2011

Emotional Feedback Generation for Physical Therapy

Matthew Kusner
Macalester College

Honors Project

Macalester College

Spring 2011

Title: Emotional Feedback Generation for Physical Therapy

Author: Matthew Kusner

EMOTIONAL FEEDBACK GENERATION FOR PHYSICAL THERAPY

MATTHEW KUSNER
Macalester College
Computer Science

Advisor: Susan Fox
Second Reader: Shilad Sen
Third Reader: Daniel Kaplan

May 2, 2011

Abstract

Consider a personal assistant who lives with physical therapy patients, reminding them to perform various exercises and giving patients feedback about how well they have enacted such exercises. The assistant has learned, through the patient's interactions with his or her physical therapist, to personalize its commands and responses to the patient. Having such an assistant would decrease the time it takes for patients to recover via a more regimented exercise routine and would give physical therapists a much more detailed account of the activities of their patients. Here I demonstrate a system that, given an encoding of a patient's exercise, generates emotionally-manipulated sentence feedback about how to better perform the exercise. In essence, the system is composed of two primary modules: a reasoning module and a natural language generation module. For my results, I present a technique to generating instances of emotionally-altered exercise feedback on a stern-nurturing axis. I show a method for relating feedback words to deficiencies in exercise performance. Finally, I demonstrate a proof of concept technique for generating feedback given real-time skeleton mapping software. In future work, the system will be integrated into a physical therapy social avatar, which will interact as an assistant to patients within their home.

Contents

Part I

Introduction and Background

Chapter 1

Introduction

Automated Reminder systems serve a multiplicity of purposes. Consider such simple examples as alarm clocks, seat-belt, and meeting reminder systems. More complex systems exist as well, such as various GPS systems and home security systems. Such systems are invaluable for everyday organization.

Reminder systems most often, however, tend to be very annoying, primarily because of how repetitive and general they are. Certainly alarm clock and seat-belt reminder systems exhibit this. GPS systems repeat a specific direction as a driver approaches a particular waypoint in the route, whereas a good human navigator might direct the driver's actions based on specific approaching landmarks or other relevant information. Systems that interact with patients might benefit from a diversity and specificity in responses.

Computers are specifically well-suited for the task of patient interaction. Compared to humans which might vary their feedback on account of running into traffic on the way to meet with a patients, computer systems are excellent candidates for instructing patients based on their ability to provide clear, reliable outputs to particular inputs. Indeed, it is crucial that patients receive objective, consistent feedback about how they are to improve not only for the patient's sake, but also so that a practitioner can evaluate whether such improvement measures are adequate. Computerized feedback systems then should provide accurate, pinpointed responses. There is evidence that certain patients respond better to instructions stated in a nurturing manner, while other patients respond better to stern instructions. This suggests that these systems should have the ability to modify the emotion of their instruction.

More generally, systems that allow for specific interactions with users have possible application wherever human interaction occurs. More concretely, fields which involve the flow of knowledge in one direction could benefit from these systems. Similar to patient interaction systems, any sort of activity that involves coaching could make use of such systems to supplement the advice of a

human coach. Systems of this kind would find use within the market of customer service. The immense range of application of such systems motivates this work on them.

It is often the case that patients that require physical therapy do not devote sufficient time to performing physical therapy exercises within the home setting. Various research has shown that continuing exercise apart from scheduled physical therapy visits helps patients recover in a shorter amount of time [16]. In order to promote physical therapy exercise within the home, I am working with Professor Reid Simmons at Carnegie Mellon University to develop a physical therapy coach social avatar. The avatar is designed to interact with patients as a physical therapist would. As such, it should be able to instruct patients through providing them with feedback about their exercises.

I present here a system that provides pinpointed instructive feedback and is able to adjust sentence structure to respond in a more stern or nurturing manner. In doing so, it responds in a way that takes into account what it thinks the user already knows and/or should be reminded about, thereby avoiding repetition in its responses. As an example, if a physical therapist instructs a female patient to bend her knee 90 degrees and the patient bends her knee 70 degrees, the patient feedback system indicates to the patient that she should bend her knee further. The emotion of the feedback will fall along a stern/nurturing axis based on information about the patient's preferred instructive emotion, gotten from an external system.

In order to encode information about exercises and appropriate responses, the system makes use of an ontology. An ontology is a text representation of a specific area of knowledge. Similar to object-oriented programming, there are classes to describe concepts, objects to describe instantiations of concepts, properties for particular traits of objects, and relations to connect concepts to each other, among other things. The OWL2 [12] ontology I created for this project describes the relationship between physical therapy exercises and feedback appropriate for such exercises.

To construct this ontology I use the Stanford-based semantic editor Protege. Protege provides a graphical interface to create and edit ontology information [10]. In order to extract information from the ontology I used the Pellet semantic reasoner [13]. Semantic reasoners make inferences based on the relationships created in an ontology. For example, one might use a semantic reasoner to see if an object *Gerald* is of the class *Cow*. Here I use Pellet to extract words for a feedback sentence about a certain exercise that a patient performed.

Additionally, I make use of a custom natural language generation software to convert the Pellet output into sentences. The custom software allows for the manipulation of emotion along a stern/nurturing axis via word substitution and syntactic rearrangement.

Finally, I use Microsoft's Xbox Kinect, a combination RGB camera and infrared depth sensing device, as a proof-of-concept tool for capturing information directly from exercise performance. OpenNI, a non-profit organization that promotes the use various human interaction devices, recently released software that does real-time skeleton tracking using the Kinect. I modify this software to record a pair of initial and final positions of the right arm for an arm-bending exercise. This data is transformed into information that can be processed by the reasoning module and subsequently by the natural language generation module.

A high level overview of the entire system is shown in Figure 1.1. The blue rectangles represent primary modules in the system. The orange hexagons signify module components that I have not created but am borrowing from other sources. Indeed, within the reasoning module is the semantic editor (Protege) that constructs the ontology (OWL) along with the semantic reasoner (Pellet) which infers appropriate feedback for a particular exercise, whereas OpenNI does skeletal tracking based on sensor data from the Xbox Kinect. The yellow oval is meant to represent an initial result from the system while the green diamond signifies the final system output. The purple rounded rectangle represents an external input into the system from another unknown module.
With this in mind, I had three specific aims for the project:

1. Generate emotionally-accurate sentences along a stern/nurturing scale, and
2. Avoid repetition in sentence generation, and
3. Generate accurate feedback based on data from the Xbox Kinect.

For my results I was able to manipulate sentences using a stern/nurturing score assigned to generated sentences. I found out that the goal of emotional accuracy was at odds to a certain extent with the goal of repetition avoidance. Therefore, I sacrificed emotional specificity in order to allow the possible number of generated sentences to be greater. Finally, I found data from the Kinect to be surprisingly accurate in judging upper body positions. For this reason I looked at sentence generation for arm-bending as opposed to leg bending.

The next chapter describes the current state of the fields of artificial intelligence used in this project. The third chapter goes into detail about the implementation of the project. In the fourth chapter I describe the sentence-generation results while the last two chapters are devoted to results discussion and future work.

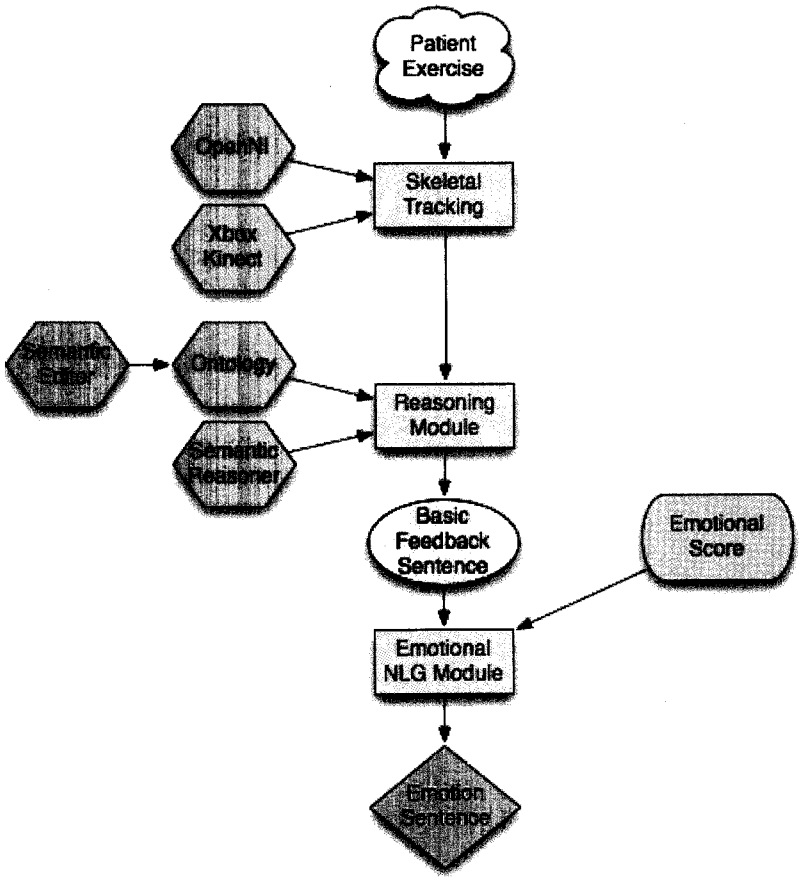

Figure 1.1: Physical Therapy Feedback System Overview

Chapter 2

Background

This project takes inspiration from four particular fields of artificial intelligence:

1. Knowledge representation
2. Reasoning
3. Natural Language Generation
4. Image processing

In this section I introduce key entities, concepts, and definitions relevant to each field. The primary goal is to set the stage for a discussion about how such systems come together for the physical therapy feedback system.

2.1 Knowledge representation

Knowledge representation is concerned with how to give computer systems access to information that is often complex and interconnected. How can one represent human knowledge in a way for computers to make the same sort of inferences that humans do? With this in mind, researchers in the field formulated the notion of an ontology. The ontology stores interconnected sets of information.

