Decision Making by Agent Committee

By

John J. Bevilacqua

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in

Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

January 30th, 2004

Author _____

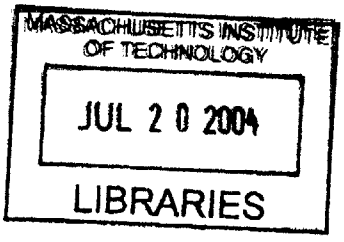
Department of Electrical Engineering and Computer Science
January 30, 2004

Certified By _____

Patrick Henry Winston
Ford Professor of Artificial Intelligence and Computer Science
Thesis Supervisor

Accepted By _____

Arthur C. Smith
Chairman Department Committee on Graduate Theses



Decision Making by Agent Committee

By John J. Bevilacqua

Submitted to the
Department of Electrical Engineering and Computer Science

January 30th, 2004

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

Computer decision making systems are aids that are becoming used in an increasing number and variety of applications. These systems allow people to interface with sources of information far too large and complicated to process themselves. As these systems become used on more advanced and complicated tasks, they will need to become more intelligent. One step towards this is the creation of a multi-agent decision making system that uses behavior inspired by the interactions of groups of people to allow a set of agents with humanistic characteristics to interact with one another. I implemented a program, AgentCommittee, which incorporates such behavior. AgentCommittee uses a set of characteristics that include extroversion, fatigue, resistance, confidence, and competitiveness in a series of one-on-one interactions to arrive at a group decision. The goal of AgentCommittee is not to find the best or optimal answer, but to produce a variety of good answers. I tested this program on a set of data from Compaq's EachMovie database and found that AgentCommittee was 20% more successful at finding relationships between the genre of a movie and a user's opinion of that movie than a random output generator.

Thesis Supervisor: Patrick Winston
Title: Ford Professor of Artificial Intelligence and Computer Science

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The author wishes to thank Patrick Winston, for introducing him to the path that led to this thesis; Paul Keel, for ideas that helped refine this project; his parents, for pushing him to always do his best; and Rebecca, for her continual support.

1. Introduction

During the past decade, the amount of information available to users has increased immensely. Search engines and databases are some of the computational aids that have been used to help manage this sea of data. As the available information increases, it is likely that more advanced, powerful, human-like tools will be needed to help users interface with it. One step towards this goal is the creation of a system for performing highly configurable, adaptive, human-like decision-making. Such a system could contain a set of agents with human-like characteristics that promote their own opinions to each other in a series of interactions to produce a joint decision.

Future multi-agent systems may require more advanced methods of combining input from different agents than the methods in use today. Situations in which a single undistinguished agent has important information may get lost in many of the current systems. Another desirable trait is the ability for systems to produce non-deterministic output. For example, a program that suggests places to eat may want to produce recommendations that aren't always the same. In addition, any decision making system should achieve an acceptable level of performance. This work was conceived as an addition to the EWall project, a multi-agent environment for the management of information (ewall.mit.edu, 2003).

I created a program, AgentCommittee, which implements a decision making algorithm that combines a set of inputs using a system of agent interactions inspired by human group decision making. The goal of AgentCommittee is to find many good answers, rather than a single best or optimal answer. AgentCommittee's agents have a

set of humanistic characteristics including fatigue, confidence, competitiveness, extroversion, and resistance which influence aspects of each agent's interactions. These characteristics dynamically change due to the agent's interactions and feedback from the user of the program. Agents store their opinions as ordered lists in which the most desirable item is in the first position and the least is last.

The rest of section 1 contains the rationale for looking for more advanced ways of performing decision making. Section 2 discusses the background for this thesis by examining related work involving multi-agent decision making systems, using emotions in agent systems, and using agents to simulate negotiation. Section 3 describes the different parts of my program, AgentCommittee. Section 4 contains information on experiments performed using AgentCommittee and examines the results. Section 5 discusses how the components in section 3 influenced the output of AgentCommittee and contains suggestions for future work. Section 6 summarizes the contributions of this thesis.

1.1. The Importance of Computer Decision Making

Decision making is the process of applying information and processes to determine a course of action. Computers are being called upon to perform this task in new circumstances and with higher expectations. At the present time, computers have been integrated into automobiles, phones, personal assistants, televisions, airplanes, and map services, to name a few. The computers in these devices are or will be asked to perform many of the duties that were formerly performed by people. Artifacts such as cars that drive themselves, personal assistants that screen e-mail, televisions that know

what you want to watch, and computers that know what you want to read are already here or on the horizon. The success of these systems will be determined by their ability to make decisions. How flexible can they be? How well can they learn? How well will they identify and incorporate relevant data into their decisions? These are a few of the questions we will use to judge these systems.

These computer decision making systems can make our lives easier and save lives. On the other hand, their failure can complicate our lives and result in the loss of life. People are becoming increasingly dependent upon information from computers when making decisions. What if a computer fault detection system had been able to detect the danger of the damage in the *Columbia* shuttle accident before it broke up in the atmosphere? What if a monitoring computer had been able to identify the circumstances that led to a brownout in 2003 that left much of the northeast United States without power? What if my spam filter hadn't blocked the job offer from Boeing? Clearly, better computer decision making technology offers potential benefits.

1.2. Why model computer decision making systems after people?

We humans make decisions with huge amounts of missing data; we learn from past decisions, and we can even be used in parallel configurations in group decision making. When considering data from other sources, our brains also make use of contextual information such as the emotional state of the person providing it.

I have taken a step toward the embodiment of such capabilities by showing that a computer decision making system inspired by the interactions between members of groups of people can learn how to recommend a movie to a user while demonstrating the following attributes: non-deterministic results and survival of weak concepts.

2. Background

Before discussing the design and implementation of AgentCommittee, I discuss some of the research that inspired AgentCommittee.

