. This system was created specifically in response to the need for robust parsing when confronted with "the noisy and ill-formed nature of spontaneous speech and the uncertainty of the speech decoding process" (Ward, 1994). It was developed largely for use in environments which are domain-specific and focuses on "information structure in an utterance rather than its grammatical structure" (Ward, 1994). Phoenix uses a language model to map words onto strings of words written into a semantic grammar for a specific domain. It uses "frames" to represent the possible moves of the task, or actions that can be taken by the system in response to user input. These frames are further broken down into "frame elements," which are mapped to a semantic representation corresponding to correct input which would move the interaction forward. The system itself understands pre-coded strings of words, and the semantic grammar aligns input to these known sequences by computing scores and selecting the maximum score across all possible frames. This highest score is selected as the correct interpretation, and allows intelligent tutoring systems to decide, for example, if a student has uttered a correct response to a tutor's question. Because of the domain specificity and the corresponding specialization potential with language systems such as Phoenix, they are used in dialogue systems more often than the earlier-mentioned, more general purpose lexical semantic representations of language. However, in spite of the great advancements in the field, there has been very little exploration of using existing lexical resources in dialogue systems in recent literature.

1.3 Summary of Thesis Objectives

PropBank-style semantic role labeling has proven coverage in several general domains, from the Wall Street Journal Corpus (Palmer et al., 2005) to the medical domain (Albright et al.,

2013). This thesis project explored the efficacy of this labeling schema in the science domain. The My Science Tutor project has an abundance of domain-specific data available to evaluate the coverage, portability, and usability of PropBank in sub-domains from the Full Option Science System (FOSS), such as *Energy and Electromagnetism*, and *Living Systems*. ClearSRL (Wu and Palmer, 2011) is the PropBank labeler that was utilized to extract predicate-argument relations from the data. The labeler usability was tested in an off-the-shelf state and compared to manual annotations to explore problematic cases and their underlying causes. A mapping of Propbank- to Phoenix-style annotation was defined based upon this analysis, and machine learning classifiers for automated output were also created and evaluated.

The project research objectives that have been completed are as follows:

1. A semantic role labeler, with no training in the My Science Tutor domain, was run on 649 sentences in total, consisting of 3,666 predicates. These sentences are divided into two groups: propositions used to guide student tutoring sessions, and a much larger body of student responses to MyST prompts. The sentences were manually annotated using PropBank-style annotation and Jubilee.
    (a) Accuracy of the automatic semantic role labeler data was evaluated by comparing it to the manual annotations.
2. A mapping from PropBank- to Phoenix-style representations was created in order to determine portability, usability, and problem areas in the semantic role labeling for the application.
    (a) Mapping began by manually comparing a small list of standard, adult-level propositions which are used to create the tutoring sessions, across Phoenix- and PropBank-style annotation. Problem areas included comparatives, implications, and causatives, as well as relative clauses and disjointed spans.
    (b) The output of the automatic semantic role labeler and the manual annotations was compared to assess the feasibility of a deterministic mapping.
3. Machine learning classifiers were developed and their performance was evaluated.

(a) The first decision tree based machine learning classifier was developed to map PropBank argument roles to Phoenix roles, as a deterministic mapping was not a practical solution. This will be referred to as the Mapping Classifier.

(b) Another machine learning classifier, also based on decisions trees, was also developed to derive Phoenix roles directly from the PropBank predicate argument structures. This will be referred to as the Context Classifier.

4. Extensive error analysis was conducted and the linguistic phenomenon and computational issues which are responsible for common mistakes by the automated system were categorized; for example, relative clauses, disjointed spans, and incorrect sense tagged instances.

Chapter 2

My Science Tutor

The benefits of tutoring students, whether by a human tutor or through a virtual agent, are well-known; however, both private and mass tutoring is an expensive endeavor. Many of the systems that have been developed to meet this demand have not focused on science studies in elementary-aged students (Ward et al., 2011), though this is an area of exceptional need in the American educational system (National Center for Education Statistics, 2011; Ward et al., 2013). Other tutoring systems that have been developed have not supported a "natural spoken dialogue between the child and the agent" Ward et al. (2013). The MyST system was designed to address this gap in children's education, and has attained statistical proof of marked improvement in test scores Ward et al. (2013). The tutor in the system attempts to elicit specific answers from students while they explain their understanding of scientific concepts. Much of the interaction is organized around visuals which illustrate basic concepts for the student and the tutor to discuss. The dialogue system operates by extracting structured information from student responses, and the next tutor move is selected by comparing specific aspects of the extracted representation to the expected representation. While there is still much work being done on improving the system in order to close the gap between human- and avatar-tutored student performance, the system does show test score increases from students who use the MyST tutor as compared to those who receive no tutoring at all.

2.1 System Design

The MyST system architecture is organized in this way:

1. BLT-CU Speech Recognizer

2. Phoenix Semantic Parser
3. Phoenix Dialog Manager
4. Flash Application
5. Speech Synthesizer
6. CU Animated 3D Avatar

The first four elements of the system in this list were built using the Boulder Language Technology (BLT) Virtual Human Toolkit (VHT). The BLT VHT gives My Science Tutor the ability to hold real-time conversations with students. It is composed of several subsystems, of which the Phoenix Semantic Parser and the Phoenix Dialog Manager components are the most theoretically interesting in the context of this paper. These two systems work together to create forward movement in a tutoring session, and they are the key elements by which the theoretical basis of their design could significantly affect performance and tutoring outcomes. Ward et al. (2011) give a thorough illustration of how these interact within the system in Figure 2.1.

2.1.1 Interface

The flash application refers to the interface of My Science Tutor that the student interacts with. The speech synthesizer for the system is either Festival (Taylor et al., 1998) or Acapela, two different text-to-speech (TTS) synthesizers. When deployed in a school setting, MyST also has the ability to use recordings by a voice talent instead of a TTS system for a more realistic avatar experience.

The interface of the MyST system is comprised of several main features, and an example screen is shown in Figure 2.2. The CU Animated 3D Avatar, the conversational agent, is named Marni, and she is always visible in the top right-hand corner of the students screen. On her left is a larger area where interactives, static visuals, and animations are shown to give students context for Marni's questions, and to drive the tutoring sessions. On the bottom of the screen is a text input option, which is not used by students in the classroom, though it is used during the testing phases. Students use the spacebar to inform the system that they are responding to Marni's prompts, and the "ON/OFF" graphic informs students when the system is actively listening to their input. It

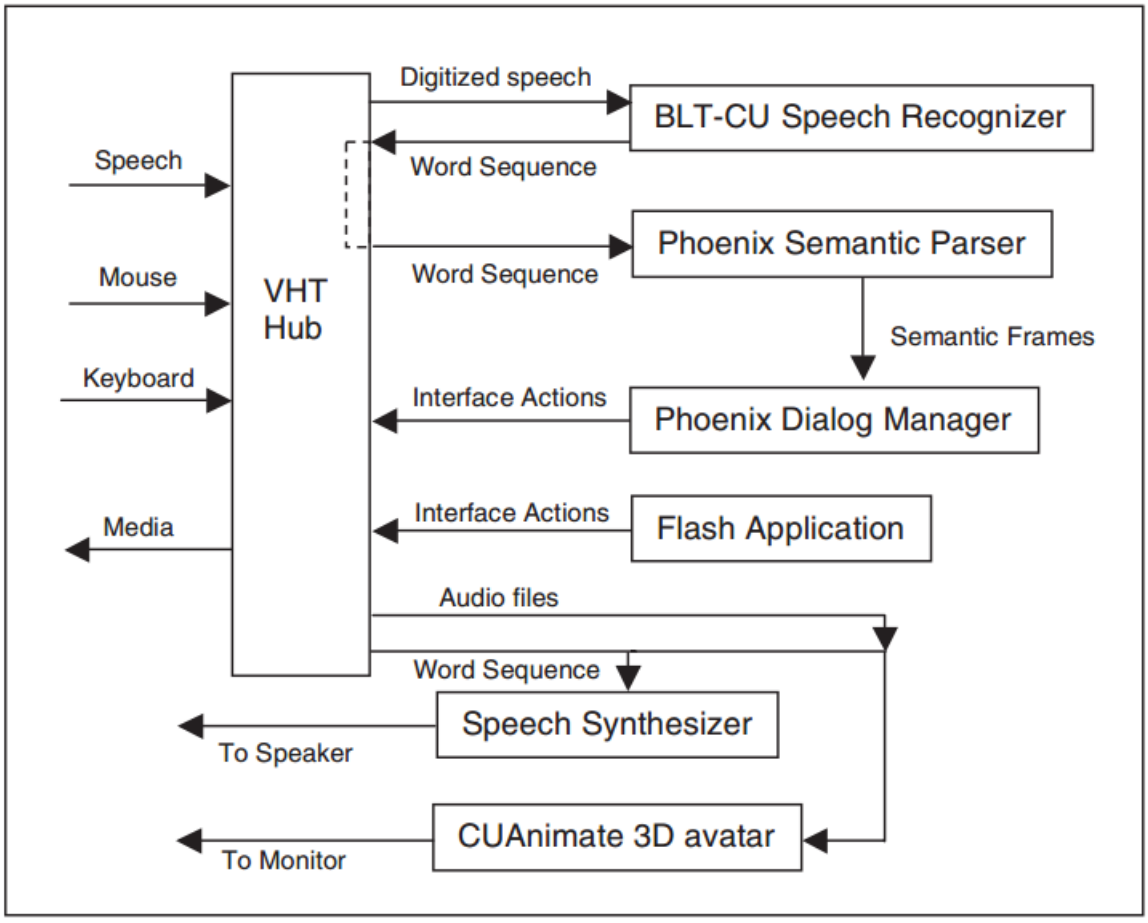

Figure 2.1: Virtual Human Toolkit Architecture

will be dimmed and say “OFF” until the student presses the spacebar, after which it will turn green and say “ON.” There is also a repeat button available for cases in which a student doesn’t hear or fully understand the agent’s last utterance. In the bottom left-hand corner, next to the “ON/OFF” graphic, is a bar which reports the volume level of the input that the system is receiving, which can be used to test whether or not a microphone is working and if the student is speaking loudly enough.

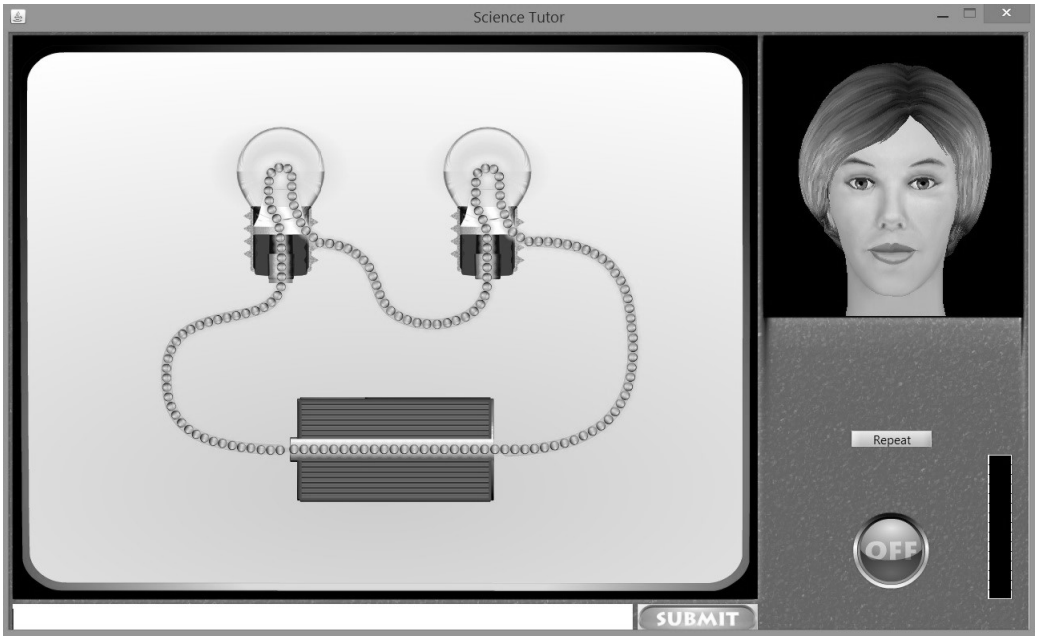

Figure 2.2: Example of the MyST Interface

Marni has the ability to produce facial expressions created by the CU Animate (Ma et al., 2004) module. This allows the system to “produce facial expressions and animation sequences during speech production, while ‘listening’ to the user, or in response to mouse clicks or other input modes” (Ward et al., 2011). The module contains many different characters with these abilities, but Marni is the single character used in all MyST tutoring sessions. During the testing phase of new or recently revised sessions, the Festival (Taylor et al., 1998) text-to-speech synthesizer is used as the voice which is played to the user. However, before the finalized session is deployed to students in schools, a voice talent records over all of the possible prompts for a more natural experience for students.

2.1.2 Phoenix Semantic Parser

The automatic speech recognition (ASR) component in the MyST system is performed by the BLT-CU Speech Recognizer (Bolaños et al., 2001). It is a large vocabulary continuous speech recognition (LVCSR) system, and uses an approach implemented in many such state-of-the-art systems:

> a Viterbi Beam Search is used to find the optimal mapping of the speech input onto a sequence of words. The score for a word sequence is calculated by interpolating Language Model scores and Acoustic Model scores. The Language Model assigns probabilities to sequences of words using trigrams, where the probability of the next word is conditioned on the two previous words. (Ward et al., 2011)

After the ASR generates text from the spoken input, the Phoenix Semantic Parser maps this text onto pre-built semantic grammar frame elements. The method in which the parser represents the semantics of an utterance is generally known as shallow semantics, or semantic role labeling, and is a common approach to computational lexical semantics (both PropBank and FrameNet are also built in this fashion). Shallow semantics is used to represent the relation between entities and events in order to build a representation of meaning in a given sentence or utterance. An example parse, from Ward et al. (2011), for the sentence "Electricity goes from minus to plus" is:

"Electricity goes from minus to plus"
[**Electricity**] (electricity)
[**Flows**] (goes)
[**DirFlow**].[**Origin**] Negative (minus)
[**DirFlow**].[**Dest**] Positive (plus)

Table 2.1: An Example Phoenix Parse

Each frame element (e.g. [**DirFlow**], [**Electricity**], [**Origin**]) has it's own grammar written when the session is created, and these elements can be updated with annotated data after deployment. A grammar, in the context of the Phoenix system, is a list of words or patterns of words which are allowed to fill a frame element. Filling a frame element is the term used for a student answer that completes the requirement for a specific line of questioning, which then moves the session forward. Unknown words in the input are ignored by the parser, and a Stop Words file is created to throw out extraneous words that will not affect the semantics of the utterance; for example, determiners, fillers, and pronouns are ignored. This method achieves a more robust parse

for each utterance, and ensures that off-topic and non-critical strings are ignored, while preserving others which are relevant. This is one way in which the problem of disfluencies is addressed.

Grammar frame capabilities are discussed in more detail in Section 2.2.

2.1.3 Phoenix Dialog Manager

The Phoenix Dialog Manager (DM) takes the semantic parse and decides on the next dialogue move that the tutor will perform. It is entirely responsible for the dialogue interaction between the system and the user, and its goals are:

(a) Maintain a context representing the history of the dialogue
(b) Select a preferred parse from a set of candidate parses given the context
(c) Integrate the new parsed input into the context
(d) Generate a sequence of actions based on the context (Ward et al., 2011)

The Phoenix DM adopts a frame-based model for dialogue management, as opposed to finite-state and inference-based models. This approach allows for a much richer internal structure for the semantic representation (through frames), and these types of systems are "especially well-suited for information tasks, where the system needs to find out some information from the user in order to execute some task (such as booking a ticket or retrieving some information from a database)" (Ginzburg and Fernandez, 2013). These systems are also exceptionally appropriate in a tutoring setting because selected dialogue moves are driven by interaction and not predetermined. This increased flexibility is useful in an educational setting because students will eventually struggle with some aspect of the subject that they are learning, and MyST has the ability to extend the amount of interaction on any topic. In this way, the system can encourage students to think about problem topics in new and different ways, leading to a more comprehensive understanding of the information.