Many modern ontologies are defined using XML. XML is a language constructed for the very purpose of being machine-readable [21], XML allows one to embed categories within others hierarchically. Thus XML is a natural choice to represent the complex, hierarchical information found in ontologies. Because of this, those working on knowledge representation developed semantic editors for creating ontologies without having to organize the XML code of the ontology. This served to decrease the time it takes to construct an ontology while decreasing the potential for errors in ontology construction. Finally, in order for a computer to be able to use the ontology to make claims, the semantic reasoner was developed. Here I describe each of the three concepts in detail.

2.1.1 Ontology

An ontology is a logical framework used to describe pieces of information and the relationships between such pieces. For instance, ontologies are used for integrating spacecraft mission monitoring systems, for constructing a unified knowledge framework allowing researchers in manipulation, planning, and machine learning to make use of results in each others fields, or for converting semi-conductor fabrication models to different formats [18]. The ontology for this project is used for reasoning about what to say to patients given an exercise they performed.

For the most part, ontologies consist of classes, individuals, properties, relations, axioms and restrictions.

A class is most often a generalized object or a type. For example, one might have a *Tree* class, which is meant to describe the concept of a tree. Individuals are particular instantiations of classes. Continuing with the example above one might have an individual, *Tree_In_Backyard*, describing a tree in a particular person's backyard.

Properties are particular qualities of classes and individuals. One might have a property *diseased*, which is boolean and is meant to represent whether a particular tree has an arbitrary disease.

Relations describe how particular classes (as well as particular individuals) and properties are related to other classes and individuals, respectively. For example, there might exist a class *Birch_Tree*, which is a subclass of *Tree*. There exists a relation between *Birch_Tree* and *Tree*: that *Birch_Tree* is a *Tree*. Similarly, with properties one may have a boolean property *birch_canker* which is a subproperty of *diseased*.

Axioms encode particular things that are true of an ontology. Stated another way, they make statements about the ontology that may be equated to background knowledge of that ontology. As an example, *Birch_Tree* may have a boolean property called *bronze_birch_borer*. One might create an axiom that states that *Birch_Tree* can only be associated with *bronze_birch_borer*, so that the property never gets accidentally grouped with another *Tree* subclass.

Restrictions create unnamed classes that consist of individuals related to each other in a particular way. The most common restrictions are property restrictions, which create classes of individuals, each of which have a certain property. One might create a restriction for a boolean property *american_chestnut_blight*, which would group such individuals with this property such as *Tree_In_Backyard* and *Tree_In_Frontyard*.

What good is this ontology? Assume one adds another individual *Healthy_Birch_Tree* which has the properties *birch_canker* and *bronze_birch_borer*, each with

values of FALSE. Assume further that there is a set of individuals *Park_Tree1*, *Park_Tree2*, and *Park_Tree3*. Finally, assume that there is another class called *Tree_Care*, whose individual consist of different tree treatment products. Each *Tree_Care* individual has properties which are the tree diseases it can treat. At the end of each day we see if the actual trees represented by *Park_Tree1* and *Park_Tree2* have either birch canker or bronze birch borer. If we find out that both trees have developed birch cankers. In the ontology we set the property *birch_canker* to TRUE. We can then create a restriction using *birch_canker*. Once created, a semantic reasoner can be used to grab the individuals of *Tree_Care* that occur in the unnamed restriction class to get a set of appropriate treatment methods. One can imagine further restricting the set based on other properties to end up with the optimal treatment method. The primary idea of an ontology is to encode the hierarchy of information inherent in particular sets of data.

The Web Ontology Language (OWL) is a World Wide Web Consortium (W3C)-endorsed set of languages for constructing ontologies. OWL's original purpose was for use in building the Semantic Web, described as a state of the web in which machine agents could usefully extract information for human users [3]. This is part of why OWL is motivated to use XML. One of XML's stated design goals is that "XML shall be straightforwardly usable over the Internet" [20]. The majority of modern web browsers support XML, making it a natural choice. In October 2009, W3C announced the release of OWL2 which came with a number of logic-based improvements over OWL, released in 2004 [12].

Ontologies are particularly well-suited for medicine, where there is often extensive information on patients and treatments. Eccher and Ferro use an ontology for describing a variety of medical therapies. Their hope is to make use of the ontology to discover errors in data entry and support automated data analysis [5].

2.1.2 Protege

Protege is a semantic editor used for manipulating a number of different ontology languages [10]. The majority of ontologies are constructed using XML and are often tedious to create and manipulate. Semantic editors such as Protege simplify ontology modification by graphically representing pieces of information as well as the relationships in an ontology. This is somewhat similar to how an integrated development environment simplifies coding in various programming languages (e.g. Eclipse for Java). Figure 2.1 shows Protege's *Classes* window. At the left is the *Class Browser* which allows one to browse the organization of one's ontology. To the right of this is the *Class Editor*. Upon selecting a class in the *Class Browser* one can create individuals of that particular class as well as associate properties with the class. Additionally, Protege lets one visualize the class hierarchy of their ontology by way of a graph-like representation in which classes are nodes and classes are connected to subclasses via directed

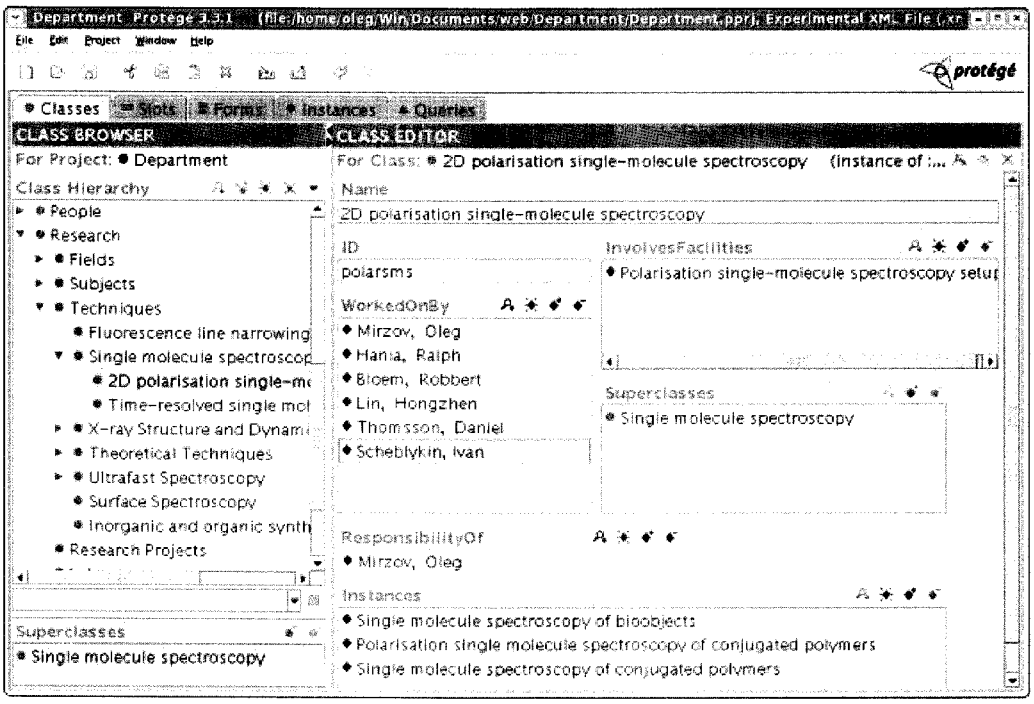

Figure 2.1: Protege semantic editor

edges. Built into Protege are various reasoners that check the consistency of the ontology as well as make simple inferences. Consistency checking is vital to ensure that a semantic reasoner makes consistent inferences. An inconsistent ontology can lead a reasoner to make conclusions that might disagree with certain portions of the ontology. An inconsistency is often quite difficult to find in an ontology, which makes Protege all the more invaluable.

I used Protege to make the OWL2 ontology for this project. For more information on the exercise ontology see Section 3.2.

Most of the time, the very purpose of constructing an ontology is infer things about it. Once a language ontology is made one can use a semantic reasoner to extract information about the ontology.

2.1.3 Semantic Reasoner

Semantic reasoners, at a high level, take as input a particular ontology and output various logical inferences. That is to say, they function as an expert which has knowledge of the details of one's ontology. More specifically, they

take the information encoded by the classes, individuals, properties, etc... of an ontology and allow one to ask questions of the ontology such as, "What are the subclasses of this class?" or "What are all of the individuals of this class?". A more complex example would be if, given an individual such as *Healthy_Birch_Tree*, one wanted to determine the set of tree individuals which have none of the same properties as *Healthy_Birch_Tree* within the tree ontology.

The majority of semantic reasoners use deduction to make inferences via forward chaining and backward chaining. Consider the following set of known syllogisms:

1. **If** it is raining - **Then** it is wet outside
2. **If** it is raining fire - **Then** then it is hot outside
3. **If** it is wet outside - **Then** then Greg is angry
4. **If** it is hot outside - **Then** then Mark is angry

Imagine that one wants to determine if Mark or Greg is angry given that it is raining. In forward chaining one would begin by scanning through the set of syllogisms and selecting the first rule because the antecedent is known to be true. The consequent we discovered to be true is then added to the knowledge set, that it is wet outside. Scanning the set of syllogisms again with this new knowledge set the third syllogism is selected. One now has the knowledge one wants, that Greg is angry. Note that forward chaining also halts if there is no antecedent for which its consequent is not already known [7].

In backward chaining one would begin by scanning through the set of syllogisms and selecting the third and fourth because they pertain to what we are interested in, finding out if Mark or Greg is angry. One then looks for consequents that talk about whether it is hot or wet outside. The first two syllogisms do this, so they are selected. Then, because the antecedent of the first syllogism is true it is concluded that Greg is angry. Backward chaining ends if a known antecedent is found such as this or if there are no consequents that say "it is wet outside" or "it is hot outside" [2].

Pellet is a open-source Java semantic reasoner which makes inferences on OWL and OWL2 ontologies [13] via forward chaining. I make use of Pellet currently to construct inferences over the physical therapy ontology. See Section 3.3 for more information about the reasoning used in the reasoning module.