2.1. Multi-Agent Decision Making Systems

The interaction system used in AgentCommittee was inspired by research done in the *Team Soar* project (Kang, Waisel, and Wallace, 1998). This project explores how the combination method used to create a decision in a system containing agents with various opinions affects the accuracy of the output. Each agent in *Team Soar* judges the threat presented by an incoming radar signature based on data from a different radar source. Two of the methods used in *Team Soar* to combine opinions are majority win with ties being broken by the agent with the most information and majority win with ties being broken by a particular agent that is designated as the leader. The other methods used are average opinion using either equal weights for each agent or dynamic weights that change based on the agent's past performance. *Team Soar* experiences phenomena such as bonus-assembly effects and an inverse relationship between the amount of information possessed and performance that have also been observed in groups of people. This

project led me to wonder what would happen if a system of agents, such as *Team Soar,* was augmented with more humanlike characteristics and interaction mechanics.

2.2. Emotions in Agents

The use of humanistic characteristics in AgentCommittee stems from previous work in systems incorporating emotional factors. One experiment demonstrated that users playing monopoly from separate locations had more successful interactions when their avatars reflected their emotional state (Yuasa, Yasumura, and Nitta, 2001).

The development of the intelligent agents, IDA and CMattie, took the premise that emotional context information allowed more successful interactions one step further (Lee McCauley, Stan Franklin and Myles Bogner, 2000). CMattie (Conscious Mattie) is an intelligent clerical agent that prepares and distributes announcements for weekly seminars. She communicates with organizers and announcement recipients via e-mail and uses emotions to keep track of how well she is doing and choose the tone of the messages she composes. IDA (Intelligent Distribution Agent) communicates with US Navy sailors using natural language to facilitate the detailing of sailors to assignments. IDA uses emotions in a similar fashion to CMattie, but has a greater amount of emotional depth and variety in actions.

Unlike CMattie and IDA, the agents in AgentCommittee interact primarily with each other. These agents change their emotions based on experience from their interactions and feedback from the user. Agents with useful contributions have their emotions changed in ways that increase their influence on other agents and the system's output. Conversely, agents that have less useful contributions lose influence.

2.3. Simulating Emotional Change and Negotiation

An agent based negotiation/simulation tool that uses a particularly large emotional space that changes dynamically was proposed in "Agents and the Algebra of Emotion" (S.S. Nemani and V.H. Allan, 2001). The goal of that project was to model changes in emotion using a system of agents that make and evaluate offers and counter–offers in a way that mimics humans. Emotions are represented by a 2-dimensional matrix, which has 11 different emotions on one axis and 4 "activators" on the other axis. Activators corresponded to types of events that cause a change in emotions. Events and interactions are evaluated using matrix multiplication. This model of emotions is more complicated than the one used by AgentCommittee. I chose to use a more traditional method of modeling emotion state and change because AgentCommittee uses a much larger number of agents and interactions.