Marni's behavior and dialogue moves are controlled by a task file, which is a linear set of instructions for the dialogue manager to use to make appropriate dialogue move decisions. The DM is a stack-driven algorithm which maintains two stacks: *current*, and *history*, and it attempts to complete frames at the top of the *current* stack, which will them be placed into the *history* stack. The task file and its frames are created to elicit responses from students concerning generic statements

about the subjects they are currently studying in school. These generic statements are also known as propositions, and they are what guide the creation of the task file, its frames and its frame elements. Each task file will aim to examine at least three or four propositions, and one frame is generally required per proposition. A single frame will contain many elements which work together to guide a student towards answering a specific question about the topic of the session and its module. A task file may also contain rules containing ancillary actions if a student does or does not "fill" a frame element, generating dialogue moves or action sequences specific to each case. These rules can also be used to mark and revoice speech directly from the student, which is an important aspect of the questioning the author tutoring style. An example task frame, adapted from Ward et al. (2011), is given in Table 2.2 below:

Frame: FlowDirection
[DirFlow]+
Action: "flash(Flow); synth(Tell me about what's going on here?);"
Action: "synth(What do you notice about the flow of electricity in this visual?);"
[DirFlow].[Origin]+
Action: "flash(Flow); synth(Which side of the battery is the electricity coming from?);"
[DirFlow].[Destination]+
Action: "flash(Flow); synth(Which side of the battery is the electricity going towards?);"
Rules:
If the student got the direction backwards
(DirFlow].[Origin] == "positive") OR ([DirFlow].[Destination] == "negative")
Action: "flash(Flow); synth(Tell me again about the flow of electricity?);"

Table 2.2: A Sample MyST Task Frame

This shows only some of the functionality available in the MyST task file, there are available commands other than *flash()* and *synth()*, such as *movie()*, *show()*, *clear()*, and *speak()*. The *synth()* command is used to call a text-to-speech synthesizer, which will read the text in parenthesis, and the *flash()* command is used to display static visuals, animations, and interactives. The *speak()* command is used for files which have been pre-recorded by a voice talent instead of the TTS audio output, and is generally used only after the session is entirely finished and is deployed in schools. The task file developer will use the frame and frame element structure in order to organize the dialogue in a sequence which allows the system to elicit speech from students to fill frame elements and demonstrate (or illicit) the session propositions.

The development of each session is an iterative process. After the session is deployed to schools, feedback from the system guides future revisions to the structure of the task file. This feedback includes session flow, details of prompt generation, use and utility of visuals, and general completion of frames (Ward et al., 2011). This allows task file developers to ensure that a fulfilling and effective session is occurring for the majority of students, and to adjust any recurrent problems.

2.1.4 The Full Option Science System

The MyST system is deployed specifically in schools in the Boulder Valley School district with science programs that follow the Full Option Science System (FOSS) in classrooms of students in grades 3^{rd}, 4^{th}, and 5^{th} grades. FOSS is a science curriculum developed at the University of California at Berkeley which meets most state standards and the National Science Education Standards. There are five major modules, or broad science subjects, that My Science Tutor covers, which are:

- Electricity and Electromagnetism
- Soil, Rocks, and Landforms
- Sun, Moon, and Planets
- Mixtures and Solutions
- Living Systems

Each of these modules is broken down into several different investigations, and each investigation is further broken into individual sessions. For example, the Living Systems module is broken down into four investigations: Systems, Nutrient Systems, Transport Systems, and Sensory Systems. Each of these investigations can be broken down further, such as the Systems investigation, which is divided into these three sessions: Everyday Systems, The Earth System, and Recycling. In general, MyST creates a single tutoring session file for each session; however, in some cases two shorter sessions may be joined in order to obtain an average tutoring session goal length of 15-20 minutes.

Generally, each individual MyST session contains three propositions, or main ideas, about the specific subject. Each session begins with an introduction, and then moves through the propositions by presenting the student with media to illustrate points. The three types of media used during tutoring sessions are 1) static visuals, 2) animations, and 3) interactive visuals or animations. The first, static visuals, are Flash drawings which are generally used to introduce a topic or point. These can also be useful for students who have a good grasp of the current topic. Animations are Flash animations, but they are not interactive. These can be used to provide further context for a topic, and to help guide the student to the correct answer. They are more informative and illustrative, providing a media format in which complex topics can be more easily grasped because of a direct visualization. The last format of media available to task developers is interactive visuals, which are animations that provide the student a means of directly collaborating with the conversational agent to come to a more complete understanding of a subject. In general, each main idea is introduced with a static visual and an open-ended question. Depending upon the correctness of the student's answer, the system will either a) move on to another sub-point of the proposition, or b) continue questioning about the current sub-point, which includes going to more involved media, such as animations and interactive visuals.

2.1.5 Questioning the Author

The questions asked by the virtual agent follow the style of Questioning the Author (QtA) (Beck and McKeown, 2006). The premise for QtA is that it is a framework and not a script for tutoring, and a major facet of the approach is a requirement that tutors ask open-ended questions. This form of questioning is intended to facilitate rich dialogue interaction between the student and agent, to allow the student to think independently of the author of the texts (i.e. to see the author as fallible), and to promote creation of associations at a deeper-level across several propositions. Murphy and Edwards (2005) meta-analysis comparing nine different tutoring methods found that QtA was one of only two approaches that has the ability to create positive learning outcomes for students, and they also found that it is especially good at promoting high-level thinking and comprehension. In dialogue systems, open-ended questioning is a requirement, as a "yes" or "no" answer would promote neither progress through the session, nor deeper-level comprehension in students.

2.2 Annotation

Phoenix-style annotation is directly driven by considering all conceivable input variations from students in their attempts at answering a question. The purpose of annotation is to provide the semantic grammar used by the Phoenix Semantic Parser with as rich of a representation of potential student answers as is possible. When a session is first written, the frame elements are created by hypothesizing all probable variations for a given answer. While each frame element is unique, and many different formats are possible for an answer, generic examples of MyST frame elements do follow a pattern. In some questions, the answer may be a simple entity. For example, if a question in the Living Systems module were to ask about yeast, with a visual that attempts to elicit a response regarding the fact that it is a single-celled organism, the frame could look like the following:

[Yeast]
(yeast)
[SingleCell]
(single cell)
(one cell)

Table 2.3: A Simple Example MyST Grammar Frame Elements

Where "**[Yeast]**" and "**[SingleCell]**" are the names of the grammar frame elements, and "(yeast)," "(single cell)," and "(one cell)" are example phrases that the task file developer is attempting to elicit. Another way of bolstering the performance of the semantic parser is to update the grammar with direct input. While students are working through the tutoring sessions, the system saves audio clips of each answer that is given by a student. This gives researchers the ability to transcribe all of these audio clips, and then annotate them with the intention of adding utterances that were not previously accounted for to the semantic grammar. This process greatly increases the robustness of the parse, and therefore, the dialogue moves which follow each student answer. Using the "**[Yeast]**" frame element example, a student might give an answer that says "yeast is unicellular" as a response. Annotation is done in exactly the same way as the frame element creation in the grammar files; however, it is done with a "flatter" representation by writing the addition on a single line, as in Table 2.4. These annotated patterns are then added to the semantic grammar under the appropriate frame element, and can be used by future students as

answers which the system will now judge to be correct.

"Yeast is unicellular"
[Yeast] (yeast)
[SingleCell] (unicellular)

Table 2.4: Frame Element Annotation Example

This annotation is a good example of the flexibility that can be encoded into the MyST grammar files. There is no need to annotate the entire string in this case (i.e. "**[Yeast]** (yeast is unicellular)") because this would not provide the system with the robustness required by spoken dialogue. For example, none of these patterns would be usable if the grammars were written in this way:

Q. 'What do you think this visual is trying to tell us about yeast?"

A1 . This visual is showing that yeast, it is unicellular.

A2 . They are unicellular.

A3 . Unicellular

While some of these answers could be arguably "better" or "worse," there is no doubt that a student responding to the question with any of these variations is comprehending the information presented; the frame element should be considered filled and put into the *history* stack. The way this is done is by splitting the sentence into two frame elements with different argument annotations.

There are many aspects of the MyST system of semantic grammar that are not illustrated by this simple example. For that, a much more complex case is required. In the Energy and Electromagnetism module, there are visuals regarding the movement of energy in a circuit. In this case, the grammar representation has the capability of creating entity frames, denoted by an underscore at the beginning of the frame name. There is also functionality that allows creators to denote a word as optional by adding an asterisk (*) before the word in the patterns section. Writers can also create a macro, which allows a frame element to contain alternations of words within the frame specifically; a macro and entity in the MyST grammars is the same contrast expressed by the difference between local and global variables in many programming languages. Macros are

denoted in the semantic grammar by a word typed in all capital letters which is subsumed by a frame element. An example grammar frame that expresses a student answering a question with "The energy flows from the negative end" could look like this:

```
[EnergyFlow]
        (FROM *the *END *THAT [_Negative] *END)
        (FROM *the *END *THAT [_Positive] *END)
FROM
        (from)
        (out of)
        (leaves)
        (leaving)
END
        (end)
        (side)
        (sign)
        (terminal)
        (symbol)
THAT
        (*that *is *has)
;

[_Negative]
        (negative)
        (no bump)
;

[_Positive]
        (plus)
        (positive)
        (bump)
;
```

Table 2.5: A More Complex MyST Grammar Frame

In this example frame, the robustness of a semantic parse of any variation of the utterance "The energy flows from the negative end" is obvious. Each of the items in the macros ("FROM," "END," and "THAT") can be substituted into the sentence, the words in "END" may come before or after the specification of the negative or positive terminal, and the optional words allow students the freedom to express the sentence in both more and less articulate manners than their peers without missing an answer because of it. The inclusion of both [_**Negative**] and [_**Positive**] allows the system to check whether they have said "negative," "positive," (or some version of the

two), or none of the above, which could indicate to the system that the student is off-topic.

The annotation of sentences such as these proceeds in much the same manner as the simpler annotation shown in Table 2.4, and is only slightly more complex. The first annotation example is the correct and preferred annotation, while the second is an also correct but dispreferred version:

"The energy flows from the minus end"
[EnergyFlow] (FROM (from) the **[_Negative]** (minus) END (end))
[EnergyFlow] (from the minus end)

Table 2.6: Annotation of the EnergyFlow grammar frame

In both examples, the same information is being annotated and added to the semantic grammar. However, the second annotation misses an important addition to the **[_Negative]** entity frame that will ensure future students have the ability to use "minus" interchangeably with "negative." This is an important aspect of annotation for direct use with the MyST system (i.e. updating grammar frames). This will be covered in more detail in Section 4.1.1.

On top of this entity annotation, there are also argument-level semantic role annotations completed, with available labels such as "Cause" and "Result." These are called Roles, and can be filled with an entire phrase or subordinate clause. A frame annotation for capturing the propositional content of the utterance is also used, and there are five available frames for annotators to choose from: Causal, Requirement, Defining Attribute, Class, and Movement. A full annotation of an example sentence is:

"The strength of the magnetic force decreases as the distance between objects increases."

Frame:	Causal
Result:	[Strength] (strength of [MagneticForce] (the magnetic force)) [_Decrease] (decreases)
Cause:	the [DistanceIncrease] ([Distance] (distance between objects) [_Increase] (increases))
Roles	**Entities**

Table 2.7: A Full Example of Phoenix Annotation

These two levels of granularity are designed to enable more interesting extraction from each annotation. They will be used to support a PropBank mapping to Phoenix-style annotation from the role annotations, and also to extract new patterns in order to bolster the robustness of the Phoenix semantic grammar from the entity annotation. A single utterance may require more

than one annotation in order to fully represent its propositional content. Appendix A contains 5 example sentences with full Phoenix and PropBank annotations.

2.3 Challenges

One of the most challenging aspects of creating a mapping between the semantic representation in existing lexical resources, such as PropBank, and the MyST system is that the Phoenix DM is not predicate-centered. For example, sentences which define attributes, such as "the negative terminal" (in response to the question "Which end of the d-cell is energy flowing from?") do not require a predicate at all. This creates a problem for automatic semantic role labeling (SRL) systems like PropBank, because there is no predicate for the system to create argument structure around, which has previously been mentioned as a shortcoming of verb-specific approaches (Yi et al., 2007b). However, the writing style of MyST questions attempts to elicit much more elaborate answers than this example. In other cases, such as "Things with iron in them stick to magnets," it is not the possible relation "stick (to)" which is the defining aspect of this sentence. In fact, "stick (to)" could easily be replaced by "are attracted (by)," "cling (to)," or "are pulled (towards)." In this example, "stick" could be considered a predicate but it is not required in the MyST representation. When conducting a tutoring session, it is important that these representations be generalized by the semantic representation required of the answer in as robust of a manner as possible. In the MyST system, the grammar representations (frame elements) may contain several predicates without problems. The point of this representation is adequate and rich semantic coverage over every conceivable version of an answer. One major goal of the MyST semantic representations is that arguments or sequences must be able to float freely across slots, as in the case of non-terminal symbols (or syntactic variables) in context-free grammars.

In another example, specifically a precondition (see Section 3.4 for a discussion of Predicate Types), "You know energy is moving because the motor is running" is not seen as a relation between you knowing something and the causer of that knowledge. In fact, "you" is generally disregarded in all MyST semantic representations. This is because the important relationship for the purposes of a tutoring session is between "energy is moving" and "the motor is running," which is a precondition: condition A is true, even without overt proof, because condition B is true.

Therefore, we can call “energy moving” an Agent because it is causing the Patient (“motor”) to run, and we can call the entire clause “energy is moving” an Agent, while calling “because the motor is running” a Patient. The underlying meaning of the clause is the most important to the understanding of the parsed phrase: “energy moving” is an Agent because it causes “the motor to run.”

Chapter 3

Background

Intelligent tutoring systems combine theoretical frameworks from across several sub-domains such as information extraction, dialogue management, and automatic speech recognition. Without any of these, a dialogue system would not be able to achieve the performance that is required for real-time, natural interaction with students. Each of these disciplines has a long history of research and theory to consider in the context of any dialogue system. Existing lexical resources should also be considered in the context of this paper, as there are many different approaches to computational lexical semantics, and deciding upon a specific schema in an attempt to map to a system such as My Science Tutor requires careful consideration of the strengths and weaknesses of each.

3.1 Information Extraction

Information extraction (also called text analytics) is an integral part of any research project involving the analysis of large amounts of text. The goal of information extraction is to distill the information presented in a text into a manageable and meaningful data structure (Jurafsky and Martin, 2009). There are many sub-problems in the field of information extraction, which taken together can comprise a sophisticated version of information extraction. For example, one the most well-known subtasks that information extraction deals with is named-entity recognition (NER). NER is the process of detecting and classifying all of the proper names in a given text. Once this is done, it will be necessary for the system to also link these named entities to any pronouns or referential entities which also refer to them, and this process is called reference resolution. Semantic parsing, or semantic role labeling, can also be considered a subtask of information extraction,

namely, how are the entities in a sentence related to each other? The last two additional subtasks of information extraction can be event detection and classification, and temporal relation detection and analysis. All of these problems are important in many natural language understanding systems, and it is very hard to have a spoken dialogue without some method of dealing with these cases.

In the context of this research project, information extraction is necessary to resolve the relevant semantic relations from spoken student input in order for the dialogue manager to decide on the next appropriate dialogue move. In MyST, this is handled by the Phoenix semantic parser, which works together with the Phoenix dialogue manager (both of these are explained in detail in Chapter 2). This is also exactly the same kind of work that the ClearSRL system will perform on the input from the MyST sessions, for which it will output the appropriate predicate-argument structure that can be compared to Phoenix-style output. The mapping that automatically produces Phoenix-style semantic representations out of the PropBank labeler uses the extracted information from both the Phoenix- and PropBank-style schemas in order to find correlations between the different predicate-argument labeling styles.

3.2 Machine Learning

A machine learning classifier attempts to build a model of a set of training data and automatically classify a smaller test set of data. Machine learning is a classification task, which involve an algorithm that learns a function "from a domain of input samples to a range of output values" (Clark and Lappin, 2013), and allows researchers an automated way to determine an output class given a vector of features. This is possible by defining a training set, which allows the computer to create an understanding of many sets of features and the output, or class, that they map onto, and then apply this training to attempt to guess unseen feature vectors. Machine learning is a subject that researchers in dialogue management have become very interested in, because a well-trained machine learning classifier could "bypass the need to hand craft the rules governing the behavior of the system" (Ginzburg and Fernandez, 2013) by learning the appropriate next move from a corpus.

While there is only a small amount of data present in the corpus available for this research,

there are still several ways to attempt a machine learning classifier. One popular model that has been used in a variety of new applications in linguistic research is Decision Tree classifiers. Decision trees are built recursively "beginning with the topmost node by (1) computing the best test for the current node according to some *splitting criterion*, (2) creating a subnode for each possible outcome of the test, and (3) recursively expanding each subnode in the same way until a given *stopping criterion* is satisfied" (Schmid, 2013). The first nodes of the Decision Tree for the Context Classifier (discussed in Chapter 4) is shown in Figure 3.1. The first decision made by the classifier is whether or not the feature Phoenix Entities is empty or not. If this is true, it checks whether the Phoenix Frame is Defining Attribute; if it is not true, it checks the PropBank verb sense. The Decision Tree continues until all of the feature vectors in the training set have been classified into the proper class. These classifier implementations are also computationally fast, both in terms of training and processing (Schmid, 2013), which is especially desirable when building a live tutoring system. In this research, the Python package that was used is scikit-learn (Pedregosa et al., 2011), which includes a multitude of options for machine learning classifiers and data analysis.