2.2 Natural Language Generation

The field of Natural Language Generation is a subfield of Natural Language Processing concerned with generating human-like sentences from particular sentence representations. Typically, natural language generation systems consist of at least three stages:

1. Content determination and text planning
2. Sentence planning
   (a) Sentence aggregation
   (b) Lexicalization
3. Realization

[15]. The content determination and text planning stage is described as "the process of deciding what information should be communicated in text" by Reiter and Dale [15]. Content determination is where the natural language generation system decides upon and organizes the semantic content to be transmitted by the system. Reiter and Dale describe the output of content determination as sets of messages that contain structured information about what needs to be conveyed by output sentences. Text planning then involves arranging the messages in the proper order. This often occurs via a tree structure which has these messages for leaves and conceptual message groupings as internal nodes.

Sentence planning consists of two substeps: sentence aggregation and lexicalization. Sentence aggregation takes the aforementioned tree structure and produces another tree with leaves consisting of grouped messages. Lexicalization describes the process of choosing the words which will represent grouped messages.

Lastly, realization forms grammatically sound sentences given the output of the lexicalization process. Specifically, a realizer must conjugate verbs, ensure agreement occurs properly, insert pronouns if necessary (e.g. for sentences in which the subject refers to him or herself), etc...

Early examples of natural language generation systems were CENTRIFUSER for summarizing documents of a similar knowledge domain [9]. The motivation for this system is to help web searchers decide whether or not to read a document by generating tags or features for documents after analyzing them. For instance, if there is a document about black widow spiders CENTRIFUSER might generate the feature 'poisonous' to help group the document with other similar ones. A system called Autotext generated text given weather information from a meteorological database [19]. For example, if in a weather database it was predicted to be sunny with a thirty percent chance of rain, Autotext would generate a set of sentences like: "Today it is sunny outside. There is a thirty percent chance of rain as well." COMMENTATOR for description of various simple computerized scenes [17]. Specifically, given input about objects on a screen COMMENTATOR produced sentences about facts it inferred about such objects, such as their distances and orientations. As an example, say that a cat and a dog were known to be at coordinates $(50, 50)$ and $(50, 55)$ on a screen and they are both facing right. COMMENTATOR would produce a sentences to the effect of, "The cat and the dog are five units away." and "The cat and

the dog are facing right." A recent use of natural language generation systems is to generate jokes. In [8], they use natural language to generate Tom Swiftys. Example Tom Swiftys are, "'I am not a girl', said Tom boyishly" or "'this work is dull', said Tom pointedly" [8].

Emotion pervades our everyday use of language. It affects our tone of voice and how we structure our sentences. I am particularly interested in the latter capacity. Consider the two sentences, "Now, bend your leg, is that clear?" and "Could you please bend your leg for me?". Both convey the same information, yet the first sentence is sterner, more imperative, while the second is more gentle and nurturing.

Research in natural language processing has only recently began to consider the link between emotion and language. Mohammed and Turney recently developed an emotional lexicon based on an extensive emotion survey [11]. The survey, taken by roughly one thousand people, asked participants to describe how strongly a particular word evoked a set of ten emotions including joy, fear, trust, etc... From this data, the researchers, for a particular word, correlated each emotion's most popular intensity with that word (e.g., word *a* strongly negative, does not evoke joy).

In addition, Das and Bandyopadhyay have developed a classification system for tagging emotion in Bengali blogs [4]. The idea is to extract various textual features in blogs and use machine learning to identify the particular emotion conveyed in a given blog.

There has been some preliminary research in working to incorporate emotion into natural language generation systems. Fleischman and Hovy developed a framework for varying emotion in a natural language generation system for the Mission Rehearsal Exercise (MRE) virtual training environment [6]. The MRE virtual training environment is a virtual reality space designed to test how well army cadets operate in challenging situations. The natural language generation system developed by [6] is used by virtual actors in the MRE. Emotion in the MRE natural language generation system is introduced by the emotional state of a particular actor at a given time. Emotional sentences are scored by a human annotator. Specifically, they score a verb's effect on the creation of positive/negative feelings for the object and subject in a sentence. Sentences are then chosen for an actor based on their fit with the actor's emotional state. Compared to Fleischman and Hovy I am looking to develop a system that is based on the emotional content of words independent of their role within a sentence.

I have constructed a custom natural language generation system designed specifically for instructive phrases, implemented in the Python programming language. The system generates sentences with varying levels of stern or nurturing emotion, starting with an appropriate input sentence. For instance, the system

may generate sentences ranging emotionally from "Please keep your knees together. Thank you." to "Bend your knee further, alright?" and ranging syntactically from "Keep your body straight" to "Your body should be kept straight". Section 3.4 describes the software in greater detail.

2.3 Image Processing

At the front end of our system, we want to be able to capture the movement of a patient to assess whether or not any feedback is necessary about the patient's exercises.

Currently, there are a variety of methods of doing this. In movies, it is common to attach markers onto an actor and track the movements of such markers as the actor moves in space. For capturing the movement of a specific body-part one could attach an accelerometer, a sensor that records its movement and orientation, onto the body-part of interest.

Ultimately, we chose to use skeleton-tracking software developed for Microsoft's Xbox Kinect. The Kinect consists of an RGB camera and an IR emitter and sensor. Originally intended for reading human movement to control video games, the Kinect has been made to perform a variety of tasks from playing a virtual piano to turning a television into a touchscreen (http://openkinect.org/). To get information directly from exercises I used skeleton-tracking software provided by OpenNI (http://openni.org/) for the Kinect. The software keeps track of elbow, hand, shoulder, waist, torso, knee, and foot positions once properly initialized.

I have customized the software so that the elbow and hand positions of the right arm can be recorded at the beginning and end of a arm-bending exercise using two different keystrokes. The pairs of coordinates for a patient's elbow and hand are written to a file which is analyzed by a Java routine and converted into a arm-bend angle. The routine then sends the information to the reasoning module to generate feedback.

As mentioned before, this is purely a proof-of-concept approach, however, the majority of physical therapy exercises could be analyzed in a similar fashion.

Part II

System Architecture

Chapter 3

System Methods

I present a three-part system, consisting of a skeleton tracking module, a reasoning module and an natural language generation module. The skeleton tracking module is situated at the beginning of the entire system and consists of the Xbox Kinect and the OpenNI skeleton tracking software. The module takes skeleton data and outputs exercise information. The reasoning module comes after this and consists primarily of the OWL ontology and the Pellet semantic reasoner. It takes as input patient exercise information and outputs a sentence I will call an *initial phrase*, appropriate for exercise feedback. This phrase is then passed on to the natural language generation module. This module takes the *initial phrase* and manipulates it to construct a sentence that is most representative of an input emotional score.

For example, say that a patient performs a leg curl exercise and bends their leg 50 degrees instead of the desired 80 degrees. This information gets sent to the reasoning module which then formulates the sentence "Bend your leg further". This initial phrase is sent to the natural language generation module along with an emotional score of 4. The natural language generation module then produces the sentence "If you could bend your leg further. Thanks."

To begin, I discuss first how I use the Xbox Kinect to get data about patient exercises. I go on to describe how the exercise ontology is constructed and then how the system goes about reasoning over the ontology. I end with a description of the natural language generation system, describing how emotion is used in final sentence selection.

3.1 Image Processing

The Kinect is particularly well-suited for distinguishing foreground elements from background elements. This has to do with the ability of the Kinect to produce a 3-dimensional map of what is in front of it, called a point cloud. A point cloud can be constructed from the IR emitted and detected by the Kinect.

Specifically, the IR emitter on the detect sweeps the environment in front of it with IR light. That light bounces off of the environment and deflects back into the IR detector. As the Kinect knows the amount of time between the emission of an IR light ray and its detection, it can calculate the distance that a point in space is from the Kinect. The skeleton tracking module of OpenNI, the software used by the Xbox to extract information from the Kinect, exploits this fact. The module tracks feet, knees, waist, torso, shoulders, elbows, and hands. Figure 3.1 shows a still frame from the skeleton tracking module. At any one

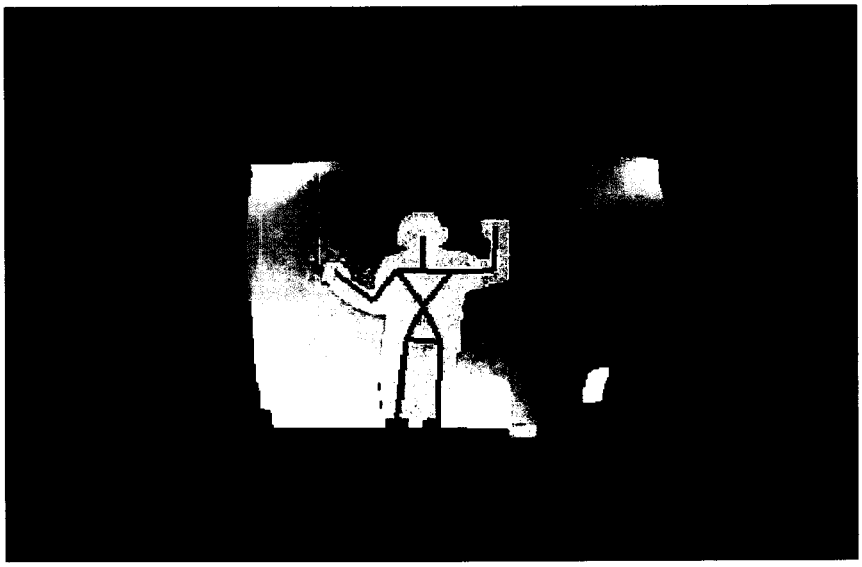

Figure 3.1: OpenNI skeleton tracking module

time the module draws the three dimensional coordinates of a mapped skeleton. I modified the source of the module to write the coordinates of the elbow and hand to a file when a set of keys are pressed.

I found that tracking knee-bending was much more prone to errors in the skeleton tracking software. Thus, I am interested in tracking the angle that a patient bends his or her arm. This was accomplished by recording an initial position (key "i") and final knee position (key "j") for the exercise. From the set of recorded coordinates I calculated the bent angle using the law of cosines. Specifically, I started by averaging the initial and final elbow coordinates. This averaged coordinate, along with the two hand coordinates, formed a triangle for which all of the side lengths are known. This allowed us to find the angle of any one of the vertices using the law of cosines. Once found, this angle is sent to the reasoning module to create a feedback sentence that is sent to the natural language generation module for final output.