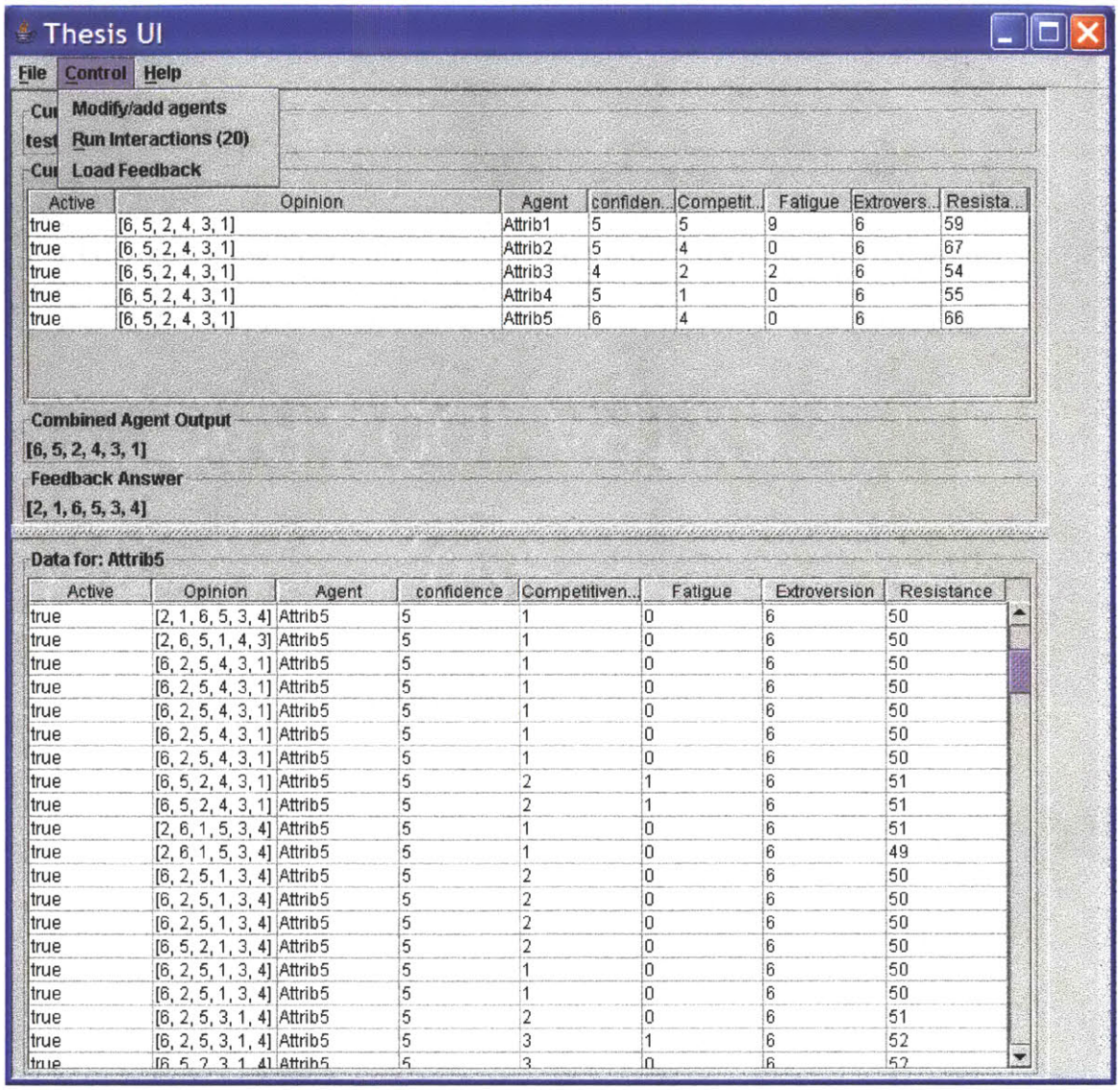


Figure 1: AgentCommittee Graphical User Interface

3. Design and Methods

I implemented a system, AgentCommittee, which performs an agent based decision making process inspired by human group decision making. To create an output, AgentCommittee allows the user to load data and create a group of agents which generate opinions from the data. Next, AgentCommittee's central control program selects pairs of

agents to engage in one-on-one interactions in which they promote their own opinions, negotiate towards a consensus, and update characteristics based on their performance in negotiations. After the agents have concluded their interactions, the central control program arbitrates a final combination if a consensus has not been reached and informs the user of the decision. At this point, the user can either begin setting up another decision or load a feedback opinion which will cause each agent to update its characteristics based on whether its original opinion was more similar to the feedback than the output.

The characteristics of each agent are explained in section 3.1, Characteristics.

Section 3.2, Interaction Mechanics, describes the process by which two agents interact.

Section 3.3, Agent Design, describes the way in which agents form and change their opinions and how they use feedback to update their characteristics.

Section 3.4, Central Control Program, explains how agents are selected for interactions, how the system stores data on agents' states, and how the user interacts with the system.

3.1. Characteristics

As the characteristics of an agent change over time, they should never create a state in which a single agent becomes either overpowering or inconsequential. In addition, the system as a whole should not enter a state of inflation or deflation. When the system enters a state of inflation, the value of a characteristic increases without control, due to insufficient curtailment from feedback systems. Deflation is the

equivalent behavior in the opposite direction. The consequence of either of these states is a situation where active agents are becoming either more or less effective than agents not involved in the recent interactions with no justification. The system should regulate itself in such a way that the characteristics have informal upper and lower bounds. Appendix D contains the code for how characteristics change due to negotiations and user feedback.

Characteristic: Effect:

Extroversion: Determines how likely an agent is to initiate an interaction and accept a request to participate in an interaction.

Competitiveness: Determines how two agents interact:

- **Cooperative** - highest weighted average – create a proposal by taking the average of the two agents' lists.
- **Coexisting** - interleave negotiating method – the agent initiating the negotiation picks the first item in the proposal and then the two agents take turns filling in the rest of the spots with their highest ranked item not yet in the proposal.
- **Combative** - perform a biased random to determine which agent chooses the item for each position on the list. Each agent has a chance to win the random of:
  $confidence_{agent}$ / $Confidence_{total}$

Agent 1 Competitiveness	**Agent 2 Competitiveness**	**Interaction Method:**
>5	>5	Combative
>5	<,=5	Coexisting
<,=5	>5	Coexisting
<,=5	<,=5	Cooperative

Table 1: Effect of competitiveness on the interaction style used by two agents

An agent's competitiveness changes due to its successfulness. Agents with competitiveness greater than 5 are classified as competitive and agents with competitiveness equal or less than 5 are uncompetitive. Each time an agent has a "successful"

interaction, its competitiveness moves towards the nearest extreme. Fatigue mitigates the confidence of agents engaging in combative interactions.

Resistance: Determines how likely an agent is to make changes in its own opinion when interacting with other agents. Increases when an agent receives positive feedback from the user or is frustrated by an interaction. Decreases due to negative feedback from other agents.

Confidence: Determines how strongly an agent holds on to its own opinion. Increases when an agent receives positive feedback from the user. Decreases when an agent receives negative user feedback.

Fatigue: Serves as a dampening factor to keep an agent from becoming too dominant. Fatigue mitigates confidence in agents engaging in combative interactions. Increases when another agent receives negative feedback. Decreases when an agent receives positive feedback from other agents or the user.

3.2. Agent Design

Each agent is identical in structure to every other agent. Agents differ based on the expertise that is used to form their opinion and the current state of their characteristics. The user assigns each agent an expertise that is used to create that agent's opinion on the central control program's current set of data. To create its opinion, the agent orders the items in the data set based on values for the variable that corresponds to the agent's expertise. Ties are broken arbitrarily. If a data set does not contain information

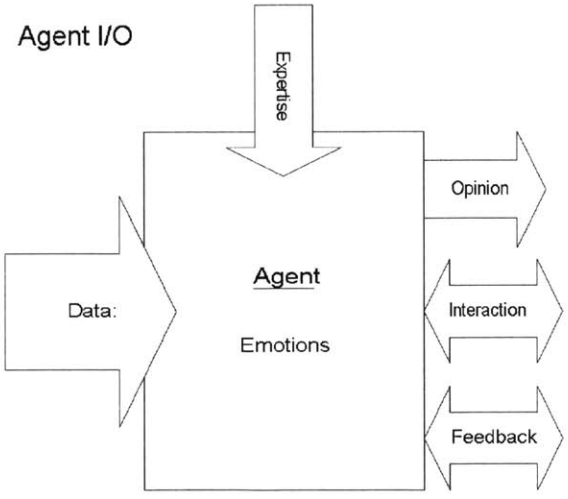


Figure 2: Agent Input/Output

for a particular agent to form an opinion on (i.e. that agent's expertise), that agent will enter a dormant state in which it neither engages in interactions nor participates in the final combination (*see Section 3.4, Central Control Program, for more on the final combination*). If a data set is loaded that allows that agent to form an opinion, it will awaken.