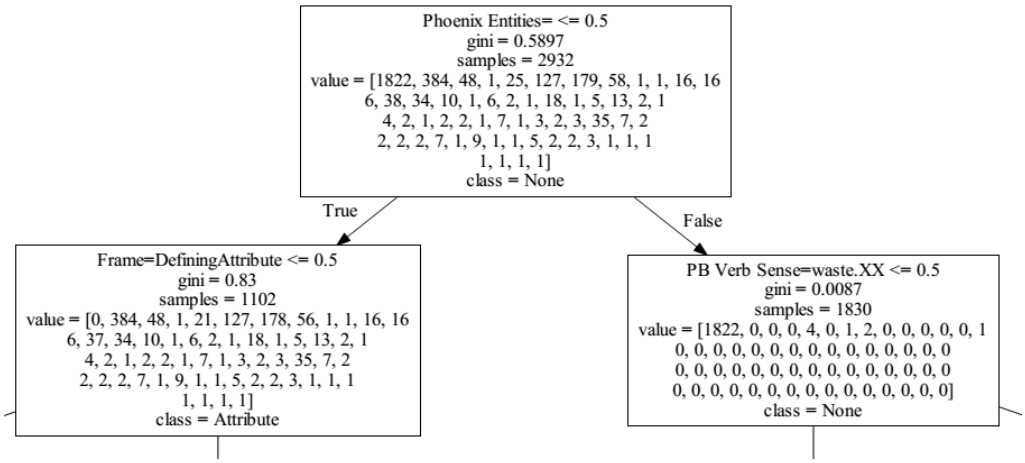

Figure 3.1: Example of a Decision Tree

One of the most powerful tools for examining the efficacy of a machine learning classifier is to look at a confusion matrix, which is a visual representation of the classifiers performance. A confusion matrix can be output as a table, with the True labels listed on the y-axis, and the Predicted labels listed along the x-axis. This allows researchers to view the distribution of true positives, false positives, and false negatives across each class; a perfect classifier would only con-

tain entries along the diagonal, where each Predicted and True class match. Confusion matrices will be presented in Chapter 5 for the machine learning classifiers which were developed.

3.3 Existing Lexical Resources

Because of the nature of this study and what it is attempting to accomplish, a review should be made of the many available lexical resources. While PropBank was the chosen lexical resource for this application, there are several others that could have also been chosen.

3.3.1 WordNet

WordNet (Fellbaum, 1998; Miller et al., 1990) is a lexical resource with psycholinguistic theories of human lexical memory as its main inspiration. WordNet is arranged similarly to, and is often compared to, a dictionary of language, but it also differs in many important ways. It divides the lexicon of languages into five categories: nouns, verbs, adjectives, and adverbs, and omits function words entirely. This organizational structure followed from the realization that word associations are divided in the brain by syntactic categories of words. WordNet attempts to organize lexical information in terms of meaning rather than orthography. An important aspect of the WordNet system is that it gives researchers the ability to create programs which compute word similarity scores (Miller et al., 1990), which could be a useful feature for analysis of the Phoenix-style representations, and more words could be easily added that may increase the robustness of the Phoenix parser. It also achieves almost complete coverage of English verbs (Shi and Mihalcea, 2005).

The major downfall of a WordNet-based semantic parser is that there is an extremely high rate of annotator disagreement when using the system (Fellbaum et al., 2001), which is not ideal in a setting where input is sparse, and restricted domains such as science contain less overall need for large numbers of possible word senses. It has also been noted that while the large number of fine-grained word senses in WordNet may seem like a strong aspect of the resource, it can also potentially become a large problem from a computational point of view, as it makes word sense disambiguation much harder (Artale et al., 1998). The problem with these large numbers of word senses tends to be that they are not based on explicit and concrete criteria; instead, many

of them are based on real-world knowledge, and these distinctions are much more problematic for a computational system Palmer (1998). The fact that WordNet does not encode any syntactic information is also a very large downside for use in many computational systems (Shi and Mihalcea, 2005), while other research has also indicated that WordNet "does not seem to have enough of even the unspecialized knowledge to substantially reduce the number of unresolvable questionable references," (Kieras, 1992).

3.3.2 FrameNet

FrameNet (Lowe et al., 1997) is another lexical resource, with its critical features described as:

- a commitment to corpus evidence for semantic and syntactic generalizations
- the representation of the valences of its target words (mostly nouns, adjectives, and verbs) in which the semantic portion makes use of frame semantics (Baker et al., 1998)

This project is centered around a theory called frame semantics, which holds that "word meanings are best understood in reference to the conceptual structures which support and motivate them" (Lowe et al., 1997). This style of semantic analysis is grounded in the belief that

> word meanings must be described in relation to *semantic frames* - schematic representations of the conceptual structures and patterns of beleifs, practices, institutions, images, etc. that provide a foundation for meaningful interaction in a given speech community. (Fillmore et al., 2003)

The most basic units of analysis used by FrameNet are the frame and the lexical unit, which are defined as a pairing of a word and a word sense. For example, Fillmore et al. (2003) gives this illustration: "the *hot* of temperature and the *hot* of taste experiences are two among the many lexical units that use the adjective *hot*." FrameNet follows many of the other current lexical resources by assigning semantic roles to arguments in order to characterize the relations between a predicate and its arguments. It is best described as denoting prototypical situations that occur, called a "frame" (De Cao et al., 2010). However, FrameNet largely rejects the idea

of a set of "universal" semantic roles, and instead focuses on "particular semantic fields" (Lowe et al., 1997) in which individual predicates may have specific argument roles unseen in any other predicates, leading to well over 1,000 semantic roles. The frames also have a tendency to encode real-world knowledge about the relations they are representing. The main strength of a FrameNet-based system is that it has the ability to generalize across nouns, adjectives, and verbs (Shi and Mihalcea, 2005).

FrameNet also has some problems in a computational setting. Frames may contain an average of 10 frame elements, most of which will not show up in a single sentence (Litkowski, 2004). The SENSEVAL-3 task, created to test the ability of a system to examine a sentence and "identify frame elements within that sentence and tag them with the appropriate frame name" (Litkowski, 2004), shows that an average precision of 0.80 is achieved for all runs across eight teams of researchers (Litkowski, 2004). Like WordNet, it also does not contain the concrete selectional restrictions required in a computational setting (Shi and Mihalcea, 2005). Lastly, because it is still under development, FrameNet is also known for having low coverage, especially compared to other lexical resources, such as PropBank (Shen and Lapata, 2007; De Cao et al., 2010; Shi and Mihalcea, 2005).

3.3.3 VerbNet

VerbNet (Kipper-Schuler, 2005; Kipper et al., 2000) has many of the strengths needed for use in dialogue systems, especially because it is one of the largest computational lexicon specific to verbs that is available (Kipper et al., 2008). It was created in order to address the need to "develop a clear consensus on guidelines for computational verb lexicons" (Kipper-Schuler, 2005). The resource is linked to WordNet and focuses on creating a representation of the possible syntactic information and predicate-argument structures associated with verb senses, which WordNet does not attempt to do in a comprehensive manner. The classes of verbs that are created are based upon the theoretical framework of Levin (1993). Whereas WordNet is focused on providing information about the semantic network while ignoring syntactic information, Levin verb classes explicitly state syntax for each class. VerbNet is dedicated to combining the "traditional lexical semantic information such as thematic roles and semantic predicates, with syntactic frames and selectional restrictions" (Shi and Mihalcea, 2005). The fundamental assumption made by this framework is

that the syntax of a sentence is a direct reflection of the underlying semantics of a sentence. Verbs which occur in the same syntactic frame are assumed to have similar syntactic behavior, leading to robust parsing expectations. It also provides over 90% token coverage of Proposition Bank data (Kipper et al., 2008).

VerbNet is not without its downsides, either. Although VerbNet captures generalizations about large classes of verbs and their possible argument structure and behavior in a sentence, the abstract nature of the thematic roles is too generic to achieve meaningful representations of many verbs (Shi and Mihalcea, 2005).

3.3.4 PropBank

The Proposition Bank (Kingsbury and Palmer, 2002; Palmer et al., 2005) was created specifically to add semantic information on top of the Penn English Treebank, which itself enabled development of accurate parsers for syntactic structures. PropBank was created to add the ability to generate correct predicate argument analysis. A major feature of this system is that argument roles are preserved across variations in syntactic structure, and it is also important that classes of verbs have the ability to be swapped out while the general syntactic order of the sentence remains constant, for example (from Kingsbury and Palmer (2002)):

- A will [meet/visit/debate/consult] (with) B
- A and B [met/visited/debated/consulted]
- There was a [meeting/visit/debate/consultation] between A and B
- A had a [meeting/visit/debate/consultation] with B

However, the semantics of the argument labels will place selectional restrictions on where and what type of arguments can occur in a given position, in order to address, for example:

- A [met/consulted/visited] with B
- The proposal [met/*consulted/*visited] with skepticism

With the addition of semantic and syntactic information provided by Levin classes, the second example here would have constraints so that it would not be possible for the inclusion

of "The proposal consulted with skepticism" or "The proposal visited with skepticism" in the language because it is not consistent with set of syntactic frames or allowable arguments which are permitted by a specific verb class.

There are some issues with PropBank in a computational setting. One of the main critiques of the schema is that much of the corpus it is built upon is from the Wall Street Journal corpus, skewing results towards newspaper-style speech, which can cause a significant drop in performance when used in other domains (Clark, 2013). The verb-centricity of the schema can also cause problems with the amount of data required to train an automated model, and also to make inferences and generalizations outside of the scope of a single verb (Yi et al., 2007b).

PropBank was the chosen lexical resource for comparison to this project because it contains the rich semantic representations and argument structures that are needed in a system such as a virtual tutor. Its direct interface with VerbNet and the clusters of verb-types there is also a strength when considering the interchangeability of predicates that is essential for natural language understanding in a scientific domain.

3.4 Dialogue Management

Each of these components are critical aspects of dialogue systems such as MyST, but the dialogue management aspect of ITS is considered the core of the system. Dialogue management is the process or procedure by which the system's next dialogue move or action will occur (Becker, 2012). As with the creation of lexical resources for correct predicate-argument structure prediction, dialogue management systems are largely hand-made, and the process is very labor-intensive.

The My Science Tutor system dialogue interaction is based around principles put forward by Questioning the Author (QtA) (Beck and McKeown, 2006). QtA is a "mature, scientifically based, and effective program used by hundreds of teachers across the United States" (Ward et al., 2013). It was designed in order to improve student comprehension of texts by creating an environment in which they must construct their own meaning of the following:

1. What is the author trying to say?

2. What does the author mean by what is said?

3. How does this fit within the context of what the author has already said? (Ward et al., 2013)

Murphy and Edwards (2005) have shown the efficacy of QtA at improving comprehension, and QtA offers the MyST system a structured system for designing dialogue prompts and moves in order to guide the student towards relevant aspects of the presented material without directly giving them answers. For these reasons, QtA was chosen as the best method of approaching an intelligent tutoring system for tutoring students in a science domain.

One major aspect that is typically left out of QtA but which is addressed by this research project is the need for computational methods and a structured representation of the spoken input. The MyST system classifies student utterances as propositions realized as semantic frames, which enable the extraction of important entities and relations from what is said, then adapting to the tutorial to the needs of each specific student. However, the system does not actually employ any method of representing the rich semantics of the propositional content and consider it in its decision on the next dialogue move.

The Dialogue Schema Unifying Speech and Semantics (DISCUSS) is an "intermediate linguistic representation that captures the semantics and pragmatics of speech while also allowing for domain-independent modeling of tutorial dialogue" (Becker, 2012). The most important aspect of this schema, in the context of an intelligent tutoring system like MyST, is the inclusion of what are called Predicate Types. A predicate type is used on top of information about the propositional content of an utterance to provide a representation which informs systems about what kind of interaction the tutor and student are having by considering the "interactions between all of the semantic roles found within an utterance" (Becker, 2012). Predicate types in the DISCUSS schema allow distinctions between, for example, an observation about the visuals being presented and a procedural description of how the distinct components of that media would interact to accomplish some goal in the real world. The ability for a tutoring system to choose between making a move as a procedural inquiry which requires a procedural predicate type in response, as opposed to an observation (which may indicate the student has misunderstood the question or does not understand the material) is ultimately the goal of this dimension of the DISCUSS system.

In the initial implementation of DISCUSS within the MyST system, a simplified version of the DISCUSS predicate types was created, and the annotation is completed using the label "frame"

instead of predicate type, which stems from their similarity to the frames of frame semantics theories. There are six predicate types:

1. Defining Attribute: observations about presented information or entities in the world, and can also be used when a student is listing entities for the purpose of answering a question
2. Causal: independent-dependent variable relationship (proto-typical causal relationships) and implications
3. Requirement: an utterance explaining the requirement or precondition for an event or entity
4. Class: list of entities (or a singular entity) which is placed in a hierarchical organizational pattern
5. Movement: any student response concerning entities moving from one place to another
6. Comparative: the relation of two entities (or class of entities)

The most important aspect of predicate types is that they allow the clustering of similar sentence structures and utterance meanings into categories associated with these predicate types which may help clarify the proper dialogue move response.

Chapter 4

Approach

All of the data used in this project was collected from the deployed My Science Tutor system. Most of the data was taken directly from student responses during tutoring sessions, and a small portion was collected from adult-level propositions used in the inital creation of the tutoring sessions. These adult-level sentences were originally used during an exploratory project in order to determine the feasibility of PropBank representations in a scientific domain. The results of that initial project were promising enough to warrant further investigation, after which a larger body of 594 sentences was selected based upon verbs that are generally elicited in order to explain complex tasks and concepts in the science domain, such as "flow" and 'break."

4.1 Data Preparation

The selection and preparation of the data for use in this project was an essential and time-consuming step. The data from the MyST annotations covers five modules, and one module in particular was selected to be the source of most of the annotations: Living Systems. This module was chosen because it is a rich source for several of the predicates that exhibit behavior that this project wanted to consider. If an automatic SRL system, or even gold-standard corrections based on PropBank can perform well on data that is both spoken and domain-specific, there are good indications that an attempt should be made to implement a direct interface between the semantic role labeler and an intelligent tutoring system. The adult-level propositions were taken from both the Living Systems and Energy and Electromagnetism modules in order to determine if there are any significant differences in outcome between the two different scientific domains. Table 4.1 lists the origination of all sentences used in this project.

Origin	Sentences	Predicates
Student Responses to Living Systems Tutoring Sessions	594	3,467
Adult-level propositions used to guide creation of EE* sessions	22	76
Adult-level propositions used to guide creation of LS* sessions	33	123
Total	649	3,666

* Energy and Electromagnetism is shortened to EE, and Living Systems is shortened to LS.

Table 4.1: Sentence Origination for this Project

The sentences were chosen based around a list of verbs that exhibited complex behavior in the Living Systems data, and the Energy and Electromagnetism propositions that were included were also important concepts (such as the flow of energy between terminals) in the scientific domain that needed to be addressed. In many cases in the MyST tutoring sessions, computational models only need to pick out important entities in order to determine if the student is understanding the presented visuals and the relevant key words present within them; however, in other cases, relationships between entities must also be extracted accurately in order to determine the correctness of a student response. For example, when the tutoring session initially begins it is common for the tutor to simply ask "What have you been studying in science lately?" and the student responds with "We've been learning about systems." In this utterance, all that is required is the comprehension of the entity "system," as this informs the dialogue manager that the student is on track and knows what the session will be covering. The Phoenix system handles this as an underspecified Defining Attribute frame with a Class (filled by the entity "system") but no Attribute role. In other cases in the Living Systems module, the tutor may ask a question about the function of the heart, and an actual student response to this, "the heart helps the blood flow through the body..." requires that the system support a more complex understanding of the utterance, which in the MyST system is represented as follows:

"the heart helps the blood flow through the body"	
Frame:	Movement
Agent:	the [Heart] (heart)
Theme:	the [Blood] (blood)
Path:	through the [Body] (body)

Table 4.2: A Basic Phoenix Grammar Representation

In cases such as this, a system based around the PropBank annotation style is hoped to

have the ability to decide on the correct arguments, to map them easily into Phoenix-style annotation (Frame, Role, and Entity annotation). Most sentences contained examples like the simple entity extraction within Defining Attribute frames, but many sentences were chosen specifically because they also contained crucial relationships among specific entities that must be assigned to the proper roles in order to correctly parse the sentence for a correctness judgment. These relationships are critical to an intelligent tutoring system and the feasibility of implementing an automated semantic role labeler is entirely dependent upon complete, accurate comprehension of these utterances.

Before PropBank could complete annotations on the selected data, not only was the Berkeley Parser used to make a parse structure out of the data, but before that step could be completed it was essential that the system have accurate and complete part-of-speech tagging. This was done in order to annotate as many relations within the sentence as possible with the PropBank schema, and was completed with the Perceptron tagger (Honnibal, 2013) from the TextBlob library (Loria, 2014). The part-of-speech tagger found 3,542 verbs out of the 3,666. Of these 3,542, many were errors such as "moose," "foxes," and "hydrosphere," while several important verbs were left out that needed to be added in manually. For example, many instances of "breathe" were correctly tagged as verbs, but others were left out.

4.1.1 Phoenix

The Phoenix-style annotation of the MyST data was completed by hand in a simple text file environment. It followed many of the conventions established in Chapter 2. The 649 sentences were broken down into 22 individual tasks with 30 sentences each (the last task contained 19). A single annotator completed all of the Phoenix-style annotations, which were then corrected by the creator of the schema in order to ensure the highest degree of accuracy and conformity to the guidelines. The specific goal of this annotation is to "support specific tests to determine if the correct entities are present in the correct roles and the required order" (Reese and Ward, 2015). A single student utterance is allowed to have as many annotations as is required to fully represent its content; however, a single clause or entity is not allowed to be annotated more than a single time. For example,

In this example, it would be desirable for the second annotation (the Movement frame an-

"Decomposers eat decaying organic matter and return nutrients to the Earth."	
Frame:	Defining Attribute
Class:	[Decomposer] (decomposers)
Attribute:	[DecomposerAtt] (eat decaying organic matter)
Frame:	Movement
Theme:	[Food] (nutrients)
Goal:	the [Earth] (earth)

Table 4.3: An Example of Phoenix Annotation

notation) to contain "**Agent**: [Decomposer] (decomposers)," but this is not allowed by the Phoenix schema unless "decomposer" is explicitly uttered twice in the student's response, as explained below. Having more than one type of frame annotated from a single utterance is extremely common, and student responses sometimes require many more than two to list out all of the contained information.