3.2 Ontology Construction

Within the system's ontology I have two primary classes that hold all of the reasoning system's information: an *Exercise* class and an *Utterance* class. The *Exercise* class generally encodes the information which is necessary for a physical therapist to evaluate whether or not a patient performed an exercise correctly.

More specifically, the *Exercise* class consists of seven subclasses, each representing a specific physical therapy exercise: *HamstringStretch*, *HamstringStretchBand*, *HipAbduction*, *KneeExtension*, *LegCurl*, *Squats*, and *ThighStretch*. Each *Exercise* has a set of properties associated with it. Such properties are representative of the most important pieces of the exercise. For example, the *Exercise KneeExtension* has the properties *exerciseDuration*, *kneeTwist*, and *iterations*, which represent the number of seconds for which an exercise iteration was performed, whether the patient twisted his or her knee while performing the exercise, and the number of iterations of the exercise that were performed, respectively. Figure 3.2 provides a sense of the organization of a subset of the Exercise class. The ovals represent classes with the dark blue oval signifying the exercise superclass and the light blue ovals signifying particular exercise subclasses. The green rectangles are ontology individuals and the orange hexagons are properties of those individuals. Finally, the pink diamonds are the values of each property. Additionally, an individual is instantiated for each *Exercise* subclass that represents the ideal performance of that exercise. I will call an individual of this sort *GI*, for good individual, see *GoodKneeExtension* in Figure 3.2. When a patient performs an exercise, an external system will take in sensor information about the performance of the exercise and construct an individual of that exercise having properties that describe the sensor information. Let us call this individual *PI*, for patient individual, *PatientExercise* in Figure 3.2.

The *Utterance* class encapsulates the words of the *initial phrase* that will be sent to the natural language generation module. More specifically, there are four *Utterance* subclasses: *Verb*, *Adverb*, *Determiner*, and *Noun*. In this way I organize the potential feedback words based on their part of speech. Different from *Exercise* classes the *Utterance* part of speech subclasses do not have a specific set of associated properties. Instead, for each part of speech subclass I create a set of individuals. Each individual represents a word that may be used within the initial feedback phrase. Additionally, each one of these individuals has a set of exercise properties. Exercise properties are associated with individuals if the name of the individual would form part of an appropriate *initial phrase* about that property. For example, the individual *Bend* has the exercise property *bentAngle*. In the current implementation of the *Utterance* class, to prevent sentence ambiguity, there is exactly one individual in each of the four subclasses that has a particular property. Figure 3.3 shows a subset of the *Utterance* class.

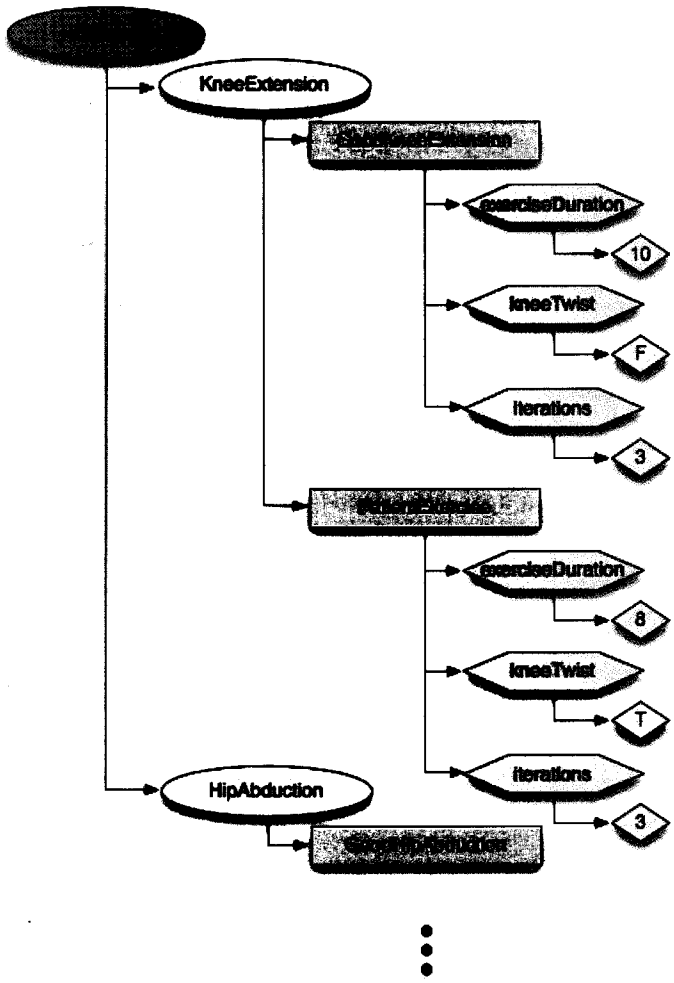

Figure 3.2: Part of the Exercise class

3.3 Reasoning

With the *Exercise* and *Utterance* classes established, the reasoning module now has a way to construct feedback for a patient. First, it compares the values of the properties of the PI with the GI for the same *Exercise* subclass. For example, imagine that $PI = PatientExercise$, which is an individual of *LegCurl*. The module would compare $GI = GoodLegCurl$ with PI. Now, let us call the

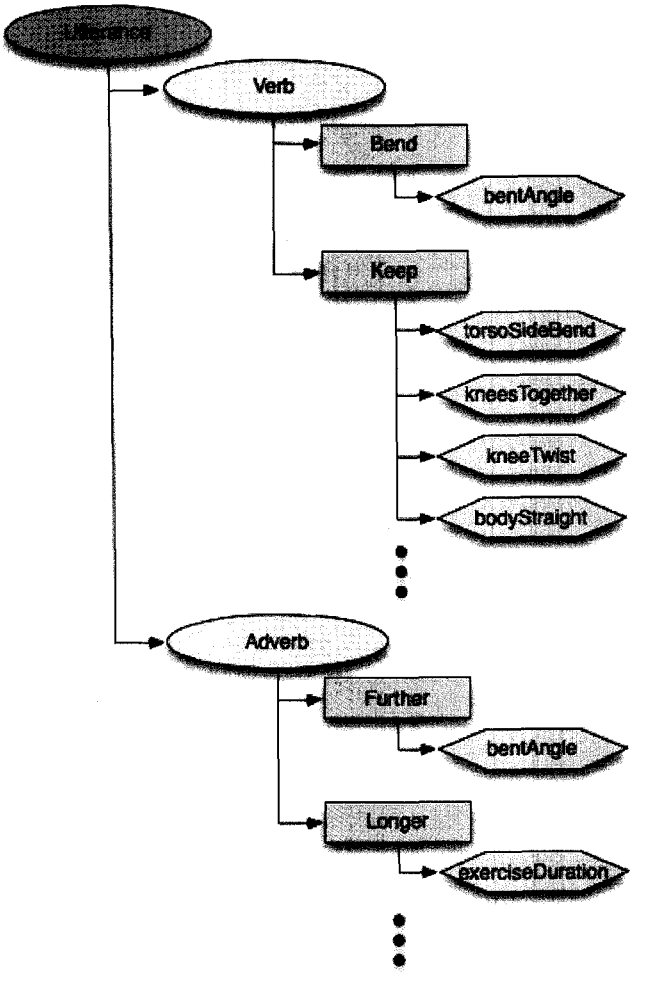

Figure 3.3: Part of the Utterance class

set of properties which were unequal between PI and GI, DP for deficient properties. Continuing the example, imagine that GI has the property *bentAngle* with value 90 and that PI has *bentAngle* with a value of 70, (see Figure 3.4). In order to produce a feedback sentence the module makes use of the Pellet reasoner to extract Utterance individuals from the ontology for each element $d \in DP$. The pseudocode below shows the routine for feedback sentence formation: In this example the *Utterance* individuals with property *bentAngle* are

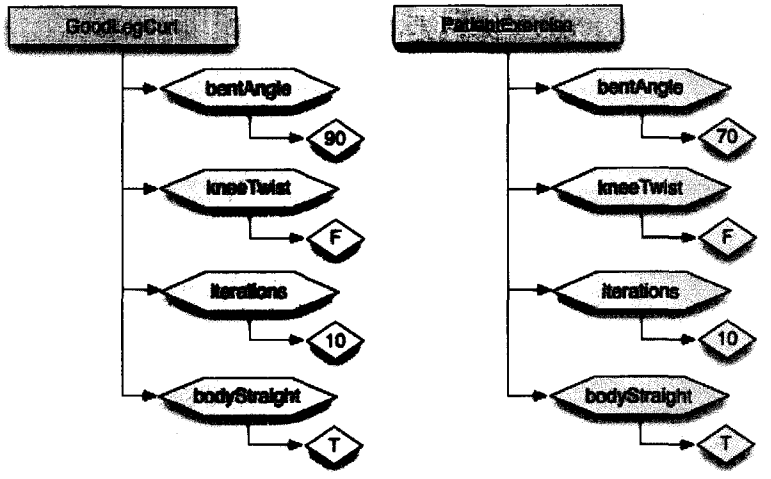

Figure 3.4: Comparison example

```
begin
  r = Pellet.reasoner.init(exerciseOntology)
  DP = r.getPropertiesOfDifferentValue(GI, PI)
  for i := 1 to length(DP) step 1
      (nouns, verbs, adverbs, dets) = r.getUtterancesWithProperty(DP[i])
      N = randomSelection(nouns)
      V = randomSelection(verbs)
      A = randomSelection(adverbs)
      D = randomSelection(dets)
      output(initial phrase = (V + "" + D + "" + N + "" + A))
  end
```

Figure 3.5: Pseudocode for feedback sentence construction

retrieved: *Bend* (of subclass *Verb*), *Further* (of subclass *Adverb*), *Your* (of subclass *Determiner*), and *Knee* (of subclass *Noun*), shown in Figure 3.6. The way I go about deciding whether a particular *Utterance* individual should have the property *bentAngle* is by looking through each exercise in the ontology and determining the ways in which an exercise could be performed incorrectly. For each way I consider sentences that would constitute appropriate feedback about what was done incorrectly. For example, the property *bentAngle* relates to individual *Bend* but not to individual *Separate* and to individual *Further* but not *Straight*. The goal is to create a set of individuals which could capture a range

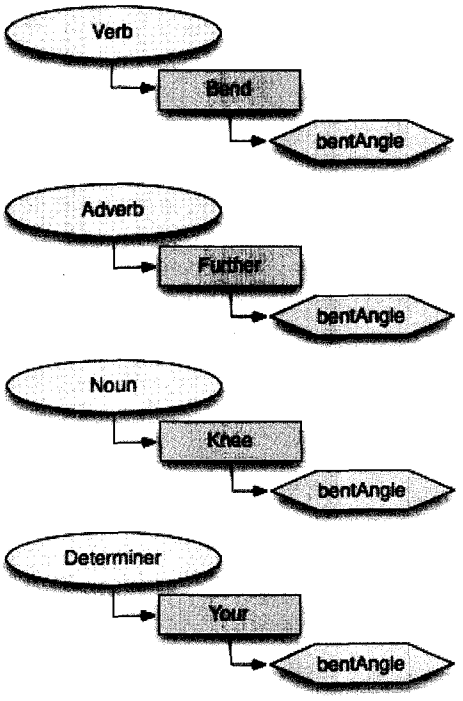

Figure 3.6: bentAngle comparison result

of different exercise types in case more are added to the ontology in the future. I make sure that there exists at least one *Verb*, *Determiner*, *Noun*, and *Adverb* individual for any exercise property in constructing the ontology. Once I have created such individuals with various exercise properties, the reasoning module extracts the *Utterance* individuals that are appropriate for a particular exercise deficiency.