After the agent has formed its opinion, it can interact with other agents to promote its opinion to them and listen to their opinions. Each interaction results in the agents updating their opinions and states (characteristics). At the conclusion of an interaction, the agents also share positive or negative feedback with each other based on the quality of the interaction from that agent's point of view. The quality of an interaction is determined by comparing the agent's original opinion to the current opinions of the two agents before and after the interaction. This comparison uses the formula if figure 3. This is not a zero-sum metric; a positive result for one agent does not require a negative result for the other agent.

```
Compare(original_opinion, other_opinion) {
        For i=1 to length(original_opinion) {
                Rank1 = i
                Item = original_opinion.itemAt(rank1)
                Rank2 = other_opinion.getRank(Item)
                Weight = length(original_opinion) – i
                Sum_differences += [weight*(rank1-rank2)]^2
                }
        return Sum_differences
}
```

Figure 3: Compare opinion pseudo-code

Agents receive user feedback in the form of a "correct" opinion. The agent then compares to see whether its original opinion was closer to the correct opinion than the output of the system using the same formula used above. If the agent's original opinion was closer, it increases its resistance and confidence and decreases its fatigue.

```
Interact(other_agent) {
        Other_chars = other_agent.getCharacteristics()
        Style = pickstyle(own_competitiveness, other_competitiveness)

        If (style == coop) { proposal = CooperativeInteraction(other_agent) }
        Else if   (style == coexist) { proposal = CoexistInteraction(other_agent) }
        Else if   (style == combat) { proposal = CombativeInteraction(other_agent) }

        myNewOpinion = this.updateOpinion(myResistance, proposal, myCurrentOpinion)
        otherNewOpinion = other.updateOpinion(otherResistance, proposal, otherCurrentOpinion)

        feedbackForMe= other.genFeedback(proposal, otherNewOpinion, myNewOpinion)
        feedbackForOther= this.genFeedback(proposal, myNewOpinion, otherNewOpinion)

        this.updateCharacteristics(feedbackForMe, myNewOpinion, OtherNewOpinion)
        other_agent..updateCharacteristics(feedbackForOther, OtherNewOpinion, MyNewOpinion)

        this.setCurrentOpinion(myNewOpinion)
        other_agent.setCurrentOpinion(otherNewOpinion)
}
```

Figure 4: Pseudo-code for interaction mechanics

3.3. Interaction Mechanics

An interaction between two agents is initiated by an agent at the behest of the Central Control Program. The next step is to exchange characteristic data and determine which interaction style to use based on the competitiveness of the agents involved (see Section 3.1 for details). Once an interaction style is chosen, the agent that initiated the contact begins an interaction of the appropriate style with the other agent. The result of this interaction is a proposal, which is a compromise between the two agents' positions. Depending on the interaction style and characteristics of the agents along with a little luck, the compromise may favor one agent's opinion over the other. After this, the two agents each decide how closely they will change their own opinion to the proposal based on resistance.

The next step is to exchange their updated opinions and generate feedback for each other based on how similar the other agent's new opinion is to the proposal compared to how similar to the proposal the agent's own opinion is. Finally, the two agents update their characteristics based

on the feedback from the other agent and an evaluation of whether the interaction served to promote the agent's original opinion. Source code for updating characteristics can be found in Appendix D. Figure 4 shows pseudo-code for the interaction process.

3.4. Central Control Program

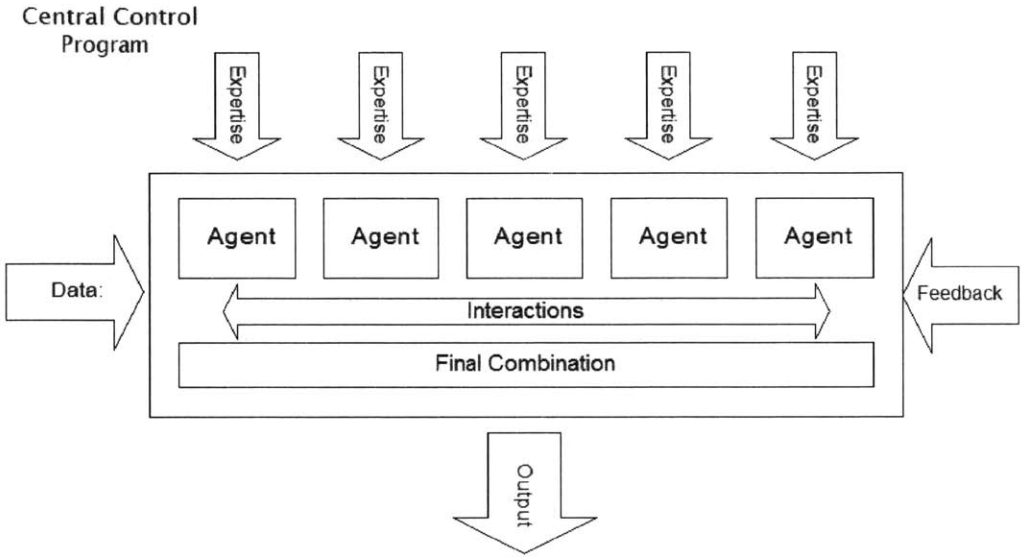


Figure 5: Internal and external data flow chart for Central Control Program

A central control program (CCP) holds the set of agents and determines when and with whom agents interact. The central control program also stores the current data set being examined and a cache of each agent's characteristics after each interaction and final combination. The CCP can use the same set of agents on different sets of data without having to reload the agents; when new data is loaded, the agents are asked to generate an opinion from it. At the conclusion of a set of interactions, the user can load feedback for the system by specifying a location in the data set that contains a "correct" opinion.