Before annotation is allowed to proceed, a "narrative" file is first written and given to annotators in order to guide their annotation of specific modules. For example, all of the high-level propositions that are used in this project were a part of the narrative of the module. A narrative can be defined most simply as the outline of all of the important topics that are covered in each part and session of the larger module (such as Living Systems). These narratives are created and edited by three members of the Boulder Language Technology team before being sent to annotators in order to maximize the scope of the outline and the entities the annotators are allowed to extract. The guideline for the annotation of student responses outlines a few general rules for role and entity extraction.

1. All instances of named entities should be annotated, as long as they are directly relevant to the topic that is being discussed in the tutoring session. The best measure of whether or not a given entity should be annotated is whether or not it is present in the module narrative.

2. All frame annotations must fall into the most specific available frame. For example, if a sentence could be both a Defining Attribute (general) or a Movement frame (more specific), the annotator should chose to annotate it as an instance of a Movement frame.

3. Each clause or entity may only be annotated a single time.

4. There is no perfect, delimiting notion of what the scope of a specific role or entity is, except that it will be "the clause which fills that argument's function" most completely (Reese and Ward, 2015). No emphasis is given to parse structure in a sentence, instead, strings of words which fit the argument as given in the narrative are considered most correct. Even in the case of "Class" role annotation, which are proto-typically single-word nouns are allowed to be entire clauses; for example, "the human respiratory system" or "the energy from the sun."

5. While certain roles are restricted to certain frames (for example, "Attribute" is only present in a Defining Attribute frame), entities (also known as frame elements) are available across any number of different roles.

The most important rule given to annotators is that they should follow the frame, role, and entity (along with interior entity structure) of each utterance as closely as they can to how the same topic is annotated within the given narrative file. Since the narrative file is closely inspected and heavily edited by experts at Boulder Language Technologies before it is given to annotators, it is always deferred to as the gold standard for a given student response. After a narrative is written, it is very rare that new entities be added unless some visual or media is overlooked upon the initial deployment of the narrative file, and this is extremely uncommon. The major difference between the annotation of the data that is present in this project and the regular annotation done within Boulder Language Technologies is that the sentences here were not surrounded by session context. In the regular annotation procedure, BLT annotators have each student response surrounded by the system's last dialogue moves, for example, any responses to the last student utterance and the question that was asked which elicited the sentence under annotation. In this project, the data was simply the sentences themselves and no context. This was done to ensure that the Phoenix and PropBank annotation was completed under the exact same circumstances, with annotators that had the same amount of background information on the sentences as possible, but also because of the complexity of creating, annotating, and extracting pertinent information from the larger context-provided files in a text-only setting.

After the annotator was finished editing the text files with the correct frame, role, and entity annotations, a Python script was written to extract all of that information and store it in

order to support direct analysis and comparison with all of the data extracted from the PropBank-style annotations.

4.1.2 ClearSRL

Before any manual annotation was completed on the MyST data, the ClearSRL (Wu and Palmer, 2011) semantic role labeling system was used to create an initial PropBank-style pass at annotating the data, which was then hand-corrected by experts in PropBank annotation. ClearSRL requires phrase structure parsing to be completed upon data before it can be run, and in this project, the Berkeley Parser (Petrov et al., 2006; Petrov and Klein, 2007) was used for this process, because it has been extensively used on English corpora and because it can be easily retrained for use with any future work on the subject.

The ClearSRL system was built around a standard approach to semantic role labeling, that it is "a multi-class classification problem requiring the identification of argument candidates for each predicate and their argument types" (Wu and Palmer, 2011). The system finds and labels each argument based on the provided parse structure by using the LIBLINEAR (Fan et al., 2008) classifier, a large linear classification library. The ClearSRL system achieved good results, an F-score of 74.4%, on the WSJ corpus using the CoNLL 2005 methodology (Surdeanu and Turmo, 2005), and it is hoped that this ports well to the more specific domain present in the MyST data.

The ClearSRL output is in the same format as the output given by the PropBank annotation program, and is not human-friendly for reading and analyzing. The program was run on the entire corpus of sentences, and then broken up in smaller task files for hand-correction. In the output files, each line is a verb from a TreeBank file, and each line has this order (with an example below):

File	**Sentence**	**Token**	**User**	**Lemma**	**Sense**	**—**	**Argument List**
sentList.parse	563	19	conger	decrease	decrease.01	—	18:1-ARG1 19:0-rel

Table 4.4: An Example of the ClearSRL and Jubilee Annotation Output

The File column is the name of the file with all of the Berkeley parser output in it, and has been slightly modified for space reasons (it is originally "/sentenceList.parse"). The sentence is the sentence number in the full list of sentences in the data, since the task files are organized alphabetically by verb lemma. The token number is a declaration of which token in the sentence is

current predicate under annotation. The lemma and sense inform the system which verb lemma is being annotated and which PropBank verb sense from that lemma has been chosen as the correct one. The argument list gives a list of all of the relevant arguments around the predicate of the sentence by their token number; for example, in this case the ARG1 of the verb "decrease" is token number 18 (since many data structures begin their indexes at 0 instead of 1, this is the same as 19), the number on the other side of the colon refers to the relative position of the argument span in the parse structure, and the "rel" is the token number of the predicate (or relation) itself.

4.1.3 PropBank Gold Standard

The PropBank annotations were completed automatically by the ClearSRL system. The data was then manually hand-corrected by trained annotators for comparison to the automatic output. The program used for the correction was Jubilee (Choi et al., 2009, 2010), an editor developed directly at the University of Colorado at Boulder for use in PropBank annotations (though it has been adapted across several universities since) which combined several historically separate annotation tools into a single one (Choi et al., 2010). The Jubilee editor directly displays the lemma and available sense IDs for that lemma in order to allow quick and easy annotations of the correct predicate sense for each annotation. Each available sense ID has a corresponding frameset file which is imported into Jubilee so that annotators can directly view the argument labels and thematic roles for each sense. Each task file is imported into the program, which displays the parse structure of the sentence and allows annotators to place argument labels on specific nodes around the predicate that is being annotated. A screen shot of the editor is provided in 4.1.

A single annotator hand-corrected all of the automatically generated output from ClearSRL so that any errors in the automated output can be analyzed. The output of this program was made to correspond directly to the output of ClearSRL system for ease of comparison. The annotator followed the guidelines as set out in the PropBank Annotation Guidelines (Bonial et al., 2015).

4.1.4 Data Format

In the case of the PropBank data, all of the verbs present in each sentence were annotated completely in both the automatic (ClearSRL) output and the hand-corrected versions of it. However, in the My Science Tutor data, this is not required; instead, only those pieces of the utterance

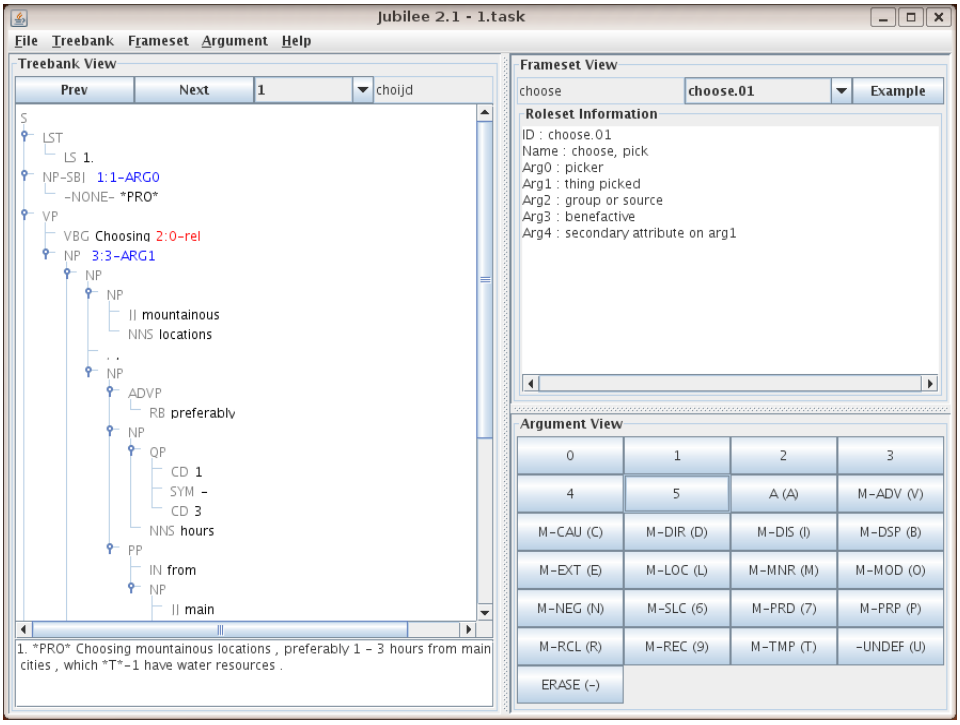

Figure 4.1: The Jubilee Annotation Tool

which were directly relevant to the tutoring task and would be needed to make a correctness judgment for the next dialogue move were considered. This creates a unique situation when mapping PropBank to the Phoenix-style annotations. On the MyST side, many of the semantic role labels and sentential relations analyzed by the PropBank annotation schema in the data can be seen as an over-generation of possible student answers that need to be extracted for the dialogue manager to choose the next move. If a semantic role labeler was ever used in an intelligent tutoring system, a system must also be created to determine which utterance components are worth considering and which are not. For the purposes of this project, all PropBank annotations (including those that do are outside of the span of the Phoenix annotation) are considered.

Several Python scripts were created in order to best map the PropBank annotation data onto the corresponding Phoenix data. While there are many intricacies to the data processing that were completed, a general overview of what has been done is as follows:

1. Extract the annotations from both the Phoenix style annotations and the PropBank style annotations. While this is listed as a single step, a separate script was written for each version of annotation in order to store the data properly.

- Since each schema generates different output and labels differing aspects of the data, Python dictionaries are used to hold all of the extracted annotations based upon the given sentence that was annotated

2. To maximize the speed of analyzing the data after this step, the data was *pickled*, a Python mechanism of data serialization, or storage. This allows other scripts to grab the annotations as a fully formed and stored data structures directly, instead of having to create the data structure every time a new script is ran or created.

3. Generate statistics based upon the shared annotations between the two schemes. A basic overview for the procedure followed by the script is:

   - If MyST generated an annotation for a given part of a sentence, compare the output of each schema.
   - If PropBank has annotations for pieces of the sentence which MyST does not consider useful for the purposes of dialogue management, generate some statistic which counts these cases where there is nothing to compare.
   - Check if PropBank or MyST annotates arguments and entities with differing span.
   - Create a listing of all of the PropBank arguments that correspond to Phoenix annotation roles

4. Generate and track a list of the argument label groupings, verb lemmas, and verb lemma senses which are present within frames in order to determine any patterns for automated classification of frame types

After the annotations were completed from both schemas, the data was transformed into a compatible format and compared in order to analyze the coverage and any mapping trends that can be established. This step had to be completed in two different ways dependent upon the format in which the two schemas output their data. The goal of the data structure was to obtain the token numbers of each annotation from the original sentence in order to analyze annotations on several levels of the Phoenix annotation.

On the Phoenix side, an example annotation is given in Table 2.7. A Python program was written to extract the token numbers from each sentence and place them directly next to the annotated section of the sentence in the stored data structure. One of the major challenges that a text-file annotation presents is ensuring a correct mapping from the role and entity annotations to the original utterance. The disfluent nature of spoken speech present in this data imparts many separate problems which need to be accounted for during the process of token number extraction. The most common instances that caused problems were word repetition, whether it was immediate recurrence (i.e. "the the the food webs") or the frequency of a token's occurrence within a single student response. The resolution of these problems is further complicated by the fact that no token numbers are ever given within the annotation.

The data from ClearSRL and Jubilee are output in the same format, an example of which is given in Table 4.4. Since token numbers are given directly by the output, that was used as the starting point for argument token number extraction. However, since the second half of the token span is given in relation to its node in the parse structure, a mapping was also required from the word in the sentence itself to the word in the parse output. After the correct word is located in the parse output, the Python program simply moved its position upwards in the parse structure from the first word in the annotated argument, as directed by the second half of the token designation, in order to find the full span of the given leaf node by ensuring a balanced amount of parentheses in the string.

The outcome of the initial mapping tests, and the machine learning classifier with this data provides an indication of the usability and portability of a system such as ClearSRL in an intelligent tutoring system. If the gold-standard data does well in a mapping to the Phoenix roles and entities, even if the automatic output does not perform as well, it will be a good indication that a system like ClearSRL could improve future performance of intelligent tutoring programs, and also bring about much more theoretically interesting dialogue management schemas.

A small portion of data (20 sentences) that was not in the original set was also selected, and was used to determine if there is an improvement of the semantic role labeling after it is retrained on the data used in this project. This data was automatically labeled by ClearSRL using three different models:

1. The original, off-the-shelf version of ClearSRL
2. A newly trained ClearSRL version, with no bias towards the MyST data previously annotated
3. A newly trained ClearSRL version, biased on the previously annotated MyST data

The goal of this task is to show that re-training the ClearSRL model will improve performance, allowing for more accurate semantic role labeling and a clearer mapping onto the Phoenix system.

4.2 Machine Learning Classifier

The most important goal of this project was to determine the usability of an off-the-shelf, automated system which evaluated the feasibility of PropBank in an intelligent tutoring system. Initially the project explored deterministic mappings from the PropBank arguments to the MyST roles, and these results are given in Chapter 5. After this was finished, a full machine learning classifier was built to evaluate an automated mapping from PropBank arguments to Phoenix roles, and another was built to derive the Phoenix roles given a PropBank annotation of a sentence. A machine learning classifier using a Decision Tree algorithm produced the most favorable results in a large number of the test classifiers written, and is used in all the classifiers that will be discussed here. There are several different scoring mechanisms available in the scikit-learn package, and a micro-averaged F-score will be used here, which is a useful technique in unbalanced datasets. The package itself defines this as being calculated "globally by counting the total true positives, false negatives and false positives" (Pedregosa et al., 2011).

The first and most straightforward way to attempt to classify the the data is to examine a version which maps directly from the PropBank arguments to the Phoenix roles. This classifier, referred to here as the Mapping Classifier, represents the distribution of all PropBank annotations across particular frames and roles in the Phoenix annotation schema. The Phoenix schema will always have only a single annotation for each word in the utterance. However, in PropBank, this is not the case. The PropBank arguments and Phoenix role spans are generally different, and there will be a different outcome if the classifier simply maps a PropBank role onto all of the Phoenix

roles which it overlaps, or if it only maps to the role with which it shares the most tokens with in a single sentence. In initially addressing the data, ignoring this fact and simply mapping all arguments onto the Phoenix schema annotations was chosen as a starting point. However, it was discovered that this created a large surplus of argument to role mappings, so the classifier was also modified to determine if performance increased with a mapping to only that role with which it overlapped the most. It is also true that a large number of PropBank roles do not map onto any Phoenix role at all, since Phoenix annotation focuses solely on the section of the utterance which is important in the context of the tutoring session. While only one annotation may be completed on a particular segment of the sentence around each relation, if there are several relation annotations in a sentence it is possible and likely that there will be several annotations which have spans including nodes which have already been annotated within the domain of another predicate. Given the work flow of the My Science Tutor system, it is assumed that the system will know what Phoenix frame it is prompting for based upon the question asked to the student. If a system such as ClearSRL is implemented inside the MyST system, we can give the Mapping Classifier PropBank output, since they will be automatically generated in a live system setting. The Mapping Classifier is closely related to the mapping statistics presented, and represents a machine learning classifier's attempt at categorizing the data.

Another way to classify the Phoenix roles given PropBank parses is by considering the larger context of the PropBank predicate argument structures and attempting to derive the most likely Phoenix roles within the span of the PropBank annotation. This classifier will be referred to as the Context Classifier. The Mapping Classifier maps a single PropBank argument to a single Phoenix role regardless of any other information except Phoenix Frame; however, this doesn't use all of the available information, particularly sequence information. The Context Classifier considers the sequence of PropBank arguments and Phoenix entities, and views them as one unit in the form of a joined list; for example, a PropBank annotation of an ARG0 followed by a rel, which is then followed by an ARG1, is given to the classifier as the string "ARG0 rel ARG1." In the same way, a Phoenix Defining Attribute annotation containing the role Class followed by the role Attribute is given to the Context Classifier as "Class Attribute." This imparts a greater context of the PropBank and Phoenix parses to the classifier. The features the Context Classifier is given are the current Phoenix frame, the current PropBank verb sense, the PropBank argument list, and the

Phoenix entity list. All of these features would be available to a live system implementation. The output class labels are built with the list of Phoenix roles in the span of the PropBank annotation. This list of Phoenix roles appears in the same way as the previously mentioned Phoenix entity and PropBank argument features to the classifier, a single unit which establishes sequence information. Since this classifier achieves a relatively high level of performance, the results here are extremely encouraging for future work on the subject. The major increase in performance was due to the greater context available to the classifier which was gained by looking at sequences instead of single arguments. There is one additional difference between the Mapping and Context classifiers, which makes the comparison of their performance indirect. In the Mapping Classifier, individual PropBank roles are mapped onto individual Phoenix roles, so it is implicit which sentence tokens should be assigned to the Phoenix role from the PropBank argument span; however, in the Context Classifier, there is no direct role filler that could be extracted because there is no span information given for individually predicted Phoenix role(s). This is not a problem in most cases; however, the MyST system sometimes requires extracted entities to make a correctness judgment. While this extra step is not fulfilled by the last classifier, it is a straightforward task that can be completed on the Context Classifier output.