Finally, the module constructs a feedback sentence (the *initial phrase*) by ordering the *Verb*, *Adverb*, *Determiner*, and *Noun* individuals from *UI* for an imperative sentence: "Bend your knee further". The individuals I have added to the *Utterance* class are such that this ordering always produces a coherent sentence. This *initial phrase* then is the input to the emotional natural language generation module.

3.4 Emotional Natural Language Generation

Motivation for developing an emotional natural language generation system stems from the desire to instruct patients in a variety of ways and to emotionally shape instructions to effect better exercises. On the whole, systems which reproduce the same output have the tendency to bore patients. Professor Simmons and I are trying to avoid a virtual therapist which has overly predictable responses. The idea in varying the sentences of the response is that we want to keep patients interested in the avatar. While we don't expect to be able to replicate the responses of an actual physical therapist, we do think we can construct painless interactions that are human-like.

Different from the classic natural language generation system schema, this natural language generation system consists of four primary phases: a parsing stage, a rule-checking stage, a rule-application stage, and a realization stage. In the parsing stage an initial appropriate sentence is taken and parsed into its parts of speech. We refer to a string that is the parts of speech of a sentence as that sentence's parsing. Additionally, these particular parts of speech are grouped into noun groups and verb groups. Essentially, a parsing is a generalization of a sentence. Next, in the rule-checking stage, a sentence's parsing is compared to a set of rules. Think of a rule as a mapping between one parsing and another parsing. The second parsing represents a sentence with the same meaning as the sentence represented by the first parsing. More specifically, say that the system is interested in the sentence is "Bend your knee further". This parsing for this sentence is $V\,NG\,R$, where V is a verb, NG is a noun group, and R is an adverb. Further imagine that the set of rules which convert sentence parsings to other parsings is shown below:

$$\begin{aligned} V\ C\ V\,NG &\rightarrow V\ C\ V \\ V\ NG\ R &\rightarrow NG\ V\ V\,VG \\ VG\ N &\rightarrow VG\ NG \end{aligned}$$

where VG is a verb group, N is a noun, and C is a conjuction. The rule checking stages collects the set of rules that have the same initial parsing as the sentence parsing, like the second rule shown above. This generates the sentence "Your knee should be bent further", given the original sentence. Once there is a set of rules to apply to the sentence, the module enters the rule-application stage, in which it generate a new set of parsings. Now, with the new set of parsings the module repeats the rule-checking stage and the rule-application stage, seeing if it can enlarge the size of the set of appropriate parsings. These stages are repeated until no new parsings can be generated. Finally, as in the classic natural language generation schema, there is a realization stage in which the module forms a sentence from each parsing. Here, it also checks for grammatical correctness.

Overall, this has the nice property of allowing for very specific syntactic manipulations that are common to instructive phrases. That is to say, the majority

of instructions are in the imperative mood. The rules that I have constructed are particularly tailored to the structure of imperative sentences. The manipulations are specific in the sense that they preserve the instruction inherent in an imperative statement whilst changing its overall sentence structure. Having the system take as input an imperative sentence situates it uniquely for physical therapy feedback.

Feedback should also convey emotion. Indeed emotion is integral to instruction in that it is able to convey more than a feedback sentence can do on its own. The physical therapy avatar will eventually be able to adjust the emotion of its responses in a human-like way to help patients exercise more efficiently.

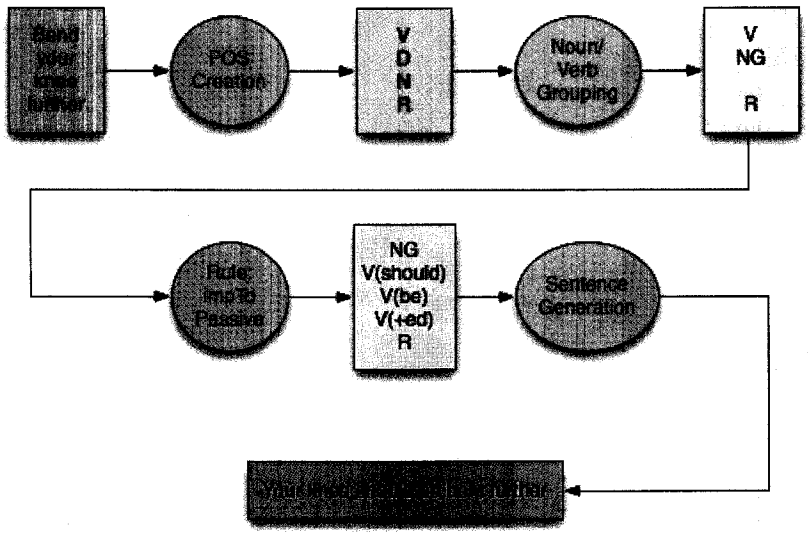

Figure 3.7: Rule-based generation system

3.4.1 Rules

A *rule* is a function which takes a parsing and converts it to another parsing and preserves the semantic content of the sentence that the input parsing represents. Rules are applied to sentences based on a sentence's *high-level object list*. A *low-level object list* for a sentence is a set of the parts of speech for each word in the sentence. The *high-level object list* for the sentence is constructed via partial noun and verb groupings on the *low-level object list*. For example,

Rule Name	Description	Example	Action[1]

impVerbSwap	Swaps the noun and verb of an imperative sentence and replaces the verb with a synonym of goes.	"Bring your body forward and stand up" → "Your body goes forward while standing up"	`^VG? NG? [RI] ->`
			`^NG? V(syn-goes) [RI]`
			OR
			`^VG?_1 NG? [RI] C VG?_2 ->`
			`^NG? V(syn-goes) [RI]`
			`C(while) VG?(+ing)_2`

whileRemove	Removes "while from the beginning of a sentence, conjugates the following verb appropriately, and changes the comma in the middle of the sentence to "and.	"While sitting, lift your leg" → "Sit and lift your leg"	`^C VG? .+ O(,) VG? ->` `^VG?(-ing) .+ C(and) VG?`
impToPassive	Converts an imperative sentence to a passive sentence.	"Bend your knee further" → "Your knee should be bent further"	`^VG? NG ->` `^NG V(should) V(be) VG?(+ed)`

Table 3.1: Examples of three different rules

for the sentence "Bend your knee further", the *low-level object list* is [Verb, Determiner, Noun, Adverb] or just $[V_1, D_1, N_1, R_1]$. The *high-level object list* is [Verb, Noun Group, Adverb] = $[V_1, NG_1, R_1]$. Noun groups are constructed by looking for any number of adjectives, determiners, conjunctions, numbers, and pronouns before a noun and grouping these objects with the noun. Essentially, any words that describe the noun occur within the noun's noun group. Verb groups are constructed by looking neighboring adverbs that further describe the verb. Effectively, verb and noun groups are superclasses, which increase the generality of the rules.

Applying the rule impToPassive converts the *high-level object list* into another one: $[NG_1, V(\text{should}), V(\text{be}), V(+\text{ed}), R_1]$. The subscripts indicate how certain parts of speech in the initial parsing are related to the parts of speech in the parsing generated by the rule. See Table 3.1 for examples of other rules implemented by the natural language generation module. The natural language generation module creates a mirrored structure which called a parsing. A parsing is a string which, similar to a sentence's object list, represents the parts of speech of the words in that particular sentence. As well, define a sentence's

[1]The actions are written in a modification of Python's regular expression format. The parentheses after a part of speech or a noun or verb group indicates either a necessary word (if in double quotes), a synonym of a necessary word (if word preceded by syn-), or a modification of a word (if a + or exists).

initial parsing and a sentence's *final parsing* as strings representative of the sentence's *low-level object list* and *high-level object list*, respectively. A sentence's parsing is what is used to see if a rule is appropriate to apply to a sentence. This is done via regular expression checking, shown in Table 3.1. This dual structure provides the flexibility to organize data without sacrificing speed for rule checking.

The system consists of a set of fourteen rules for manipulating imperative sentences and one rule for manipulating questions. Table 3.1 presents a sample of rules, a description about what each rule does, an example, and the change in the sentence's *final parsing* upon application of the rule.

3.4.2 Emotional Scoring

Each rule has an emotional score associated with it that falls on a nurturing/stern scale, -5 for very stern and 5 for very nurturing, based on the emotion the generated sentence will convey. This score is termed a sentence's *rule score*. I created each *rule score* based on my intuition of how each rule changes the nuturing/stern content of any sentence. I amended a portion of these *rule scores* based on significant results from an emotional sentence scoring survey (see Section 4.1).