Agent interactions are done with a program loop that first selects an agent to initiate the interaction then selects another agent for the first agent to interact with. Agents are selected using a biased random method that favors agents based on how high their extroversion characteristic is relative to the sum of all of the agents' extroversion.

The CCP waits for the two agents to conclude their interaction and then repeats the selection process. After each interaction, the CCP adds snapshots of the two agents involved to its history cache. At the conclusion of a set of interactions, not all agents may have the same opinion. Some agents may be especially stubborn and refuse to change their opinion. In such cases, it may not be possible or desirable for all of the agents to reach the same opinion. In these instances, the Central Control Program serves as an arbitrator and performs a final combination of the agents' opinions by performing an averaging function. This final combination could be accomplished using metrics such as:

- A weighted average of the set of agents using:
    i. user supplied weights
    ii. weights derived from a feedback system
    iii. some other derived weight
- The mode of the set of agents
- The median of the set of agents

The user is responsible for telling the system how many interactions to run before performing a final combination. Once the final combination is done, the output is returned.

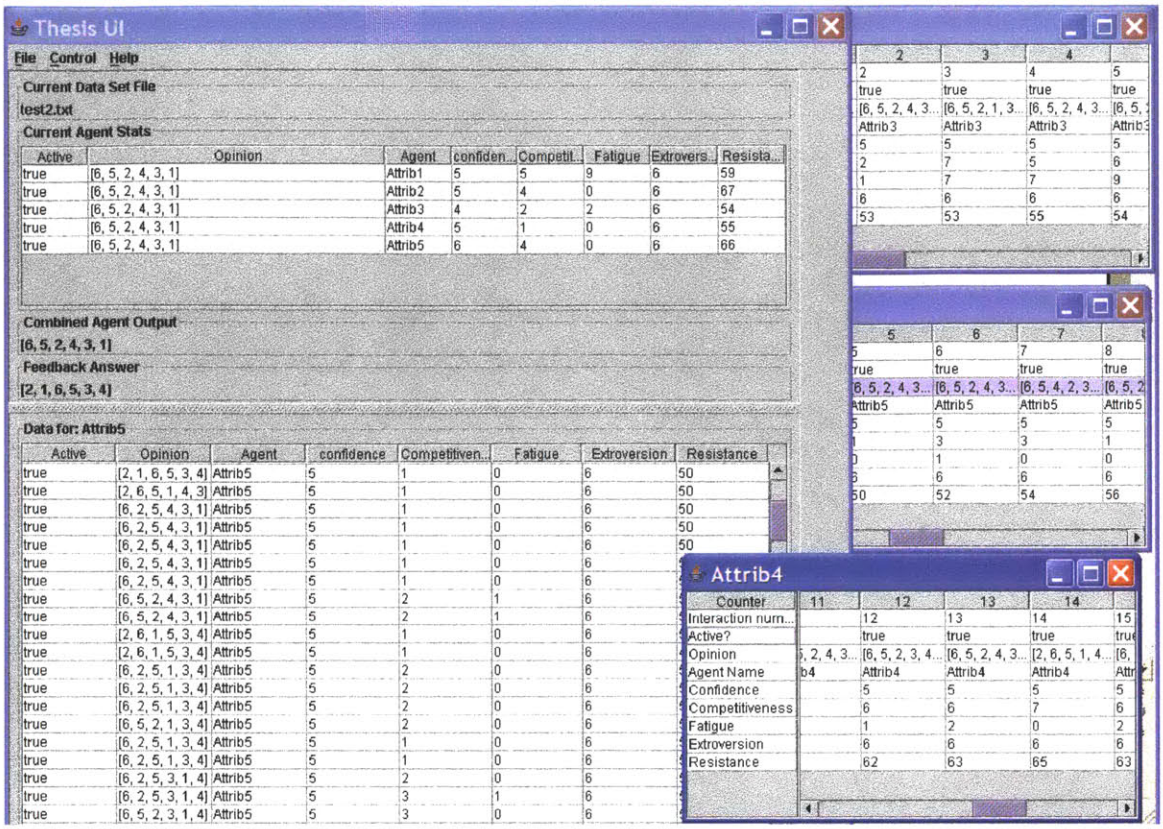


Figure 6: User interface running Survival of Weak Concepts test from 4.2.

4. Evaluating AgentCommittee

An evaluation of AgentCommittee should examine whether the system's actual observed behavior matches the design objectives. Those objectives are:

- non-deterministic output
- allow the survival of weak concepts
- produce good decisions

Section 4.1 looks at how well AgentCommittee produces non-deterministic output. For this section, it was tested with five agents that have randomly generated opinions on a

set of six objects. After generating twenty outputs, the results were compared to see how different the answers produced were and how much the highest ranked items varied.

Section 4.2 examines whether AgentCommittee allows weak concepts to survive. To determine this, the system was tested with four agents that have an identical opinion and a fifth agent that has a substantially different opinion. This test was run twenty times and the result was examined to determine the way in which the system output differed from the majority opinion.

Section 4.3, evaluates AgentCommittee's predictive performance compared to a random output generator on a set of data from Compaq's EachMovie data set. In this test, AgentCommittee was run with sets of 5 movies that a particular user rated. Two of the 5 movies in each of these sets received high ratings (4-5 stars) and the other three received low ratings (0-2 stars, usually 0-1). The results were tested to see how often AgentCommittee was able to pick out the two movies with higher ratings.

4.1. Non-deterministic Output

The first test run was to determine whether AgentCommittee was capable of producing a range of answers when run repeatedly on the same set of data. To determine this, a data set was created with five agents and six objects to be rated by the agents. Each agent had a unique ranking of the six items, which can be found in Appendix A. This set of data was run twenty times while giving the agents involved no feedback from the user. Each run consisted of 20 interactions between agents after which any remaining disagreements were broken by a final combination using an averaging function. The output from AgentCommittee in this test consists of an ordered set of the numbers 1

through 6, where *n* stands for Movie *n*. The first number in the set is the movie ranked first; the last number is the movie ranked last.