While these two classifiers were developed to accomplish different tasks, the scikit-learn implementation is very similar. First, a Python function is defined which returns an array of features and a numpy class array. This function determines the number of features and the type of class that the Decision Tree classifier will be trained on; the implementation outside of this function is exactly the same for both classifiers, and is shown in Figure 4.2.

```
featArray, classArray = buildFeatureArray(phoenixDict, pbDict)
vectorizedFeats = DictVectorizer().fit_transform(featArray).toarray()
scores = cross_val_score(tree.DecisionTreeClassifier(), vectorizedFeats, classArray, scoring='f1_micro', cv=10)
print("F1: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

Figure 4.2: scikit-learn Classifier Implementation

Chapter 5

Results and Evaluation

Determining the usability of the PropBank system in an intelligent tutoring setting is a multi-faceted problem which requires not only comprehensive statistical analysis of how well the PropBank annotations interpret the colloquial data, but also how readily the PropBank output translates into the format in which the Phoenix system understands student input. First and foremost, we must address how well a PropBank system understands both the well-formed and disfluent spoken utterances from students. After there was an understanding of the strengths and weaknesses of automated PropBank annotation in this domain, PropBank data was mapped onto the Phoenix-style annotations to assess whether or not it is portable enough to utilize systematically within an intelligent tutoring system for day-to-day usage. Lastly, the Context Classifier which builds Phoenix representations from automatic PropBank output was evaluated.

5.1 The Corpus

While some initial corpus statistics have previously been presented, a more thorough picture can be given of the content when it is viewed from both the PropBank and the Phoenix framework. In both cases, the corpus is comprised of 649 sentences and 16,616 words. A complete corpus overview is provided in Table 5.1.

Word Count	16,616 words
Character Count	67,497 characters
Average Word Length	4.06 characters
Average Sentence Length	25.6 words

Table 5.1: Initial Corpus Statistics

As stated, a small selection of lemmas was chosen for analysis when the corpus was created. There were 24 common words in the overall My Science Tutor corpus that exhibited important, domain-specific behavior that was deemed critical if a full implementation of an automated semantic role labeler was to be ported directly into the Phoenix system. Another important aspect of these words was their frequency in the student data; more common utterances such as "We have been learning about..." are necessary aspects of tutoring sessions that PropBank annotation must be able to handle. While the PropBank roles are very important in the scope of this project, examination of the VerbNet classes for the lemmas could also yield important and useful generalizations for any further research. The group of 24 words originally chosen is listed out in Table 5.2, along with the relevant PropBank framesets and their corresponding VerbNet classes. In total, there were 231 unique lemmas that were chosen for annotation by PropBank.

5.2 Phoenix

The Phoenix style schema is more sparse than the PropBank data, and will therefore be considered first. This annotation style only includes aspects of the sentence that can be directly used to move forward the tutoring session by deciding whether the student has included an important entity or relationship within their response. There were 1,031 completed annotations on the corpus, 186 different entities present that were considered annotation-worthy, and a total of 3,077 frames and 2,839 entities annotated across all of the data. Table 5.3 summarizes more of these general statistics about the annotations.

Almost all of the sentences contained entities and relationships that the Phoenix system would pull out of the sentence for use by the dialog manager. 20 sentences, unfortunately, contained important lemmas that were selected in the beginning of the project, but did not have entities or roles around them to be considered useful in the progression of the tutoring session. For example, the sentence "things that things that eat that thing and then that thing it goes to that thing and" was originally chosen by the algorithm focusing on specific verbs present in the student data, in this case, "eat." However, the arguments around that verb ("things that" and "that thing") are never used by the system when making correctness judgments because of their non-specificity. If it were possible to recover or elicit the intended entity of the referring expression,

Verb	PropBank Framesets	VerbNet Classes
break	break.01, break.02	break-45.1, split-23.2, break_down-45.8
breathe	breathe.01	breathe-40.1.2-1
carry	carry.01	carry-11.4
cause	cause.01	engender-27
circulate	circulate.01	modes_of_being_with_motion-47.3
digest	digest.01	N/A
eat	eat.01	eat-39.1-1
flow	flow.01	appear-48.1-1
go	go.01, go.02, go.03, go.04, go.08, go.09, go.12, go.15, go.17, go.23	meander-47.7, escape-51.1, become-109.1, escape-51.1, become-109.1, function-105.2, meander-47.7, escape-51.1, become-109.1, function-105.2, occurrence-48.3.1-2, say-37.7
learn	learn.01	learn-14-2-1
leave	leave.01	escape-51.1-1, leave-51.2-1, fulfilling-13.4.1, future_having-13.3, keep-15.2
live	live.01	exist-47.1-1
make	make.01, make.02	build-26.1-1, render-29.90-1
move	move.01	roll-51.3.1, slide-11.2
produce	produce.01	create-26.4, engender-27
pull	pull.01	carry-11.4, push-12-1
push	push.01	carry-11.4-1, push-12-1-1
put	put.01	put-9.1-2
react	react.01	marvel-31.3-9
run	run.01	function-105.2
show	show.01	indicate-78-1-1, transfer_mesg-37.1.1-1-1
survive	survive.01	exist-47.1-1-1
transport	transport.01	send-11.1
warn	warn.01	advise-37.9

Table 5.2: High-Frequency and High-Priority Verbs Chosen for Initial Corpus Creation

Total Annotations	1,031
Role Annotations	3,077
Number of Different Entities Present	186
Total Number of Entities Annotated	2,839
Total Number of Words Annotated	6,269
Sentences with:	
0 Annotations	20
1 Annotation	402
2 Annotations	132
3 Annotations	56
4 Annotations	22
5 Annotation	6
6 Annotations	7
7 Annotations	0
8 Annotations	2
9 Annotation	0
10 Annotations	1
11 Annotations	1
Average Number of Annotations Per Sentence	1.59

Table 5.3: General Phoenix Annotation Statistics

these would have been good sentences to include in the analysis of the data. However, this is not the case and these responses are unusable by the system. Due to the nature of the My Science Tutor project and the source of the data itself, many sentences like this are common occurrences in the corpus.

An overview of the number of Frame and Role annotations is given by Table 5.4 and Table 5.5, respectively. The data shows that the Defining Attribute is by far the most common frame annotation, and this is caused by the fact that many sentences are students naming entities or answering questions in which they do not fully expound upon an answer as expected by the system. It can also include cases where a student says some entity, uses a hesitation particle or filled pause, and then continues with their thought by repeating the original entity. This is also the reason that the Class role is the most common in the Phoenix annotation data, because a Defining Attribute with only a Class is the way that these instances are handled. In some cases, these answers are allowed to fill frame elements in order to move the session forward; however, one major goal of the My Science Tutor program is to encourage students to explain their answers as fully as possible, which is why there are a large amount of other, more complex frames. It should be noted that

not all Defining Attributes are simple listings of entities, however. The Defining Attribute frame is also extremely important for use when students are defining an entity or explaining the function of a concept or entity, which is where all of the Attribute role annotations are found.

Frame Annotations	
Defining Attribute	673
Class	136
Movement	94
Requirement	70
Causal	58

Table 5.4: Phoenix Frame Annotation Statistics

Role Annotations	
Agent	99
Attribute	576
Cause	57
Class	793
Duration	1
Goal	86
Member	105
Path	15
Result	61
Source	10
Subclass	63
Theme	180

Table 5.5: Phoenix Role Annotation Statistics

In the Phoenix annotation schema, many of the roles are restricted based upon what frame is being annotated. It is also possible for there to be several of the same roles annotated within a single frame annotation; for example, an utterance from a student listing entities that they see on a screen will be annotated as a single Defining Attribute frame with several Class roles. Another example is if a single Cause is present in a Causal frame but several Result roles are provided for the singular Cause annotation. Since it is possible for an annotation of a frame to be missing any of the roles inside of it, and it is also possible for an annotation to contain several of the same roles, it is expected the number of frame and role annotations will not add to equal amounts. There were no annotations of the Comparative frame found in the data, which is present in the Phoenix system and also expressed as an ARGM-CXN in PropBank. This is because it is a rare frame overall in the corpus, especially in the Living Systems module.

The final outcome of the annotation of the corpus shows that the project was successful in selecting a large variety of data for use in this project. While there were a large number of Defining Attribute frame annotations, this was expected in the beginning because of its frequency in the overall corpus. Most of the sentences with Defining Attribute annotations also contained other annotations that were important in the project, and a large body of sentences contained no

Frame	Role	Number of Annotations
Defining Attribute		673
	Class	661
	Attribute	576
Class		136
	Class	132
	Subclass	63
	Member	105
Movement		94
	Theme	86
	Agent	67
	Goal	40
	Path	13
	Source	10
	Duration	1
Requirement		70
	Theme	94
	Goal	46
	Agent	32
Causal		58
	Result	61
	Cause	57
	Path	2

Table 5.6: Phoenix Frame and Role Annotation Statistics

Defining Attribute annotations at all, as other frame annotations covered the presented data more correctly.

5.3 PropBank

The PropBank annotations rely on parsed output from the Berkeley Parser and consider all relations in a given sentence. Since this project is attempting to determine the feasibility of the semantic role labeler in an intelligent tutoring system, the ClearSRL system was first used to generate automated output that was then hand-corrected by experienced PropBank annotators. This gave two separate layers of annotations for the PropBank schema data, one Gold Standard and one automatic output, which allowed for a direct test of the performance of the ClearSRL system on spoken data. After this was completed, ClearSRL was retrained on all of the original data and another 20 sentences of data were annotated and adjudicated in order to create a test set with which to judge if there were any performance gains with the addition of the small amount of new training data.

5.3.1 ClearSRL Output

A table of the general statistics for the automatic, ClearSRL output of the corpus is given in Table 5.7. The data given to the ClearSRL system already had all of the relations demarcated from words which the PropBank schema does not annotated. The system picked out 6,889 arguments around the 3,666 predicates that were present in the data. There were 231 unique lemmas (base words) in the data, and 375 unique tokens, the lemmas with their inflections, were annotated, and ClearSRL chose a total of 320 unique senses for the 3,666 relation annotations. It used 19 different argument labels across all of the data, and had an average of 1.88 arguments per predicate annotation. The most common argument label used was ARG1, followed by ARG0 and ARG2.

5.3.2 Manual Output

A table of the statistics for the hand-corrected output of the ClearSRL output is given in Table 5.7. After the completion of the gold-standard, hand-corrected ClearSRL output, there were a total of 7,336 arguments annotated. There were 375 unique tokens, 230 unique lemmas, and 310

unique senses that were selected by annotators. The annotator used a total of 25 argument labels, and the average number of arguments per annotated relation was 2.00. The most common argument label was ARG1, followed by ARG0 and ARG2, mirroring the overall distribution results for the top arguments given automatically by ClearSRL.

	ClearSRL	**Hand-Corrected**
Total Number of Annotated Relations	3,666	3,666
Total Number of Annotated Arguments	6,889	7,336
Total Number of Annotations	10,555	11,002
Total Number of Words Annotated	15,467	15,798
Number of Unique Annotated Tokens	375	375
Number of Unique Annotated Lemmas	231	230
Number of Unique Lemma Senses	320	310
Number of Unique Argument Labels	19	25
Average Number of Arguments Per Relation	1.88	2.00
Average Number of Annotations Per Word	1.47	1.44
ARG0	1,960	1,958
ARG1	2,662	2,672
ARG2	775	849
ARG3	32	41
ARG4	34	39
ARG5	0	1
ARGA	0	1
ARGM-ADJ	2	8
ARGM-ADV	202	453
ARGM-CAU	95	120
ARGM-DIR	41	54
ARGM-DIS	201	158
ARGM-EXT	0	5
ARGM-GOL	1	12
ARGM-LOC	129	142
ARGM-LVB	0	9
ARGM-MNR	155	136
ARGM-MOD	255	259
ARGM-NEG	124	123
ARGM-PRD	1	53
ARGM-PRP	61	96
ARGM-PRR	0	7
ARGM-REC	0	1
ARGM-TMP	159	139
rel	3,666	3,666

Table 5.7: ClearSRL and Hand-Corrected PropBank Annotation General Statistics

5.3.3 Evaluation of the PropBank System

Looking at the previous two tables, there are obvious differences in the amount of data that was annotated in both cases: while the ClearSRL system picked out 6,889 arguments, the annotator afterward picked out 7,336. The number of unique lemma senses dropped in the Gold Standard version, and the number of arguments used increased. ClearSRL did not label any arguments as ARG5, ARGA, ARGM-EXT, ARGM-LVB, ARGM-PRR, or ARGM-REC. In most of these cases, the reason that these arguments did not show up in the automatic output is that they are under-represented in the data that the system was originally trained on.

The overall results for the comparison of the ClearSRL system and the hand-annotated, Gold Standard data are as follows:

General Statistics	
Gold Standard Predicate has More Arguments	623
ClearSRL Predicate has More Arguments	343
Agreed on Argument Number	2,700
Agreed on Sense Selection	3,050
Disagreed on Sense Selection	616
Including rel annotations	
Number of Correct Annotations	9,370
Number of Incorrect Annotations	1,669
Recall	84.89
Precision	88.77
F-Score	0.87
Not including rel annotations	
Number of Correct Annotations	5,765
Number of Incorrect Annotations	1,571
Recall	78.59
Precision	83.68
F1-Score	0.81

Table 5.8: ClearSRL and Hand-Corrected PropBank Annotation Comparison