Specifically, I created a survey that included 51 generated sentences from the natural language generation module. Subjects were asked to compare the stern/nurturing content of the previous sentence relative to that of the sentence in a given question (50 questions in all). Figure 3.8 shows a sample question of the survey. Individual questions within the survey were constructed to be orthogo-

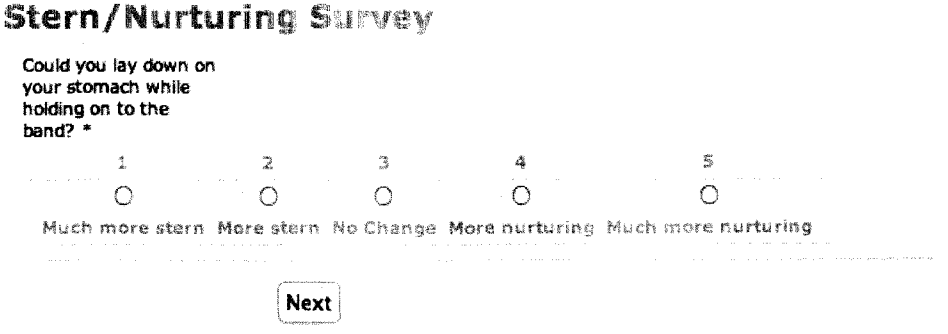

Figure 3.8: Emotional Scoring Survey

nal to each other by only varying one part of a command at a time (e.g., adding "Please" to the beginning of a sentence or taking away "is that clear?" at the end of a sentence). 13 survey responses were received. I manually created two tables

describing the *state* of a generated sentence (e.g., if a "Could you" started the sentence) and describing the change in state or $\Delta state$ of a generated sentence (e.g., if a "Now you will" was added to the start of a sentence). I composed these tables with the survey data itself to arrive at the final dataset.

Using R [14], I developed a set of 5 models on the complete dataset with the help of Professor Danny Kaplan. The models are (using R's linear model notation):

1. $m1 = surveyScore \sim state_1 + state_2 + ... + state_k$
2. $m2 = \Delta surveyScore \sim state_1 + state_2 + ... + state_k$
3. $m3 = surveyScore \sim \Delta state_1 + \Delta state_2 + ... + \Delta state_k$
4. $m4 = \Delta surveyScore \sim \Delta state_1 + \Delta state_2 + ... + \Delta state_k$
5. $m5 = surveyScore \sim sigState_1 + sigState_2 + sigState_f + sigState_i * sigState_j$

where $sigState_l$ refers to a state with a significant p-value (after Bonferroni correction) and $a * b$ refers to an interaction between variables a and b. On the left side of the models are the independent variables. These are the variables manipulated by the survey. $state_i$ refers to a sentence after rule i has been applied to the *input phrase* gotten from the reasoning module. $\Delta state_i$ refers to a change in state of a sentence before and after a rule is applied. $surveyScore$ refers to the absolute survey score while $\Delta surveyScore$ represents a change in survey score. Thus each model asks a different question. $m1$ asks, "Does the state of a sentence directly affect its absolute score?" $m4$ asks, "Does a change in the state of a sentence affect a change its overall score?" $m2$ and $m3$ are quite similar. Finally, $m5$ asks "Do statistically significant states and an interaction between two such states directly affect the absolute score of a sentence?"

Models $m1$ through $m4$ found significant results. For example, the rule which prefixes "Could you" and suffixes "?" to a sentence was significant in $m1$ (p-value: 2.61e-15) and the rule which prefixes "If you could" to a sentence was significant in $m4$ (p-value: 3.63e-6). For more information on statistically significant rules see Section 4.1.

3.4.3 Sentence Selection

Given an emotional score, e, gotten from an external system the natural language generation module filters out all of the sentences within the interval $[e-\varepsilon, e+\varepsilon]$, where ε is a tuned threshold value. Call the filtered set of sentences S. If this is the first sentence to be provided by the natural language generation module, the module randomly selects an $s \in S$ for output.

3.4.4 Repetition Avoidance

Now, it may happen that the natural language generation module needs to repeat a feedback sentence. Without any intervention, if the reasoning module sends the same *initial phrase* to the natural language generation module and the same emotional score is requested, the same sentence will be generated. To combat this the natural language generation module has a "memory" of the sentences that it has already produced in the form of a first-in-first-out (FIFO) queue. Upon outputting a sentence, it is placed at the end of the queue.

To select a sentence, the natural language module proceeds as discussed in the previous section. However, instead of outputting a random s, a pairwise similarity is generated between the sentence that was most recently said r and the sentences in S. Using these similarity scores we construct the set of sentences A:

$$A = \min_{s \in S}(\frac{LCS(r,s)}{avg(r,s)}) \tag{3.1}$$

, where A consists of the same sentences in S, starting with the least similar sentence in S and increasing in similarity throughout. $LCS(x,y)$ gives the length of the longest common subsequence of whole words between sentences x and y and $avg(x,y)$ gives the average word length of x and y. Now, starting at the first sentence $w \in A$, the system checks if w has been said before. If not, it is outputted. If so, the system checks the next sentence in A and asks the same question. If it turns out that every sentence in A has been said before, the system outputs the sentence that was said least recently.

It is important to note that the previously said sentence could even be about another exercise, all that matters is that a sentence is generated that is syntactically different than the previously said sentence.

Chapter 4

Results

I look now at how well the original goals are met by the current system, observing: the accuracy of emotional sentence generation, the quality of repetition avoidance in the system, and the appropriateness of feedback based on data from the Kinect.

4.1 Survey Results

I found significant p-values in models $m1$ through $m4$, described in Section 3.4.2. Model $m1$ $=surveyScore \sim state_1 + state_2 + ... + state_k$ had four significant manipulations (after Bonferroni correction):

1. suffixing "is that clear?" (score: -1.887)
2. prefixing "Could you" and suffixing "?" (score: 2.490)
3. prefixing "If you could" (score: 2.346)
4. prefixing "I would like you to" (score: 1.968)

Thus, suffixing "is that clear?" onto a sentence gives that sentence a score of -1.887. Therefore, I changed each of the corresponding *rule scores* based on the above results. Model $m4 = \Delta surveyScore \sim \Delta state_1 + \Delta state_2 + ... + \Delta state_k$ had two significant variables:

1. prefixing "Could you" and suffixing "?" (score change: +2.031)
2. prefixing "If you could" (score change: +2.654)

Thus, if the previously generated sentence included prefixing "If you could" and the sentence to generate did not, the score of the sentence being generated should have 2.654 subtracted from it. As well, if this manipulation existed in the sentence currently being generated and did not in the previously generated sentence then 2.654 should be added to the sentence's current emotional score.

As for the mixed models $m2$ and $m3$, the same two manipulations were significant in model $m2 = \Delta surveyScore \sim state_1 + state_2 + ... + state_k$, whereas in model $m3 = surveyScore \sim \Delta state_1 + \Delta state_2 + ... + \Delta state_k$ there were six significant variables:

1. prefixing "Could you" and suffixing "?" (score: 2.052)
2. suffixing "Thank you." (score: 2.792)
3. prefixing "If you could" (score: 3.582)
4. prefixing "Please" (score: 5.017)
5. suffixing "Thanks." (score: 3.052)
6. prefixing "I would like you to" (score: 4.207)

Similar to above, if we suffix "Thank you." onto a sentence to be generated and it did not occur in the previously-generated sentence the sentence should have an emotional score of 2.792. It should be noted that, at this moment, the system only implements results from model $m1$ and $m4$.

Finally, I attempted to construct model $m5 = surveyScore \sim sigState_1 + sigState_2 + sigState_f + sigState_i * sigState_j$, however R indicated that any interaction that occurred within the generated sentences (e.g. the manipulations prefixing "Could you" and suffixing "Thank you.") among significant variables was not significant.

Looking at the explanatory value of each model the R-squared value of each model is 0.2908, 0.1077, 0.2599, 0.1764 for $m1, m2, m3$, and $m4$, respectively. However, if I control for the person taking the survey in each model I get slightly more explanatory models: 0.3146, 0.0893, 0.2826, 0.1515, for $m1, m2, m3$, and $m4$, respectively. This is to say, if we account for who is taking the survey, the ratio of the explained sum of squares to the total sum of squares in the models increases.

4.2 Accuracy of Emotional Sentence Generation

4.2.1 Emotional Accuracy

How emotionally-accurate are the generated sentences? I evaluated this in two ways. First, by looking at generation across emotion, holding exercises constant and, second, by looking at generation across exercises, holding emotion constant. Table 4.2 shows results of the system, holding exercise constant, while Table 4.1 has annotation information about how the results were generated. In all of the results, for each exercise, I compared a pre-made individual having exactly two exercise properties that differ from the good individual for that exercise (GI from Section 3.3). In Table 4.1 the column **Deficiency** describes

these properties for each exercise. Additionally, the **Initial Phrase** column describes the *initial phrase* sent from the reasoning module to the natural language generation module (see *initial phrase* from Section 3.2). Finally, in Table 4.2 the **Score** column describes an emotional score that can vary between -5 (very stern) and 5 (very nurturing). Note that the sentences for scores -5

Exercise	Deficiency	Initial Phrase
Hip Abduction	iterations	Do the activity more.
	torsoSideBend	Keep your body straight.

Table 4.1: Exercise, deficiencies, and initial phrases used to generate sentences in Table 4.2

Score	Generated Sentence
-5	Do the activity more, alright? Keep your body straight, understand?
-3	Now, do the activity more. Keep your body straight, is that clear?
0	You should do the activity more, is that clear? I would like you to keep your body straight. Thanks.
3	I would like you to do the activity more. Please keep your body straight. Thank you.
5	If you could, do the activity more. Could you keep your body straight?

Table 4.2: Sentence generation, holding exercise constant.

and -3 seem more demanding (questions versus statements) and thus may be somewhat more stern. Indeed, they question whether a patient understands the command at the same time that they command, effectively making the one responsible for two responses.
The first sentence of score 0 does seem more stern than the second in having the suffix "is that clear?". However, where the suffix was meant to inquire about whether the patient understood the commands in the -5 and -3 score cases, here the suffix asks the patient whether he or she understands the rationale behind the exercise. This is asking about something that the patient could reasonably have a question about whereas it is clear that the majority of patients understand commands about positioning their body or performing an activity a greater number of times. Therefore, the suffix plays a different role here. The second 0 score sentence and the first 3 score sentence convert the *initial phrase* command to a desire of the therapist. This takes the focus off the command itself, making the sentence somewhat more nurturing.