The output from this test has a very random distribution (see Table 2). This seems intuitive given the amount of variance in the input data. It is worthwhile to note, however, that many decision making systems would produce a single uniform answer when given this set of data. Pure averaging and voting metrics are two examples of such systems.

When run with five agents that all have similar characteristics and vastly different opinions, the interactions favor no agent more than the others. Each agent's highest ranked item was ranked first between 10 and 40% of the time. Movie 3 is never ranked first in the output, which is expected due to every agent initially ranking it no higher than second. Movies 5 and 6 have the highest average ranking, which is reflected in the results.

Table 2: Output from AgentCommittee on 20 runs on the same data (see Appendix A for the data set).

Run	Output
1.	[1, 3, 5, 6, 4, 2]
2.	[5, 4, 6, 2, 1, 3]
3.	[6, 5, 1, 4, 3, 2]
4.	[5, 1, 4, 3, 2, 6]
5.	[2, 4, 5, 6, 1, 3]
6.	[5, 2, 3, 4, 1, 6]
7.	[5, 6, 2, 3, 4, 1]
8.	[4, 6, 2, 5, 1, 3]
9.	[5, 3, 1, 6, 2, 4]
10.	[6, 5, 4, 1, 3, 2]
11.	[6, 1, 5, 4, 3, 2]
12.	[6, 1, 2, 5, 3, 4]
13.	[5, 6, 2, 1, 4, 3]
14.	[6, 4, 5, 3, 2, 1]
15.	[2, 5, 4, 6, 1, 3]
16.	[5, 6, 1, 3, 4, 2]
17.	[5, 4, 6, 2, 3, 1]
18.	[4, 5, 6, 1, 2, 3]
19.	[1, 6, 4, 3, 2, 5]
20.	[5, 6, 2, 4, 1, 3]

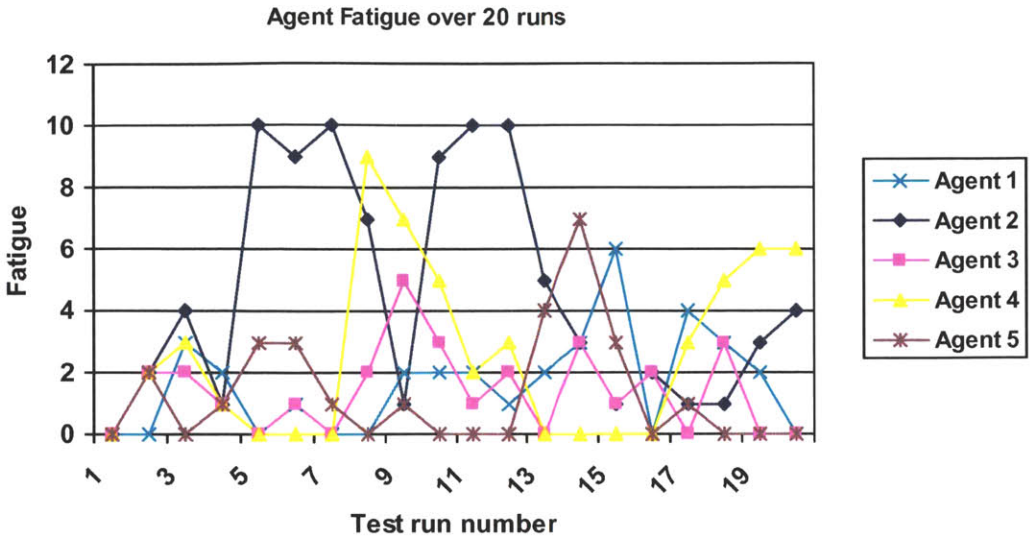


Figure 7: Agent fatigue in non-deterministic test.

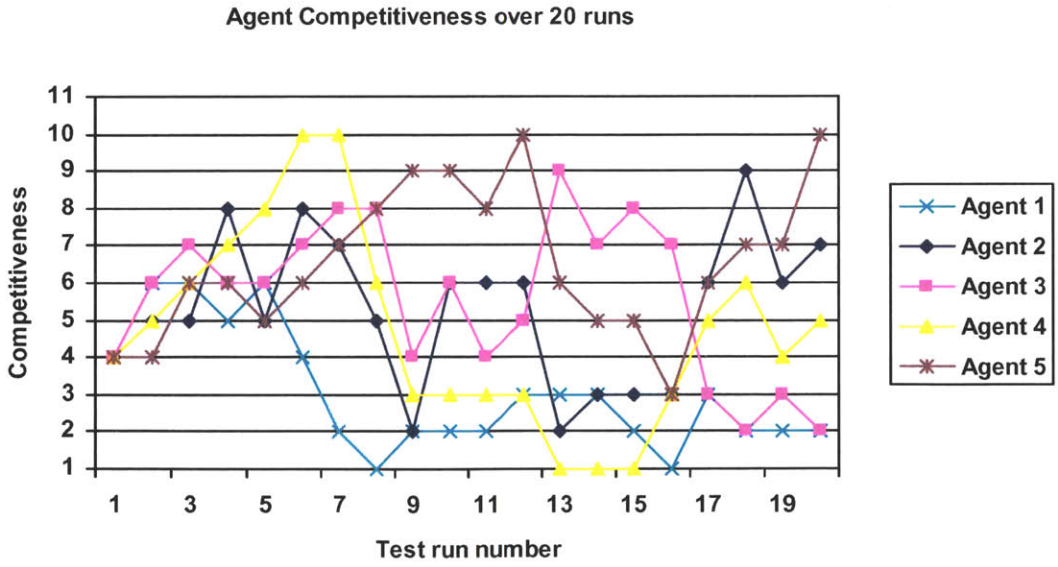


Figure 8: Agent competitiveness in non-deterministic test.