The most important piece of this chart is the bottom half, which computes the statistics without including the actual relation annotations, since these were all given with the original data (i.e. ClearSRL did not pick its own relations, it was given a set of relations in each utterance that it should annotate). However, the system still performed well on the disfluent data: a 78.59% recall rate, an 83.68% precision rate, and an F1-Score of 0.81 when compared to the Gold Standard

annotations. It is also encouraging that in a vast majority of cases, 73.65%, ClearSRL agreed with the annotator on the actual number of arguments that needed to be annotated around a given relation.

The ClearSRL system was able to correctly identify the correct PropBank verb sense in 3,050 cases, and was incorrect in only 616 cases. While the automated system did not pick out any cases of ER or YY PropBank verb senses, the Gold Standard annotation included 95 cases of ER and 73 of YY. In PropBank, the ER frameset is used to denote a token that was picked out in the data as a relation, but which is not actually being used as one in the sentence and these cases are not annotated. In much the same way, the YY roleset is used in cases where a noun is selected which is either non-eventive or not a head noun. In the schema, these cannot be annotated either. This means that there were a total of 168 relations (out of 3,666) that were chosen for annotation but which could not be completed by the PropBank schema. Therefore, of the 616 incorrect sense selections by ClearSRL, 168 were comprised of ER and YY rolesets.

There were four main cases in which the ClearSRL system made errors in its assignment of roles and role labels. Table 5.9 provides a basic overview of these four types of mistake. The most obvious of the two are mislabeled arguments and unlabeled arguments. In the first instance, the ClearSRL system labeled an argument with a different label than the Gold Standard annotation. In the latter, the ClearSRL system did not label an argument which was labeled in the hand corrected data.

Unlabeled Arguments	889
Mislabeled Arguments	401
Relative Clauses and Disjointed Spans	249
Gold Standard Did Not Annotate	212

Table 5.9: Initial Corpus Statistics

The most common errors made by the ClearSRL system were arguments that it did not annotate but should have, followed by arguments that it mislabeled. Table 5.11 provides a list of the top ten most common arguments that were mislabeled in the data. ARG0 and ARG1 are both top two most commonly unlabeled and mislabeled arguments, which is not unusual since they are also the two most common arguments to be labeled.

Table 5.11 shows the most common mislabeled arguments and the correct label given to

ARG0	219
ARG1	190
ARGM-ADV	181
ARG2	66
ARGM-PRP	41
ARGM-PRD	31
ARGM-DIS	29
ARGM-CAU	24
ARGM-LOC	24
ARGM-MNR	14

Table 5.10: Most Common Unlabeled Arguments

ClearSRL	Gold Standard	Occurrences
ARG0	ARG1	74
ARG1	ARG2	62
ARG1	ARG0	43
ARGM-TMP	ARGM-ADV	19
ARG2	ARG1	16
ARGM-MNR	ARGM-PRD: 14	
ARGM-PRP	ARGM-ADV	13
ARG1	ARGM-ADV	12
ARGM-DIS	ARGM-ADV	11
ARG2	ARGM-DIR	10

Table 5.11: Most Common Mislabeled Arguments

them in the hand corrections of the data. Again, it is unsurprising to see ARG0 and ARG1 top this list since they are exceedingly more common than the next most popular tags.

Relative clauses and disjointed spans, can be broken down into several different cases. In the data, there were a total of 399 cases of relative clauses across 225 relation annotations. The ClearSRL system annotates relative clauses differently than Jubilee does, which is the only case in which the Gold Standard and automatic annotations needed to be mapped to each other in some way. Figure 5.1 shows the difference in marking between the two. On the left-hand side is the ClearSRL output, which uses "0:1-ARG0" and "1:1-R-ARG0" to indicate a relativizer for the ARG0 of the predicate. In Jubilee, however, this is shown by a "*" link: "1:1*0:1-ARG0."

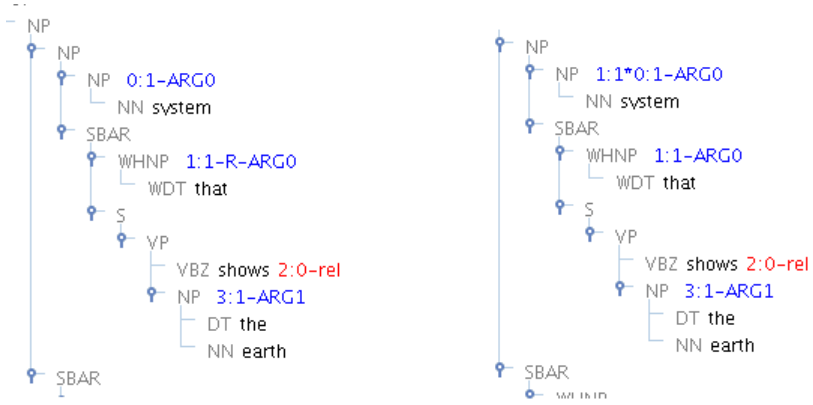

Figure 5.1: Relative Clauses in ClearSRL and Jubilee

In some cases, ClearSRL did not succeed in fully capturing the relative clause. However, out of the total 399 instances of relative clauses containing 824 total arguments, 575 of the refer-

ent arguments were marked correctly, i.e. in Figure 5.1 ClearSRL correctly chose "system" as the ARG0. Out of these 575 correctly marked cases, 336 also had the correct relativizer marked, referring again to Figure 5.1, ClearSRL correctly marked "system" as the ARG0 and also correctly marked "that."

The last important case in which the ClearSRL system failed was with arguments that it labeled but the Gold Standard did not. Table 5.12 gives the 5 most common arguments with which this occurred.

ARG1	64
ARGM-DIS	64
ARG0	37
ARGM-ADV	10
ARG2	9

Table 5.12: Most Common Arguments ClearSRL Labeled but Gold Standard Did Not

ClearSRL was also re-trained on the entire corpus and run on 20 more sentences from the My Science Tutor corpus in order to test if there was any improvement in performance. While the initial corpus is quite small, it was hoped that there would be some difference in performance that could be measured on a small amount of data. There were three models evaluated, the original model, a re-trained model, and a model that is biased towards the MyST data by weighting it more heavily. The results of this test are summarized in Table 5.13 and Table 5.14.

In the 20 sentences, there were 58 PropBank annotations completed. While overall system performance fell because of the size of the corpus it was run on, the performance increases between the old model and the new models biased towards the MyST data it was retrained on are encouraging. The version of ClearSRL that was not trained on MyST data received an F1-Score of 0.67 without rels included, and the new biased model achieved an F1-Score of 0.68. While this is a small and statistically insignificant increase, the fact that the retraining was completed on only 649 sentences and 3,666 relation annotations and the performance increased at all is encouraging for future work using PropBank in intelligent tutoring systems.

Old Model Performance	
Number of Correct Annotations	134
Number of Incorrect Annotation	35
Recall	77.91
Precision	78.36
F1-Score	0.78
New Model	
Number of Correct Annotations	133
Number of Incorrect Annotation	33
Recall	77.33
Precision	78.70
F1-Score	0.78
New Model, Bias Towards My Science Tutor Data	
Number of Correct Annotations	134
Number of Incorrect Annotation	32
Recall	77.91
Precision	79.29
F1-Score	0.79

Table 5.13: ClearSRL Retraining, PropBank rels Included

Old Model Performance	
Number of Correct Annotations	76
Number of Incorrect Annotation	35
Recall	66.67
Precision	67.25
F1-Score	0.67
New Model	
Number of Correct Annotations	75
Number of Incorrect Annotation	33
Recall	65.79
Precision	67.57
F1-Score	0.67
New Model, Bias Towards My Science Tutor Data	
Number of Correct Annotations	76
Number of Incorrect Annotation	32
Recall	66.67
Precision	68.47
F1-Score	0.68

Table 5.14: ClearSRL Retraining, PropBank rels Not Included

5.4 Mapping Results

The mapping began by looking at some hand-generated statistics on the most common arguments present in each Phoenix role. This data is presented in Table 5.15 and Table 5.16. These statistics restrict the comparison within frame annotations, and do not allow the mapping of an argument if the PropBank rel token is not within the span of the Frame annotation. ClearSRL annotated 7,503 arguments within the span of the Phoenix frames, while the Gold Standard annotation assigned 8,077 roles. Some general patterns are noticed by looking at just the role annotations, regardless of frame. The Class role is overwhelmingly an ARG1 with 359 instances or ARG0 with 353 instances, and the third highest is ARG2 with 83 instances, in the Defining Attribute frame. However, in the Class frame this distribution changes: ARG2 is much more common than ARG1 and ARG0 is hardly present. Considering that the numbered PropBank arguments are used for arguments required by the predicate, such as agent and patient, neither of these are surprising results for a Class role. While the Gold Standard and ClearSRL annotations may not have assigned the exact same numbers of these arguments, they agree on which three arguments will be the most likely in a Class role, regardless of the frame. Attribute is another role in which the Gold Standard and ClearSRL annotations agree on the most prominent arguments. Attribute is overwhelmingly ARG1, contains a rel, and the second (or third) most common argument is ARG2, with well under half of the presence as ARG1. These results are promising; ClearSRL achieves the correct distribution of arguments in the Defining Attribute frame, which is the most typical frame in the larger MyST corpus, and also the most populated role of the Class frame, which has the second largest distribution in the corpus. The automatic semantic role labeler also agrees with the correct sequence of the top argument roles in the Class frame's Subclass and Member role.

While the Movement, Requirement, and Causal frames were not as commonplace in Living Systems, they are extremely important in other My Science Tutor modules. The ClearSRL system, in every role underneath these frames, agrees with the Gold Standard annotation as to the most widely annotated argument. In fact, the automatic annotation agreed with the hand corrected version on the most common annotation in all 17 roles (restricted by Frame). Some of the PropBank roles do seem to occur freely across frames and roles, for example, the ARGM-NEG argument simply indicates negation and can be applied very generally, it will not add additional

information for improving the mapping.

While the general distributions of PropBank arguments across Phoenix roles is exceptionally consistent, more high-level generalizations can be made if we look at what full patterns are contained within each role annotation. Roles like Attribute give us a large structure which is broken further down by Phoenix entity annotations, but a breakdown of PropBank argument annotation patterns within these roles may provide a richer account of the semantics of each, allowing a more accurate account of what a mapping would look like. While some of the annotations show problems with the sparsity of data (and the restrictions placed upon the mapping), many of the roles show especially promising patterns of PropBank arguments in this analysis.

Tables 5.17, 5.18, and 5.19 demonstrate the extent that the interior structure of the roles is determined by the frame that they are present in. In almost all cases, a single pattern is shown to have a much higher occurrence rate than the next closest pattern. For example, a lone ARG0 in a role annotation is almost always a Defining Attribute's Class, or the Agent of a Movement frame. However, the large number of other PropBank arguments generally present across Phoenix frames argues against a deterministic mapping and for a more intricate machine classification task.

5.4.1 Mapping Classifier

The Mapping Classifier was built to do an automatic mapping directly from the PropBank arguments to Phoenix roles, irrespective of the context in which they appeared. The classifier was built with only two features in the feature array, the Phoenix Frame and the PropBank argument annotation; given this information, it selects the most likely Phoenix Role for each PropBank argument, one at a time. There were two different versions of this classifier, one which mapped a PropBank argument to all the Phoenix roles that it overlapped with (because of span differences), and another that mapped only onto the Phoenix Role with which it shared the most tokens. They obtain the results given in Table 5.20.

In these cases, the largest amount of the classification is done on cases where a PropBank argument does not map to a Phoenix role. In a training set of 2,270 PropBank arguments, for example, there were 1,265 cases of these None mappings, or 55.7%, creating a 0.59 F1-Score baseline result for the case where only one overlapping Phoenix role was selected, and a 0.55 F1-Score baseline for the classifier version where all overlapping Phoenix roles are mapped. Focusing on

ClearSRL Mapping Results

Defining Attribute Frame	
Class	ARG1: 355, ARG0: 309, ARG2: 78, rel: 31, ARGM-ADV: 10 ARGM-MNR: 6, ARGM-TMP: 4, ARGM-LOC: 3, ARGM-PRP: 2, ARGM-DIR: 2 ARGM-CAU: 2, ARGM-NEG: 1, ARGM-ADJ: 1, ARG4: 1
Attribute	ARG1: 2,121, rel: 899, ARG2: 802, ARG0: 578, ARGM-MNR: 123 ARGM-LOC: 76, ARGM-ADV: 35, ARG3: 35, ARGM-MOD: 33, ARGM-PRP: 30 ARGM-TMP: 21, ARGM-CAU: 19, ARGM-NEG: 16, ARGM-DIR: 10, ARGM-DIS: 9 ARG4: 4, ARGM-GOL: 2
Class Frame	
Class	ARG2: 137, ARG1: 55, rel: 10, ARG0: 5, ARGM-PRP: 2 ARGM-NEG: 1
Subclass	ARG1: 25, ARG2: 14, ARG0: 7, ARGM-CAU: 4, rel: 3
Member	ARG1: 117, ARG2: 24, ARG0: 7, rel: 4, ARGM-NEG: 1
Movement Frame	
Theme	ARG1: 118, ARG2: 10, rel: 2, ARG0: 1
Agent	ARG0: 69, ARG1: 25, rel: 4, ARGM-PRP: 3, ARG2: 2 ARGM-MNR: 2
Goal	ARG1: 44, rel: 27, ARG2: 13, ARGM-PRP: 12, ARG4: 11 ARG0: 10, ARGM-DIR: 6, ARGM-MOD: 2, ARGM-MNR: 2
Path	ARGM-DIR: 14, ARG2: 9, ARGM-LOC: 6, ARG1: 6, ARGM-MNR: 3
Source	ARG2: 9, ARGM-DIR: 2, ARG1: 1, ARG0: 1
Duration	ARG0: 1, rel: 1
Requirement Frame	
Theme	ARG1: 58, ARG0: 32, rel: 10, ARGM-MNR: 9, ARGM-PRP: 6 ARGM-CAU: 2, ARG2: 2, ARGM-NEG: 1
Goal	rel: 46, ARG1: 34, ARGM-PRP: 19, ARG2: 5, ARGM-MNR: 2 ARGM-LOC: 2, ARGM-NEG: 1, ARGM-MOD: 1, ARGM-CAU: 1
Agent	ARG0: 55, ARG1: 3, ARGM-CAU: 1, ARGM-ADV: 1
Causal Frame	
Result	ARG1: 152, rel: 84, ARG2: 41, ARGM-PRP: 36, ARGM-MOD: 19 ARGM-NEG: 14, ARG0: 11, ARGM-ADV: 8, ARGM-CAU: 4, ARGM-MNR: 4 ARG3: 2, ARGM-LOC: 2, ARGM-DIS: 1, ARGM-DIR: 1
Cause	ARG1: 123, rel: 73, ARGM-ADV: 54, ARG0: 53, ARG2: 16 ARGM-CAU: 15, ARGM-NEG: 12, ARGM-LOC: 10, ARGM-TMP: 10, ARGM-MOD: 6 ARGM-MNR: 4, ARGM-PRP: 1
Path	ARGM-LOC: 4, ARG0: 2, rel: 1

Table 5.15: ClearSRL Mapping to Phoenix

Gold Standard Mapping Results	
Defining Attribute Frame	
Class	ARG1: 359, ARG0: 353, ARG2: 83, rel: 31, ARGM-ADV: 28 ARGM-MNR: 12, ARGM-CAU: 6, ARGM-TMP: 4, ARGM-LOC: 3, ARGM-PRP: 2 ARGM-DIR: 2, ARGM-PRD: 1, ARGM-NEG: 1,ARG4: 1, ARG3: 1 ARGM-ADJ: 1
Attribute	ARG1: 2,131, ARG2: 981, rel: 877, ARG0: 549, ARGM-ADV: 141 ARGM-MNR: 124, ARGM-PRP: 73, ARGM-LOC: 67, ARG3: 37, ARGM-MOD: 30 ARGM-CAU: 26, ARGM-PRD: 24, ARGM-DIR: 24, ARGM-NEG: 16, ARGM-DIS: 7 ARG4: 5, ARGM-GOL: 4, ARGM-TMP: 4, ARGM-LVB: 3, ARGM-PRR: 2 ARGM-EXT: 1
Class Frame	
Class	ARG2: 135, ARG1: 42, rel: 10, ARG0: 4, ARGM-GOL: 2 ARGM-ADV: 2, ARGM-NEG: 1
Subclass	ARG1: 25, ARG2: 20, ARG0: 5, ARGM-CAU: 4, rel: 3
Member	ARG1: 114, ARG2: 26, ARG0: 7, rel: 4
Movement Frame	
Theme	ARG1: 116, ARG2: 13, ARGM-ADV: 7, ARG0: 6, rel: 2 ARGM-PRD: 1
Agent	ARG0: 88, ARG1: 22, rel: 4, ARGM-ADV: 4, ARGM-PRP: 3 ARG2: 2, ARGM-MNR: 2
Goal	ARG1: 43, rel: 27, ARG2: 16, ARG4: 16, ARGM-PRP: 14 ARG0: 9, ARGM-DIR: 8, ARGM-LOC: 3, ARGM-MOD: 2, ARGM-MNR: 2 ARGM-CAU: 1, ARGM-ADV: 1, ARGM-GOL: 1
Path	ARGM-DIR: 23, ARG1: 9, ARGM-LOC: 6
Source	ARG2: 8, ARG0: 5, ARGM-DIR: 4, ARG1: 1
Duration	ARG0: 1, rel: 1