Finally, the 5 score sentences are much more nurturing relative to the -5 and -3 score sentences. The sentence "Could you keep your body straight?" is quite

nurturing in that that physical therapist is checking in to ensure that the exercise can be accomplished as opposed to making sure of it as is the case for the sentences having scores of -5 and -3. Table 4.4 shows results holding the emotional score constant whilst varying exercises. Qualitatively, note that the

Exercise	**Deficiency**	**Initial Phrase**
Leg Curl	bentAngle	Bend your knee further.
	iterations	Do the activity more.
Thigh Stretch	exerciseDuration	Do the activity longer.
	kneesTogether	Keep your knees together.
Knee Extension	exerciseDuration	Do the activity longer.
	kneeTwist	Keep your knee straight.
Squats	bentAngle	Bend your knee further.
	kneeTwist	Keep your knee straight.
Hip Abduction	iterations	Do the activity more.
	torsoSideBend	Keep your body straight.

Table 4.3: Exercises, deficiencies, and initial phrases used to generate sentences in Table 4.4

Score	**Generated Sentence**
3	I would like you to bend your knee further. Thank you. Please do the activity more.
3	Please do the activity longer. Thanks. Could you keep your knees together?
3	Please do the activity longer. Thank you. If you could, keep your knee straight.
3	Bend your knee further. Thank you. It is important that you keep your knee straight, alright?
3	Please do the activity more. Keep your body straight. Thank you.

Table 4.4: Sentence generation, holding emotion constant.

majority of generated sentences seem to be nurturing on the whole. Sentences beginning with "Could" and "Please" or ending in "Thanks" and "Thank You" exhibit a stronger nurturing quality than their corresponding *initial phrases*. There do seem to be outliers such as "It is important that you keep your knee straight, alright?". However, as mentioned for similar sentences in the previous table, the suffix "alright?" seems to be checking if the patient needs clarification about why a particular part of the exercise is important, as opposed to seeing if the patient understood a simple command. As the survey results tell us, it is somewhat difficult for people to agree on what a sentence having an emotional score of 3 should look like, so it is perhaps more instructive to analyze whether

the above sentences are moderately nurturing which, as I argued, seems to be the case.

4.3 Repetition Avoidance

Table 4.6 demonstrates repetition avoidance for repeated generation of two different *initial phrases* for a Thigh Stretch exercise. The order in which sentences

Exercise	**Deficiency**	**Initial Phrase**
Thigh Stretch	exerciseDuration	Do the activity longer.
Thigh Stretch	kneesTogether	Keep your knees together.

Table 4.5: Exercise, deficiencies, and initial phrases used to generate sentences in Table 4.6

Score	**Generated Sentence**
2	I would like you to do the activity longer. Please do the activity longer. Thank you. If you could, do the activity longer. The activity goes longer. Please do the activity longer.
2	Your knees should be kept together. Could you keep your knees together? You should keep your knees together. Now, keep your knees together, understand? It is necessary that you keep your knees together.

Table 4.6: Sentence repetition avoidance

are generated in Table 4.6 is the first sentence for the exerciseDuration **Deficiency**, then the first sentence for the kneesTogether **Deficiency**, the second sentence for exerciseDuration, the second sentence for kneesTogether, and so on. Note that there is a significant qualitative difference between pairs of rows for each **Deficiency**.

Further, note that this occurs somewhat at the expense of emotional content. Indeed it should be noted that for the results produced here and elsewhere in order to have a large enough set of sentences to choose from the natural language generation module looks for sentences having scores within a possible range above and below the emotional score asked for. For this reason there are a range of acceptable sentences, from somewhat more stern sentences such as "Now, keep your knees together, understand?" to somewhat more nurturing sentences such as "Please do the activity longer. Thank you." In fact, insofar as the aforementioned formula used to calculate the sentence similarity score

(Equation 3.1) preferences sentence that use different words it prefers to choose a different-looking stern sentence after choosing a nurturing sentence.

4.4 Kinect-Based Feedback

I used right arm bending as an example exercise on which to test the Kinect. Particularly, I tested a patient starting with his arm 90 degrees with respect to the ground and ending with his arm near parallel to the ground (see Figure 4.1). The resulting calculated angle, upon averaging the positions of the patient's elbow at the initial and final state of the exercise was 83.41 degrees. This angle was sent to the reasoning module, which, given a desired bend angle of 90 degrees formulated an *initial phrase* for the deficient property *bentAngle*. Now, because I did not have an arm bend exercise within my exercise ontology I treated the bend angle gotten from the Kinect as a leg curl exercise bend angle. Thus the *initial phrase* produced was "Bend your kneed further". This was sent to the natural language generation module with an emotional score of 1 causing the production of the sentence "Your knee should be bent further".

Note that the system indeed produces a sentence that indicates what a patient would need to do to improve upon their original exercise. Additionally, the sentence seems to be representative of a relatively neutral emotion, as was asked for. Thus, I posit that this system would have appropriate application in instructing physical therapy patients.

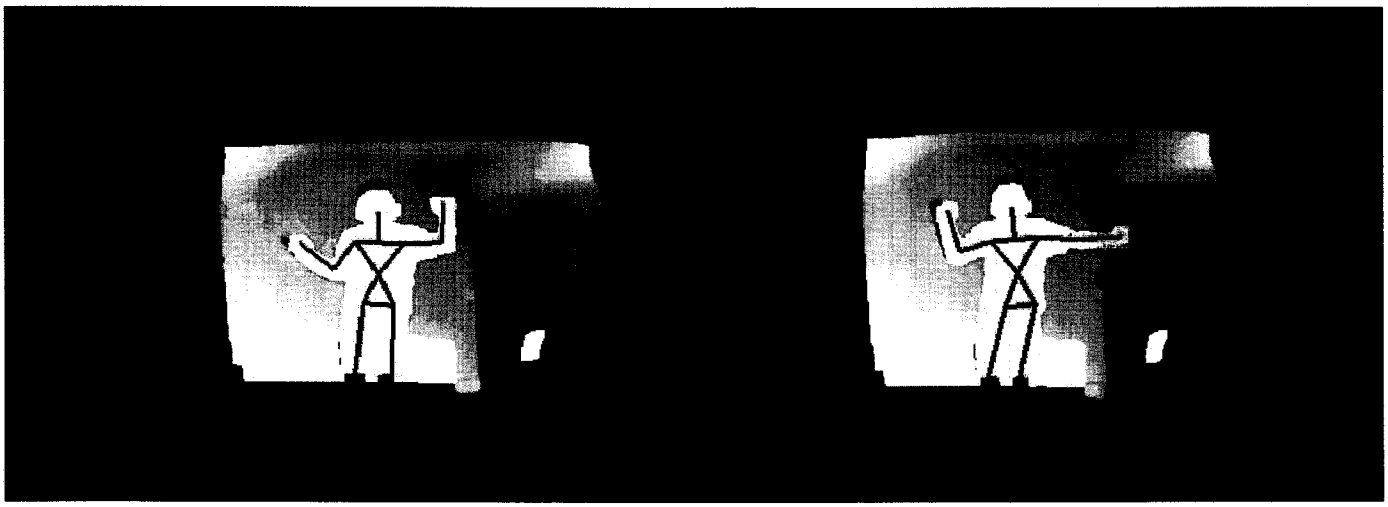

Figure 4.1: Initial and final states of the arm-bend exercise

Part III

Wrap up

Chapter 5

Discussion

One issue in the results is that a similar emotional score often produces sets of sentences that are quite similar in structure. For example, the template "____ bend your knee further ____" used for the bentAngle deficiency in the *Squats* and *Leg Curl* exercises. This is no doubt a function of the *initial phrases* formulated by the reasoning system. The way that the reasoning system was constructed, these sentences all have a basic structure: verb, determiner, noun, adverb. The absence of relative clauses, conjunctions, and prepositions precludes the application of many of the rules within the natural language generation system to these basic sentences, resulting in a limited number of generated sentences. Further, because the *initial phrases* all have the same basic structure, the sentences that are generated for a given phrase will resemble those generated for any other phrase in form. Specifically, the number of generated sentences will be somewhat constrained. Their emotional scores will also be quite similar, because of how a sentence's emotional score is constructed. That is, each generated sentence's rule score will be the same for any *initial phrase*, because the same set of rules are being applied to each *initial phrase*. In the next chapter I describe what can be done to remedy both of these problems.

Another question that must be asked is whether there is actually a trend in the emotional content of the sentences going from stern to nurturing as the emotional score goes from -5 to 5. I see a few points of weakness. First, the feedback for the property *kneesTogether* for the *Squat* exercise in Table 4.4, "Could you keep your knees together?", depending on how it is said, can be somewhat patronizing to a patient, making the sentence much sterner. In fact, for any generated sentence, the tone used by the individual saying the sentence significantly affects the emotional content of the sentence. Indeed, sentences that I have rated to be nurturing could appear somewhat stern if stated in a sarcastic manner. Certainly, this needs to be accounted for. Second, there doesn't seem to be as obvious a distinction in emotion between the sentences generated for the *Hip Abduction* exercise having score 3 and score 5, in Table 4.2, as there is between other exercises that have neighboring emotional scores.

Perhaps something that would remedy this is determining the rule used to generate the 3 score sentences and changing the associated rule scores to be more neutral.

I believe that issues in the emotional scores of the system have two primary causes. First, it seems that a significant number of survey takers thought that I was asking for them to rate sentences absolutely. That is, asking them to say that this sentence is stern while this sentence is nurturing, when in fact we were interested in seeing emotional *differences* between sentences. Second, this interpretation, that we wanted absolute emotional scores for each sentence, produces inconsistent results in the survey.

Specifically, I mentioned earlier that the significance of variables in particular models indicated that survey participants were most likely ranking sentences in one of two ways: First, relative to the sentence before it (as I had intended) and second, on an absolute scale from -2 to 2. These two interpretations led to significantly different responses for each question. Even if I was able to partition surveys by response type I would not then have a large enough set of surveys in each set to generate statistically significant results (assuming at least 3-4 surveys fall into each group).