The agents' fatigue in this test exhibits a number of peaks followed by quick descents. The drop in Agent 2's fatigue in the 9th test run corresponds with a change in interaction style as its competitiveness changed from the high end of the spectrum to the

lower end. Figures 7 and 8 illustrate that high levels of competitiveness tend to cause that agent's fatigue to increase, which penalizes that agent in combative interactions. Once an agent's fatigue has grown high, its success in combative interactions greatly decreases. This causes its competitiveness to move towards the lower portion of the competitiveness spectrum. The system as a whole keeps approximately the same total resistance throughout the tests, although Agent 2's resistance drops significantly. This is likely the result of bad luck in Agent 2's interactions as there is no compelling reason for its resistance to drop significantly more than the other agents'.

Figure 9: Agent resistance in non-deterministic test

4.2. Survival of Weak Concepts

The next test was to determine whether the interaction mechanics in AgentCommittee allow for the survival of weak concepts. When there is a set of agents such that a majority of them have very similar opinions, this opinion is called the majority opinion. The opinions of other agents that significantly differ from the majority

opinion are called minority opinions. Survival of weak concepts refers to the ability for a minority opinion to overcome the majority opinion and take precedence in the answer. To test whether AgentCommittee allows this behavior, I created a data set in which there are four agents with the same ranking of 6 movies and a fifth agent that has a very different ranking of the movies.

Table 3: Output rank of highest item in minority opinion in a set of twenty runs

First:	1
Second:	2
Third:	12
Fourth:	3
Fifth:	2
Sixth:	0

Majority Opinion: [6, 5, 4, 3, 2, 1]
Minority Opinion: [2, 1, 6, 5, 3, 4]

Like the first test, this test was run twenty times with 20 interactions per run. The results of these tests are in Appendix B. In most runs, the highest ranked item by the minority agent, Movie 2, was ranked higher in the output than it was in the majority opinion. The minority agent was able to promote its highest ranked item into the top rank in the output 5% of the time and in the second rank 10% of the time. The second ranked item by the minority agent, Movie 1, was ranked last by the rest of the agents. In 85% of the runs, Movie 1 was in the last or second last position. This is as expected since neither agent would ever rank Movie 1 above Movie 2 in any of their interactions, causing Movie 1's rank to become bounded by Movie 2.

The minority agent's competitiveness has a tendency to stay in the extreme ranges of the spectrum, most commonly in the very uncompetitive range, as illustrated in Figure 10. Low competitiveness usually causes agents to receive little or no fatigue from their interactions. This is due to avoiding combative interactions, which offer potentially large gains in influence with another agent at the expense of suffering negative feedback.

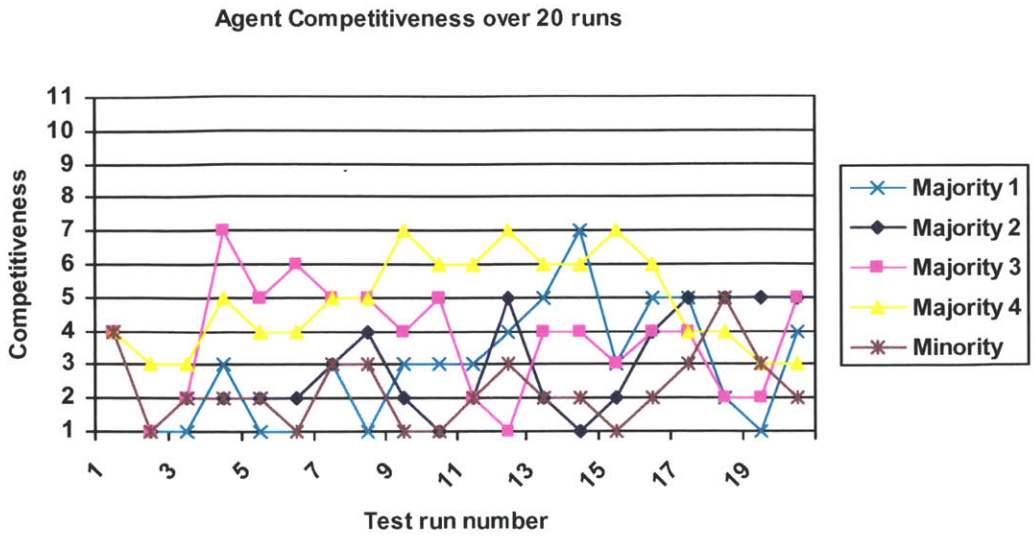


Figure 10: Agent competitiveness in Survival of Weak Concepts test

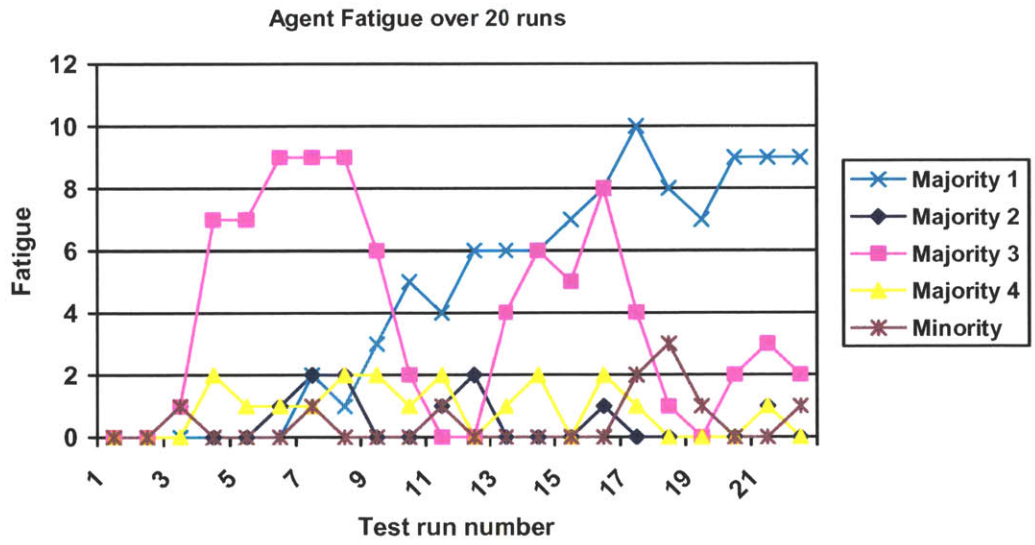


Figure 11: Agent Fatigue in Survival of Weak Concepts test

Coexisting interactions are the most effective interaction method for the minority agent to engage in due to the agent having a good chance to promote its highest ranked item to the top rank on another agent's list. The minority agent remained uncompetitive for the majority of this test. However, there have been other instances when most of the