Requirement Frame	
Theme	ARG1: 68, ARG0: 22, rel: 10, ARGM-MNR: 7, ARG2: 7 ARGM-PRP: 5, ARGM-ADV: 5, ARGM-CAU: 4, ARGM-NEG: 1 ARGM-PRD: 1
Goal	rel: 46, ARG1: 35, ARGM-PRP: 25, ARG2: 6, ARGM-MNR: 3 ARGM-LOC: 2, ARGM-ADV: 2, ARGM-NEG: 1, ARGM-MOD: 1, ARGM-CAU: 1
Agent	ARG0: 62, ARG1: 3, ARGM-ADV: 2, ARGM-CAU: 1
Causal Frame	
Result	ARG1: 162, rel: 80, ARG2: 42, ARGM-ADV: 33, ARGM-PRP: 32 ARGM-MOD: 17, ARGM-NEG: 13, ARG0: 11, ARGM-PRD: 8, ARGM-CAU: 4 ARGM-MNR: 4, ARG3: 2, ARGM-LOC: 2, ARGM-DIS: 1, ARGM-DIR: 1
Cause	ARG1: 147, ARGM-ADV: 74, rel: 73, ARG0: 61, ARGM-CAU: 26 ARG2: 22, ARGM-NEG: 12, ARGM-LOC: 10, ARGM-TMP: 10, ARGM-MOD: 6 ARGM-DIR: 4
Path	ARGM-LOC: 4, ARG0: 2, rel: 1

Table 5.16: Gold Standard Mapping to Phoenix

Defining Attribute		
Class		
	ARG0	138
	ARG1	102
	rel	16
	ARG0 ARG1	9
	ARG2	7
Attribute		
	rel ARG1	65
	ARG1 rel	31
	ARG2 rel	15
	ARG2	12
	rel ARG1 rel	11
Class		
Class		
	ARG2	58
	ARG1	20
	rel ARG2	4
	ARG0	3
	ARG2 rel	2
Subclass		
	ARG1	11
	ARG2	3
	ARG0	3
	rel	2
Member		
	ARG1	57
	ARG2	10

Table 5.17: Defining Attribute and Class Frame Argument Patterns

Movement		
Theme		
	ARG1	51
	ARG2	3
Agent		
	ARG0	27
	ARG1	2
Goal		
	ARG4	4
	ARG2	3
	ARG0	2
Path		
	ARG2	2
Source		
	ARG2	3
Duration		
	ARG0 rel	1
Requirement		
Theme		
	ARG1	21
	ARG0	5
	ARGM-MNR	4
	rel	4
	ARG0 ARG1	3
Goal		
	rel	19
	rel ARGM-PRP	4
	AR1 rel ARG1	3
	PRP rel	2
Agent		
	ARG0	26

Table 5.18: Movement and Requirement Frame Argument Patterns

Causal		
Result		
	rel	9
	rel ARG1 rel	5
	ARG1 rel	4
	rel ARG1	3
Cause		
	rel ARG1	4
	rel ARG1 rel	3
	TMP rel ARG1	2
	rel ARG0	2
	ARG0 ARG1 rel	2

Table 5.19: Causal Frame Argument Patterns

PropBank Arguments onto All Overlapping Phoenix Roles	
Training Set Size	9,078
Test Set Size	2,270
10-Fold Cross-Validation F1-Score	0.56
PropBank Arguments onto One Overlapping Phoenix Role	
Training Set Size	8,444
Test Set Size	2,111
10-Fold Cross-Validation F1-Score	0.61

Table 5.20: A Classification of PropBank Arguments to Phoenix Roles

training with the PropBank arguments that have been mapped to only those Phoenix roles with which they most overlap (the highest number of tokens) increases the F1-Score by 0.05 points, to achieve a 10-fold cross-validation F1-Score of 0.61, even with a smaller training and test set. In both classifiers, the sparse classes performed the worst as the training and test data was very small; these included Agent, Duration, Member, Path, Source, and Subclass. In both versions, the highest performance was found when the PropBank arguments were mapped to no Phoenix roles, and this is followed by Result in the version with only the highest overlapping role mapping, and Theme with the other. The results also show that the system benefits from being trained with only those Phoenix roles with which the argument overlaps the most.

Figure 5.2 shows the Confusion Matrix for the Mapping Classifier. This visualization makes it clear that the major problem with this classification task is the overwhelming number of None mappings; however, a faint line also runs along the diagonal, proving that the Decision Tree classifier performs better than the baseline example (choosing only the most prevalent Class). It also shows the most frequent mistakes the classifier made, the most common of which are the prediction of the Phoenix role Class instead of Goal, Cause instead of Path, and Subclass instead of Duration. It also shows that the Phoenix role Theme was extremely sparse, and was not found in the testing data.

The data in the above tables was all compiled using the automatic data from ClearSRL. However, Gold Standard annotation was also completed on the corpus, so the results can be reviewed on that data as well. The performance dropped to an F1-Score of 0.60 points for one overlapping PropBank argument to Phoenix role mapping, and 0.55 for all overlapping PropBank argument to Phoenix role mapping. This is likely due to the slightly more nuanced annotations that are completed in the manual annotation, which create more complexity for a machine learning classifier.

5.5 Context Classifier

The Context Classifier, which derives Phoenix roles given the full PropBank predicate argument structure, was also evaluated. Since the My Science Tutor uses Phoenix directly, the classifier was completed by building Phoenix representations from the PropBank output. Although

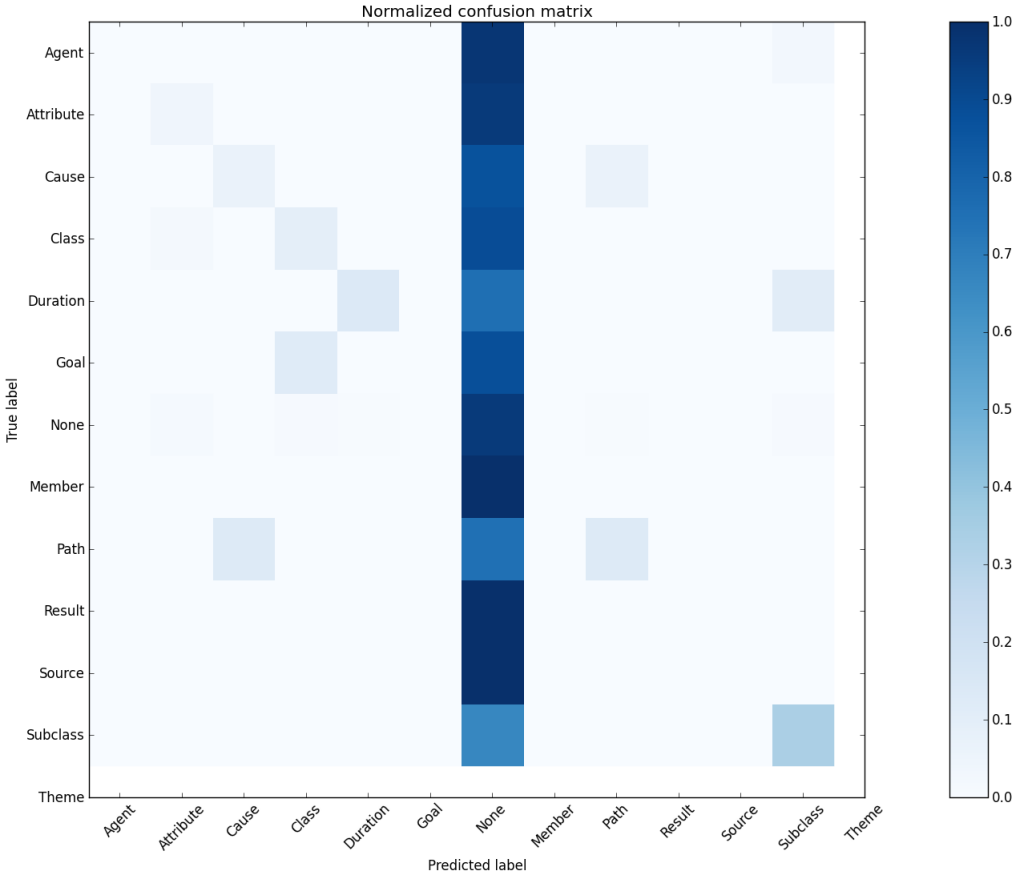

Figure 5.2: Confusion Matrix for the Mapping Classifier

PropBank annotations cover more of the semantic information in the sentence than is required by the system, and the project primarily just interested in the way in which PropBank analyzes the Phoenix frames, roles, and entities themselves, instead of the entirety of the sentence, it is not possible for a live system to understand what is and is not relevant automatically to the Phoenix schema. Therefore, the None mappings were included in the Context Classifier. While a large amount of the training set is still examples of PropBank annotations that do not map to any Phoenix roles (468 out of a test set of 734, or 59.6%, giving a 0.61 baseline F1-Score result for the most populated class), the classifier achieves impressive performance using 10-fold cross-validation with an F1-Score of 0.88. The Context Classifier is given the current Phoenix Frame, the PropBank verb sense, a list of all of the PropBank argument annotations, and the Phoenix Entities; all of which features would be available in a live tutoring setting. It then decides which of the Phoenix roles are most likely to be present inside of the span of the PropBank annotation, and compares it to the class array to return the results. Table 5.21 summarizes the full results.

Training Set Size	2,932
Test Set Size	734
10-Fold Cross-Validation F1-Score	0.88

Table 5.21: Building Phoenix Representations from PropBank Argument Configurations

This classifier was considered the most important test in the project, and an F1-Score of 0.88 in an exploratory project using an off-the-shelf SRL system is extremely promising for future work. The most commonly correct derived Phoenix role was the None class, followed closely by both single Attribute and single Class Phoenix roles. There were many sparse classes as well, creating a large amount of classes scoring a very low average F1-Score. Figure 5.3 shows the Confusion Matrix for this classifier, which contains a large amount of white space, confirming that there were many sparse classes. It also shows a very prominent diagonal line, which illustrates the high performance of the Context Classifier. There are many cases shown on the matrix which show a high error rate for specific classes, such as "Theme Path Goal," which was regularly marked as the Phoenix role Attribute, or "Class Class Class," which was also very regularly marked as Attribute. The sparsity of the data is shown here as well, the classes which are completely white (0.0) were not in the testing data.

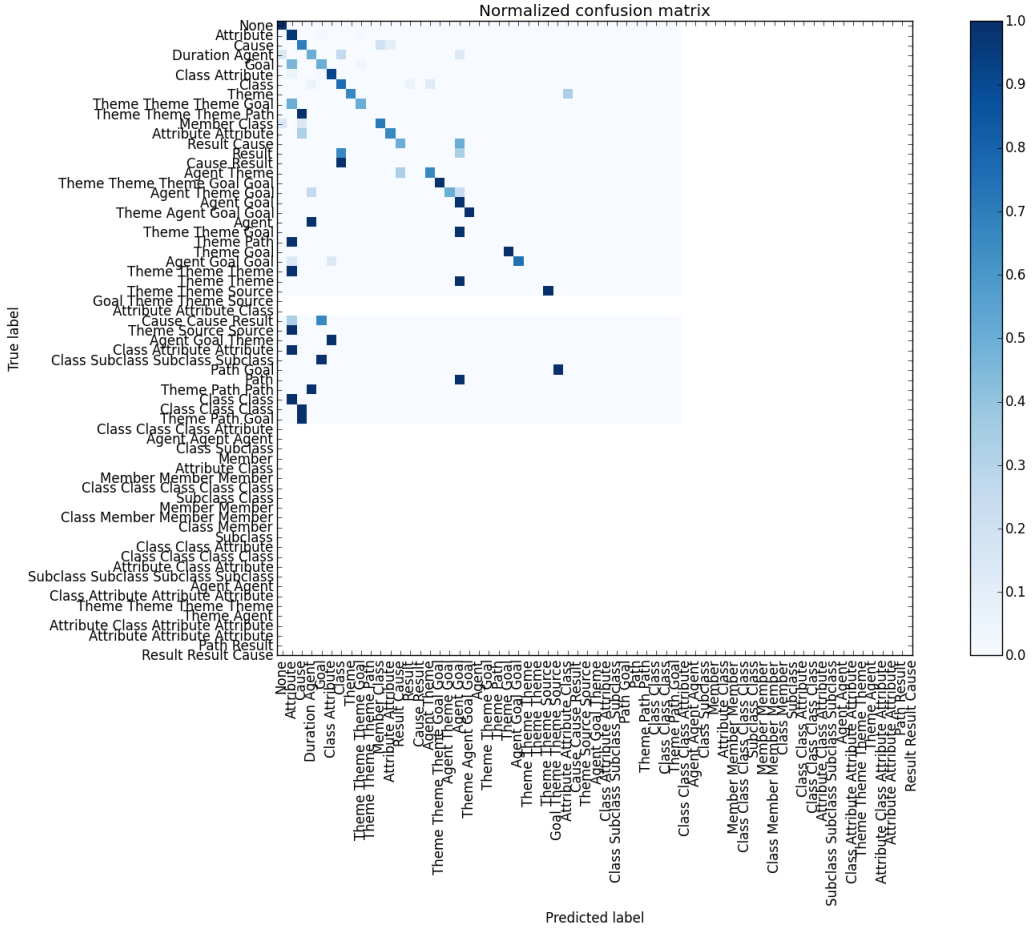

Figure 5.3: Confusion Matrix for the Context Classifier

5.6 Major Problem Areas

There are many problems with the automatically generated data that can be considered for future work. One of the earliest attempts in this project involved using ClearNLP (Choi, 2012) to generate output which could then be hand-corrected by annotators using Anafora. It was found that the automatic dependency parses generated by ClearNLP required a large amount of corrections which were too difficult to map back to the original parses. Therefore, ClearSRL was used because it employs phrase structure parsing instead of dependency parsing.

Another main problem is that the PropBank roles are sufficient for providing argument labels inside of the Phoenix roles, but they cannot relate across the roles themselves. For example, in a Causal frame with a Cause and Requirement role, PropBank provides both the Cause and the Requirement frame with the interior roles; however, it does not have a higher-level annotation which includes the relationship between these two roles.

The largest restriction of the project was the limited data set upon which the SRL system was run, re-trained, and evaluated; this limitation is also important to note for the machine learning classifier, which would greatly benefit from a much larger training corpus. More training data would lower the number of unlabeled and mislabeled arguments that affects ClearSRL's F1-Score, and would also improve performance of the Decision Tree classifier in determining the correct class for less-prevalent feature vector patterns.

Chapter 6

Conclusion

The ability to directly exploit an automatic semantic role labeling system such as ClearSRL in an intelligent tutoring system would be an enormous step forward in the discipline; it would open up a completely new avenue for judging whether the student is both engaged and comprehending the material. Tutorial systems such as My Science Tutor require rich semantic extraction in order to make correctness judgments when moving the tutoring session forward, and any mistakes in the completion of this task can lower student interest and create a feeling of boredom for the student. While the Phoenix system does employ some methods to ensure that this does not happen, superimposing an automatic semantic role labeling system such as ClearNLP on top of the existing structure allows researchers a very promising opportunity to extract more information from a single utterance, which would allow a better correctness judgment to be made for each student's response. Boulder Language Technologies has amassed a large corpus using their My Science Tutor system in schools, and we now have the ability to analyze the usability and portability of an off-the-shelf semantic role labeler, in this case, ClearSRL, and two machine learning classifiers as well.

In order to accomplish this task, experienced annotators from The Proposition Bank have worked together with Boulder Language Technologies using a readily available corpus of spoken student data. 649 sentences, all chosen specifically for the verbs and relations that were deemed important in examining this possibility, have been automatically parsed, tagged, and then hand-annotated for Gold Standard data by PropBank annotators. An annotator at Boulder Language Technologies was given the exact same sentences, which were then annotated in the Phoenix schema and afterwards checked and hand-corrected by the creator of the schema in order

to ensure accuracy. The automatically generated output from ClearSRL and the Gold Standard versions were compared in order to obtain the precision and recall of the system without any previous training in the domain. The entirety of the output data from PropBank, along with the data from the Phoenix annotations, was then transformed into a compatible format to allow comparisons in order to ascertain any general patterns that were found across the data. Another 20 sentences were selected in order to check if there were any performance gains after re-training ClearSRL in the domain, even if on only a limited amount of data.

Another major restriction in porting an automatic semantic role labeler is the sparsity of data that has been added into the training of the off-the-shelf natural language processing tools which are widely available. However, while this is still a major impediment to ClearSRL performance, it was able to achieve an F1-Score over 0.81 (78% Recall, and over 83% Precision scores) on the data (excluding the given relation annotations) without any previous training in the domain, which greatly exceeded expectations. While it was initially thought that the limited domain on which most of PropBank annotations have been completed would be a great hindrance to its performance in a more specific domain such as science, this project has demonstrated the system's ability to perform well across domains, even on a corpus of entirely spoken data. This level of performance is exceptionally promising for future work, especially if the automatic semantic role labeler were to be trained on a much larger amount of data, which would require a much more extensive body of PropBank annotation on the My Science Tutor corpus. Another reason for optimism after the results of this project is that ClearSRL requires parsed sentence structure in order to label the data, and this project had no Gold Standard parses; all of the automatic and hand-corrected annotation was completed on top of the output from the Berkeley Parser. Lastly, the small increase seen after re-training ClearSRL and evaluating the system on another 20 sentences suggests that training on more data would, in fact, bolster system performance.

The machine learning classifiers were the most important test for the implementation of a system like PropBank in the intelligent tutoring domain, and the results were especially promising for future work. Using the output of ClearSRL, the Mapping Classifier was able to achieve a 10-fold cross validation F1-Score of 0.61 for mapping the PropBank arguments onto Phoenix roles, and the Context Classifier achieved an F1-Score of 0.88 when deriving appropriate Phoenix role sequences given a PropBank predicate argument structure.

A larger corpus on which to train these classifiers would result in an improved performance, especially if more dedicated work was completed on feature engineering and more complex machine learning classification methods. However, the Decision Tree achieved impressive performance with a minimal data set, and is known for its low computational and processing requirements. All of the results which were obtained in this thesis suggest that the cutting edge in machine learning, computational semantics, and dialogue management has produced an exciting opportunity to meld the three into a robust tutoring experience that allows the natural flow of conversation between an avatar and students.

6.1 Future Work

While this project had extremely promising results, it also paved the way for future research into problems that should be resolved in order to guarantee the best implementation.

One of the major aspects of the most recent research on computational semantics that has been left out entirely in this project is the encoding of the larger relationships between the arguments that PropBank would annotate, such as discourse relations. The Abstract Meaning Representation (AMR) schema was previously used on 25 sentences of My Science Tutor data in order to determine usefulness in increasing the semantic role labeler performance. In the science domain, there is good possibility that certain relations will not be fully encompassed and analyzed by PropBank annotation, and it is expected there are aspects of the semantics of, for example, independent and dependent variable relationships that may not be well-covered by an automatic semantic role labeler. Another example of this is negative precondition relationships, such as:

"Without grinders organic waste would build up and not get broken down"