Additionally, for people that ranked the results along an absolute scale it is likely that they had a difficult time agreeing on what a 2 score sentence was compared to a 1 score sentence or even a 0 sentence. This is because at the beginning of the survey, participants do not have a distribution over which to judge the emotional content of sentences. Indeed, part of the difficulty in trying to judge whether a sentence is of a particular score comes from the fact that one can only have knowledge of the meaning of a score after one knows the possible sentences that can be generated. This is true for how one learns the meaning of any adjective. One sees a number of pumpkins and, without being told whether a particular one is big, one judges it against other pumpkins that have been seen before. What this means is that one may start out thinking that a particular sentence is quite nurturing and give it a score of 2 only to find out that a later sentence is much more nurturing. While one would also give that sentence a score of 2, the survey was constructed so that one could not go and change previous answers. This was so that relative rankings would not be corrupted by participants rethinking their answers. Thus, this is a potential reason why the survey wasn't as informative as I would have hoped.

Chapter 6

Future Work

Each portion of the system has points of weakness that can be improved upon. For the skeleton tracking module there are problems in using the Kinect to get exercise data. The reasoning system is somewhat constrained in the responses it can give about exercises. Finally, there are a few things that can be done to more correctly score generated sentences with an emotion.

Using the Kinect, I only looked at a simple exercise in which there is only one body part moving (arm bend). I did not investigate exercises that involve crouching or making use of exercise equipment. Exercises that require one to crouch would most probably cause the skeleton tracking to be incorrect. Indeed, in testing the skeleton tracking module I found that, upon standing after crouching, the skeleton would often be translated away from its initial position, see Figure 6.1. Also, exercises that use some sort of machine, or even a band as the *HamstringStretchBand* exercise does, have the potential to cause errors in skeleton tracking. Indeed, it is very possible that the Kinect would consider any sort of moving parts as part of the exercising individual and subsequently map the skeleton onto the band or machine. To remedy both of these issues multiple sensors should be used to get exercise data from a patient. Specifically, accelerometers attached to the body part(s) being exercised could eliminate provide less noisy data than the Kinect. It is my understanding that there are researchers at Carnegie Mellon University who are looking at this direction of data collection.

In terms of the reasoning system, for the future, I plan on attempting to add more Utterance individuals and include a greater amount of randomness in the creation of the initial feedback phrase. Further, I may work to create an Adjective Utterance subclass in order to further promote a greater amount of differentiation in feedback. As well, incorporating a sort of modification architecture at the end of the reasoning module that would modulate the initial feedback phrase depending on how well a certain exercise was performed, would add greater specificity to the module. Particularly, I might want to incorporate

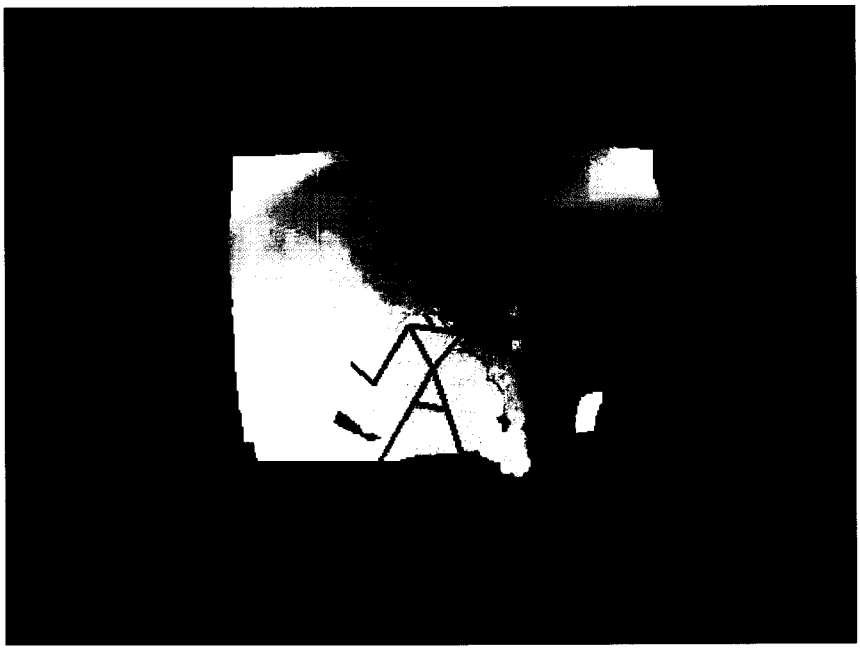

Figure 6.1: Skeleton tracking when returning from crouching position

words like much or slightly to be even more precise as to how the patient should adjust their exercise performance.

One issue was that a number of people were confused about how to rate sentences within the survey. One thing that would probably ameliorate some of the emotional scoring issues arising from this is constructing another survey which, makes it more explicit that sentences are to be scored relatively. I could also try another analysis approach using support vector machines. There exists a program SVMRank which allows one to convert information about dyadic statements such as A is more nurturing than B into rankings for A and B. This might be worth investigating to see if I get significantly different results.

Additionally, attempting to incorporate an emotional lexicon into the current scoring system could improve scoring dramatically. Mohammed and Turney have recently developed an emotional lexicon using an extensive Mechanical Turk survey. [11]. I have been in contact with Mohammed and he is planning on releasing the lexicon in the near future.

Chapter 7

Conclusion

In the introduction I posed three goals that the system presented should meet:

1. Generate emotionally-accurate sentences along a stern/nurturing scale, and
2. Avoid repetition in sentence generation, and
3. Generate accurate feedback based on data from the Xbox Kinect.

For the first goal I believe that there is an emotional dichotomy between sentences with a negative (stern) score and sentences with a positive (nurturing) score. Assuming that sentences are stated with a neutral tone, the more negative sentences seem more stern and urgent and the more positive sentences seem more nurturing and imploring. With this, I do not see a significant difference between positive (negative sentences with different scores. Thus, I have partially fulfilled this goal in that there are clearly two ends of the stern/nurturing scale, but there is not a clear emotional gradient along the scale. As for the second goal, Table 4.6 demonstrates syntactically-different statements with the same semantic content. Therefore, with an eye towards areas of improvement (see Chapter 6), I see this goal as accomplished. Finally, for the third goal, I am able to generate exercise feedback from Kinect data (see Section 4.4). Area for improvement is more heavily weighted on the front end in ensuring correct data is retrieved from the Kinect. Thus, this is a partially accomplished goal, with exciting avenues still to explore.

Bibliography

[1] Baader et al. (2002). The Description Logic Handbook: Theory, Implementation and Application, Cambridge University Press.

[2] Backward chaining. (2011, March 24). In Wikipedia, The Free Encyclopedia. Retrieved 22:05, April 2, 2011, from http://en.wikipedia.org/w/index.php?title=Backward_chaining&oldid=420437620

[3] Berners Lee T., Hendler J., Lassila O. (2001). The Semantic Web. *Scientic American*, http://www.sciam.com/2001/ 0501issue/0501berners-lee.html.

[4] Das, A. & Bandyopadhyay, S. (2009). Subjectivity Detection in English and Bengali: A CRF-based Approach. *International Conference on Natural Language Processing*, Hyderabad, India.

[5] Eccher, C., Ferro, A., & Pisanelli, D. (2010). An Ontology of Therapies. *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, 27, 139-146.

[6] Fleischman, M., & Hovy, E. (2002). Towards Emotional variation in speech-based natural language generation. *International Natural Language Generation Conference*, Arden House, NY.

[7] Forward chaining. (2010, October 19). In Wikipedia, The Free Encyclopedia. Retrieved 22:07, April 2, 2011, from http://en.wikipedia.org/w/index.php?title=Forward_chaining&oldid=391546892

[8] Lessard, G., & Levison, M. (1992). Computational modeling of linguistic humour: Tom Swifties. *ALLC/ACH Joint Annual Conference*, Oxford, p.175178.

[9] Kan, M., McKeown, K. R., & Klavans, J. L. (2001). Applying natural language generation to indicative summarization. Paper presented at *Proceedings of the 8th. European Workshop on Natural Language Generation*, Toulouse, France.

[10] Knublauch, H., Fergerson, R. W., Noy, N. F., & Musen, M. A. (2004) The Protege OWL plugin: An open development environment for semantic web applications. *Proceedings of the 3rd International Semantic Web Conference (ISWC)*, Hiroshima, Japan.

[11] Mohammad, S. & Turney, P. (2010). Emotions Evoked by Common Words and Phrases: Using Mechanical Turk to Create an Emotion Lexicon. *Proceedings of the NAACL-HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, June 2010, LA, California.

[12] Motik, B., Patel-Schneider, P., & Parsia, B. (2008). OWL 2 web ontology language: Structural specification and functional-style syntax.

[13] Parsia, B., & Sirin, E. (2004). Pellet: An OWL DL reasoner. *Proceedings of the 3rd International Semantic Web Conference (ISWC)*, Hiroshima, Japan.

[14] R Development Core Team (2011). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. http://www.R-project.org

[15] Reiter, E. & Dale, R. (1997). Building applied natural language generation systems. *Natural Language Engineering* 3, 1, 5787.

[16] Sashika, H., Matsuba Y, & Watanabe Y. (1996). Home program of physical therapy: Effect on disabilities of patients with total hip arthroplasty. *Arch Phys Med Rehabil.* 77: 273277.

[17] Sigurd, B. (1983). Commentator: a computer model of verbal production. *Linguistics*, 20.

[18] Uschold, M. & Gruninger, M. (1996). Ontologies: Principles, Methods and Applications. *The Knowledge Engineering Review* **11**(2): 93-136

[19] Vavargard, T. (2000). Autotext. Paper presented at *Preprints 2nd Conference on Artificial Intelligence*, Long Beach, CA.

[20] XML 1.0 Origin and Goals (2009). In W3C, The World Wide Web Consortium. Retrieved 21:26, April 2, 2011, from http://www.w3.org/TR/REC-xml/#sec-origin-goals

[21] XML. (2011, March 30). In Wikipedia, The Free Encyclopedia. Retrieved 20:26, April 2, 2011, from http://en.wikipedia.org/w/index.php?title=XML&oldid=421454321