```
(a / and
      :op1 (b / build-05
            :ARG1 (w / waste
                  :mod (o / organic)))
      :op2 (b2 / break-12 :polarity -
            :ARG1 w)
      :condition (o2 / organism :polarity -
```

```
:ARG0-of (g / grind-01)))
```

While PropBank would be able to tag individual arguments and how they relate to the predicate, the AMR representation on top of the PropBank annotations may serve to clear up any ambiguity left over after the PropBank labels have been added. In the previous, exploratory project, performance jumped 9.6% with the inclusion of the manual AMR representation on top of the PropBank labels, a dramatic performance boost even in a small data set.

Another approach that could improve performance if used as a feature in the Context Classifier is backing off to VerbNet class membership instead of PropBank sense IDs. The VerbNet classes include more generalized predicate-argument structures associated with classes of verbs, instead of those most commonly seen around a single verb, as in the PropBank senses. This abstraction could allow the classifier to more accurately pick out correct Phoenix roles for a larger amount of predicate argument structures.

The largest portion of future work requires a larger set of data that automatic classifiers could be built on top of. The results obtained by this project substantiate the hypothesis that current state of the art systems such as PropBank are portable to the domain of intelligent tutoring, even with only an extremely small portion of the MyST corpus being annotated.

Bibliography

Albright, D., Lanfranchi, A., Fredriksen, A., Styler, W., Warner, C., Hwang, J., Choi, J., Dligach, D., Nielsen, R., Martin, J., Ward, W., Palmer, M., and Savova, G. (2013). Towards comprehensive syntactic and semantic annotations of the clinical narrative. *Journal of the American Medical Informatics Association*, 20(5):922–930.

Artale, A., Goy, A., Magnini, B., Pianta, E., and Strapparava, C. (1998). Coping with WordNet sense proliferation. In *First International Conference on Language Resources and Evaluation*.

Baker, C., Fillmore, C., and Lowe, J. (1998). The berkely FrameNet project. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 1*.

Beck, I. and McKeown, M. (2006). *Improving Comprehension with Questioning the Author: A Fresh and Expanded View of a Powerful Approach*. Scholastic, New York.

Becker, L. (2012). *DISCUSS: Toward a Domain Independent Representation of Dialogue*. PhD dissertation, University of Colorado at Boulder.

Bloom, B. (1984). The 2 sigma problem: The search for methods of group instruction as effect as one-to-one tutoring. *Educational Researcher*, 13(6):4–16.

Bolaños, D., Cole, R., Ward, W., Borts, E., and Svirsky, E. (2001). FLORA, fluent oral reading assessment of children's speech. *Journal of the ACM*, 2(3):111–130.

Bonial, C., Bonn, J., Conger, K., Hwang, J., Palmer, M., and Reese, N. (2015). English propbank annotation guidelines. Unpublished internal document, Center for Computational Language and Education Research, Institute of Cognitive Science, University of Colorado at Boulder.

Choi, J. (2012). *Optimization of Natural Language Processing Components for Robustness and Scalability*. PhD thesis, University of Colorado Boulder.

Choi, J., Bonial, C., and Palmer, M. (2009). Jubilee: Propbank instance editor guideline (version 2.1). Technical report, Institute of Cognitive Science, University of Colorado Boulder.

Choi, J., Bonial, C., and Palmer, M. (2010). Propbank instance annotation guidelines using a dedicated editor. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC'10)*, pages 1871–1875.

Choi, J. and Palmer, M. (2013). Fast and robust part-of-speech tagging using dynamic model selection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*.

Clark, A. and Lappin, S. (2013). Unsupervised learning and grammar induction. In Clark, A., Fox, C., and Lappin, S., editors, *The Handbook of Computational Linguistics and Natural Language Processing*, chapter Current Methods, pages 197–220. Wiley-Blackwell.

Clark, S. (2013). Statistical parsing. In Clark, A., Fox, C., and Lappin, S., editors, *The Handbook of Computational Linguistics and Natural Language Processing*, chapter Domains of Application, pages 333–363. Wiley-Blackwell.

Cohen, P., Kulik, J., and Kulik, C. (1982). Educational outcomes of tutoring: A meta-analysis of findings. *American Educational Research Journal*, 19(2):237–248.

De Cao, D., Croce, D., and Basili, R. (2010). Extensive evaluation of a FrameNet-WordNet mapping resource. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation*.

Devin-Sheehan, L., Feldman, R., and Allen, V. (1976). Research on children tutoring children: A critical review. *Review of Educational Research*, 46(3):355–385.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Fellbaum, C. (1998). *WordNet*. MIT Press, Cambridge, MA.

Fellbaum, C., Palmer, M., Dang, H., Delfs, L., and Wolf, S. (2001). Manual and automatic semantic annotation with WordNet. In *SIGLEX Workshop on WordNet and other Lexical Resources, NAACL-01*.

Fillmore, C., Johnson, C., and Petruck, M. (2003). Background to FrameNet. *International Journal of Lexicography*, 16(3):235–250.

Ginzburg, J. and Fernandez, R. (2013). Computational models of dialogue. In Clark, A., Fox, C., and Lappin, S., editors, *The Handbook of Computational Linguistics and Natural Language Processing*, chapter Domains of Application, pages 429–481. Wiley-Blackwell.

Honnibal, M. (2013). A good POS tagger in about 200 lines of Python. Retrieved 10 April 2015.

Hwang, J., Nielsen, R., and Palmer, M. (2010). Toward a domain independent semantics: Enhancing semantic representation with construction grammar. In *Proceedings of the NAACL HLT Workshop on Extracting and Using Constructions in Computational Linguistics*.

Jurafsky, D. and Martin, J. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall.

Kieras, D. (1992). Semantics-based reference resolution in technical text processing: An exploration of using the WordNet database in the computerized comprehensibility system. Technical report, Michigan University, Ann Arbor, MI.

Kingsbury, P. and Palmer, M. (2002). From TreeBank to PropBank. In *Proceedings of the Third International Conference on Language Resources and Evaluation, LREC-02*.

Kipper, K., Dang, H., and Palmer, M. (2000). Class-based construction of a verb lexicon. In *Proceedings of the Association for the Advancement of Artificial Intelligence, AAAI-00*.

Kipper, K., Korhonen, A., Ryant, N., and Palmer, M. (2008). A large-scale classification of english verbs. *Language Resources and Evlauation*, 42(1):21–40.

Kipper-Schuler, K. (2005). *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. PhD dissertation, University of Pennsylvania.

Kulik, C. and Kulik, J. (1991). Effectiveness of computer-based instruction: An updated analysis. *Computers in Human Behavior*, 7:75–94.

Lane, H., Hays, M., Core, M., and Auerbach, D. (2013). Learning intercultural communication skills with virtual humans: Feedback and fidelity. *Journal of Educational Psychology*, 105(4):1026–1035.

Levin, B. (1993). *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press, Chicago, IL.

Levin, H., Glass, G., and Meister, G. (1987). Cost-effectiveness of computer-assisted instruction. *Educational Review*, 11(1):50–72.

Litkowski, K. (2004). Senseval-3 task: Automatic labeling of semantic roles. In *Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*.

Loria, S. (2014). Textblob: Simplified Text Processing. Retrieved 10 April 2015.

Lowe, J., Baker, C., and Fillmore, C. (1997). A frame-semantic approach to semantic annotation. In *Tagging Text with Lexical Semantics: Why, What, and How? Proceedings of the Workshop*.

Ma, J., Cole, R., Pellom, B., Ward, W., and Wise, B. (2004). Accurate automatic visible speech synthesis of arbitrary 3D models based on concatenation of diviseme motion capture data. *Computer Animation and Virtual Worlds*, 15(5):485–500.

Miller, G., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K. (1990). Introduction to WordNet: An online lexical database. *International Journal of Lexicography*, 3(4):235–244.

Murphy, P. K. and Edwards, M. N. (2005). What the studies tell us: A meta-analysis of discussion approaches. Presented at the American Educational Research Association, Montreal, Canada.

National Center for Education Statistics (2011). The nation's report card: Science 2011 (NCES-2012-465). U.S. Department of Education, Institute of Education Sciences, Washington, D.C.

Nederhof, M. and Satta, G. (2013). Theory of parsing. In Clark, A., Fox, C., and Lappin, S., editors, *The Handbook of Computational Linguistics and Natural Language Processing*, chapter Formal Foundations, pages 105–130. Wiley-Blackwell.

Niemiec, R. and Walberg, H. (1987). Comparative effects of computer-assisted instruction: A synthesis of reviews. *Educational Researcher*, 3(1):19–37.

Palmer, M. (1998). Are WordNet sense distinctions appropriate for computational lexicons? In *SIGLEX-98, SENSEVAL*.

Palmer, M., Gildea, D., and Kingsbury, P. (2005). The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–105.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Lin-*

guistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL).

Petrov, S. and Klein, D. (2007). Improved inference for unlexicalized parsing. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Pilkington, R. M. (1999). Analysing educational discourse: The DISCOUNT scheme. Technical Report 99/2, Computer Based Learning Unit, University of Leads.

Reese, N. and Ward, W. (2015). Phoenix semantic annotation guidelines. Unpublished internal document, Boulder Language Technologies.

Renals, S. and Hain, T. (2013). Speech recognition. In Clark, A., Fox, C., and Lappin, S., editors, *The Handbook of Computational Linguistics and Natural Language Processing*, chapter Domains of Application, pages 299–332. Wiley-Blackwell.

Rohrbeck, C., Fantuzzo, J., Ginsburg-Block, M., and Miller, T. (2003). Peer-assisted learning interventions with elementary school students: A meta-analytic review. *Journal of Educational Psychology*, 95(2):240–257.

Roscoe, R. and McNamara, D. (2013). Writing pal: Feasability of an intelligent writing strategy tutor in the high school classroom. *Journal of Educational Psychology*, 105(4):1010–1025.

Schmid, H. (2013). Decision trees. In Clark, A., Fox, C., and Lappin, S., editors, *The Handbook of Computational Linguistics and Natural Language Processing*, chapter Current Methods, pages 180–196. Wiley-Blackwell.

Seneff, S., Wang, C., and Chao, C. (2007). Spoken dialogue systems for language learning. *NAACL HLT Demonstration Program*, pages 13–14.

Shen, D. and Lapata, M. (2007). Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Shi, L. and Mihalcea, R. (2005). Putting pieces together: Combining FrameNet, VerbNet, and WordNet for robust semantic parsing. In *Computational Linguistics and Intelligent Text Processing*, pages 100–111.

Steenbergen-Hu, S. and Cooper, H. (2013). A meta-analysis of the effectiveness of intelligent tutoring systems on k-12 students' mathematical learning. *Journal of Educational Psychology*, 105(4):970–987.

Surdeanu, M. and Turmo, J. (2005). Semantic role labeling using complete semantic analysis. In *Proceedings of CoNLL-2005 Shared Task*, pages 221–224.

Taylor, P., Black, A. W., and Caley, R. (1998). The architecture of the festival speech synthesis. In *The Proceedings of the 3rd ESCA Workshop in Speech Synthesis*, pages 147–151.

Topping, K. (1996). The effectiveness of peer tutoring in further and higher education: A typology and review of the literature. *Higher Education*, 32(3):321–345.

VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, 46(4):197–221.

Ward, W. (1994). Extracting information in spontaneous speech. In *Proceedings of the Third International Conference on Spoken Language Processing, ICSLP-94*.

Ward, W., Cole, R., Bolanos, D., Buchenroth-Martin, C., Svirsky, E., Van Vurren, S., Weston, T., Zheng, J., and Becker, L. (2011). My Science Tutor: A conversational multimedia virtual tutor for elementary school science. *ACM Transactions on Speech and Language Processing*, 7(4).

Ward, W., Cole, R., Bolanos, D., Buchenroth-Martin, C., Svirsky, E., and Weston, T. (2013). My Science Tutor: A conversational multimedia virtual tutor. *Journal of Educational Psychology*, 105(4):1115–1125.

Wu, S. and Palmer, M. (2011). Semantic mapping using automatic word alignment and semantic role labeling. In *Proceedings of ACL Workshop on Syntax and Structure in Statistical Translation (SSST-5)*.

Yi, S., Loper, E., and Palmer, M. (2007a). Can semantic roles generalize across genres? In *Processings of the NAACL HLT*.

Yi, S., Loper, E., and Palmer, M. (2007b). Combining lexical resources: Mapping between PropBank and VerbNet. In *Proceedings of the Seventh International Workshop on Computational Semantics*.

Appendix A

Annotation Examples

"the brain helps you move the lungs let you breathe and the heart pumps blood"

Phoenix

Frame: DefiningAttribute
Class: the [Brain] (brain)
Attribute: [BrainAtt] (helps you move)

Frame: DefiningAttribute
Class: the [Lung] (lungs)
Attribute: [LungAtt] (let you [Breathe] (breathe))

Frame: DefiningAttribute
Class: the [Heart] (heart)
Attribute: [HeartAtt] (pumps [Blood] (blood))

PropBank
ClearSRL

11 2 UserId help help.01 —— 0:1-ARG0 3:2-ARG1 2:0-rel
11 4 UserId move move.01 —— 3:1-ARG0 5:1-ARG1 4:0-rel
11 7 UserId let let.01 —— 8:2-ARG1 7:0-rel
11 9 UserId breathe breathe.01 —— 8:1-ARG0 9:0-rel
11 13 UserId pump pump.01 —— 11:1-ARG0 14:1-ARG2 13:0-rel

Gold Standard

11 2 conger help help.01 —— 0:1-ARG0 2:0-rel 3:2-ARG1
11 4 conger move move.01 —— 3:1-ARG0 4:0-rel 5:1-ARG1
11 7 conger let let.ER —— 7:0-rel 8:2-ARG1
11 9 conger breathe breathe.01 —— 8:1-ARG0 9:0-rel
11 13 conger pump pump.01 —— 11:1-ARG0 13:0-rel 14:1-ARG2

"it ca n't breathe so it dies"

Phoenix

Frame: Causal
Cause: [Breathe] ([Negative] (ca n't) breathe)
Result: [Die] (dies)

PropBank
ClearSRL

58 3 UserId breathe breathe.01 —— 0:1-ARG0 1:0-ARGM-MOD 2:0-ARGM-NEG 3:0-rel
58 6 UserId die die.01 —— 5:1-ARG1 6:0-rel

Gold Standard

58 3 conger breathe breathe.01 —— 0:1-ARG0 1:0-ARGM-MOD 2:0-ARGM-NEG 3:0-rel 4:1-ARGM-PRD
58 6 conger die die.01 —— 5:1-ARG1 6:0-rel

"the tires make it go and the engine makes the car run"

Phoenix Annotation

Frame: Movement
Agent: the [CarPart] (tires)
Goal: [CarPartAtt] (make it go)

Frame: Movement
Agent: the [CarPart] (engine)
Theme: the [Car] (car)
Goal: run

PropBank Annotation
ClearSRL

149 2 UserId make make.02 —— 0:1-ARG0 3:2-ARG1 2:0-rel
149 4 UserId go go.02 —— 3:1-ARG1 4:0-rel
149 8 UserId make make.02 —— 6:1-ARG0 9:2-ARG1 8:0-rel
149 11 UserId run run.XX —— 11:0-rel

Gold Standard

149 2 conger make make.02 —— 0:1-ARG0 2:0-rel 3:2-ARG1
149 4 conger go go.01 —— 3:1-ARG1 4:0-rel
149 8 conger make make.02 —— 6:1-ARG0 8:0-rel 9:2-ARG11
149 11 conger run run.01 —— 9:1-ARG1 11:0-rel

"The nervous system (made of brain spinal cord and nerves) is the part of the body that detects a stimulus and sends a message for a reaction (if there is one)"

Phoenix Annotation

Frame: Class
Class: The [NervousSys] (nervous system)
Member: [Brain] (brain)
Member: [SpinalCord] (spinal cord)
Member: [Nerve] (nerves)

Frame: DefiningAttribute
Attribute: [NervousSysAtt] (detects a [Stimulus] (stimulus) and sends a message for a [Reaction] (reaction))

PropBank Annotation
ClearSRL

603 4 UserId make make.01 —— 0:1-ARG1 5:1-ARG2 4:0-rel
603 12 UserId be be.01 —— 0:2-ARG1 13:3-ARG2 12:0-rel
603 19 UserId detect detect.01 —— 13:2-ARG0 20:1-ARG1 18:1-R-ARG0 19:0-rel
603 23 UserId send send.01 —— 13:2-ARG0 24:1-ARG1 26:1-ARG2 18:1-R-ARG0 23:0-rel
603 25 UserId message message.XX —— 25:0-rel
603 28 UserId react react.XX —— 28:0-rel
603 32 UserId be be.02 —— 33:1-ARG1 32:0-rel

Gold Standard

603 4 conger make make.01 —— 0:1-ARG1 4:0-rel 5:1-ARG2
603 12 conger be be.01 —— 0:2-ARG1 12:0-rel 13:3-ARG2
603 19 conger detect detect.01 —— 18:1*13:2-ARG0 19:0-rel 20:1-ARG1
603 23 conger send send.01 —— 18:1*13:2-ARG0 23:0-rel 24:1-ARG1 26:1-ARGM-PRP
603 25 conger message message.YY —— 25:0-rel
603 28 conger react react.01 —— 28:0-rel 29:1-ARGM-ADV
603 32 conger be be.02 —— 32:0-rel 33:1-ARG1

"Batteries in series must have opposite ends touching for electricity to flow"

Phoenix

Frame: Requirement
Agent: [SeriesCircuit] ([Battery] (Batteries) in series)
Theme: [Orientation] ([_Opposite] (opposite) [Terminal] (ends) touching)
Goal: [ElectricalFlow] ([Electricity] (electricity) to flow)

PropBank
ClearSRL

632 4 UserId have have.03 —— 0:2-ARG0 5:2-ARG1 3:0-ARGM-MOD 4:0-rel
632 6 UserId end end.XX —— 6:0-rel
632 7 UserId touch touch.04 —— 5:1-ARG0 7:0-rel
632 11 UserId flow flow.01 —— 9:1-ARG1 11:0-rel

Gold Standard

632 4 conger have have.03 —— 0:2-ARG0 3:0-ARGM-MOD 4:0-rel 5:2-ARG1
632 6 conger end end.YY —— 6:0-rel
632 7 conger touch touch.01 —— 5:1-ARG0 7:0-rel 8:1-ARGM-PRP
10:2-ARGM-PRP
632 11 conger flow flow.01 —— 9:1-ARG1 11:0-rel