majority agents became uncompetitive which caused the minority agent to become highly competitive. If the minority agent only engages in cooperative interactions, the net averaging effect created by those interactions will heavily favor the majority opinion. The minority agent is in a race each test run; it must convince a majority agent to adopt its opinion before the combined influence of the other agents cause it to abandon its own opinion.

The ability of the minority agent to overcome the majority opinion in this system is a result of a combination of chance and system mechanics. The agent's feedback mechanisms induce it to avoid areas of the characteristic-space that have been associated with unsuccessful interactions. An advantageous set of characteristics will not cause the minority agent to succeed by itself. The agent also needs a degree of luck. It must encounter the right agents first. The other agent must accept the proposal that is created in the interaction. In the event that another agent does adopt its opinion, that agent must have some success at spreading the minority agent's opinion further. Appendix F follows a test run interaction by interaction in which the minority agent is able to promote its highest ranked item into the output.

4.3. Performance

The purpose of the final test I ran on AgentCommittee was to see how well the program could identify relationships between movies that a user likes/dislikes and the genre(s) of that movie. I hypothesized that such relationships do exist, but do not have a full correlation with the user's evaluation of a movie. To perform this test, I created a set of data out of the EachMovie data set, a data set available online by request from

Compaq Research. The EachMovie data set consists of three files – one of users, one of movies, and one of evaluations of items in the movies file by people from the users file. The movie file contains data for roughly 1600 movies in the format:

ID Name ... Action Animation Art_Foreign Classic Comedy Drama Family Horror Romance Thriller

I omitted the headings for variables regarding production dates and internet URL links because I chose not to use that information in the tests I ran. ID is an arbitrary numerical identifier for a movie, Name is the actual movie name, and the rest of the listed headings are positive/negative (1/0) indications of whether the movie fits into that category. Movies could fit into multiple categories.

The votes file contains information on user ratings for movies. The data format in the votes file is:

UserID MovieID UserRating ...

UserID is a numerical identifier for a user and can be looked up in the user file (I never had reason to do this). MovieID corresponds to the ID column from the movie file. User rating is a numerical rating from 0 to 1.0 in 0.2 increments. A rating of 1.0 equals 5 stars, 0.80 equals 4 stars, etc. A rating of 0.00 meant either 0 stars or that the movie sounded so bad the user had no interest in seeing it. The rest of the attributes in the votes file, such as date, were not used in my tests.

For this test, I made a data file that had seven to ten sets of 5 movies that had been reviewed by the same user for the six users in the test. Each set of 5 movies included two movies that were ranked high (4-5 stars) and three movies that were ranked low (0-2 stars). In several instances, I used the same movie in two sets if a particular user had a

shortage of high or low ranked movies. This happened in less than 10% of the sets. This data set is shown in Appendix C. I created an agent for each of the classification attributes in the movie data set and added them into a central control program. Each agent ranks movies that have a positive classification for the agent's assigned attribute above movies that have a negative classification. In the case that multiple movies have a positive classification for this attribute, ties are broken in favor of the first movie encountered by the agent. Likewise, the first entered movie with a negative classification is ranked higher than the rest of the movies with a similar classification for that attribute. I ran forty interactions for each run and measured how many times the output correctly identified one of the two high ranked movies by ranking it first or second in the output. There were six different users involved and a total of 50 sets of movies. Each test ran the ranking system on these 50 sets a total of 10 times. I ran this test 3 times, giving a total of 30 runs on each set of 5 movies. Each test run was done in identical circumstances; there is no reason to expect any run to be more or less successful than the others. The average number of correct classifications per output, organized by user and test run were:

UserID	**Test run 1**	**Test run 2**	**Test run 3**
1	.8667	.8444	.8444
17	.8857	.8429	.9
23	.7375	.7375	.725
27	1.04	1.03	1.06
71	1.2	1.1875	1.025
119	.9625	.9625	1.0625

Table 4: Average correct classifications by AgentCommittee sorted by userID

A completely random ranking system would average .80 high-ranked agents correct per each set. The highest possible score for any set is 2.00 agents ranked correctly. The higher the correlation is between the genre of a movie and the user rating

of a movie, the better a system making predictions on rating based on type should perform. The test data in Table 4 indicates that this correlation is user specific, which makes sense. The poor performance on data for user 23 is probably a result of that user having a low correlation between his/her rating of movies and their genre. Users 27 and 71 exhibit the strongest correlations between movie classification and rating. Overall, the system averaged roughly .94/2.0 correct classifications. I think this performance is fairly good for the data set due to a few factors including:

- Non-full correlation between movie category and movie rating
- Bias towards movies that fit into more categories due to higher number of agents promoting them. Low likelihood of good items being ranked highly by a high percentage of agents
- Arbitrary tie-breaking system combined with large number of ties due to data consisting of true/false values

The performance might be improved by introducing agents that rank based on ascending order to complement the agents that rank in descending order that are currently in the system. This would allow an agent to promote movies that are not in a certain classification (e.g. Not a horror movie). This improvement might be mitigated by the fact that arbitrary tie breaking has a greater detrimental affect on data with a high number of ties for high ranked spots, such as movies that don't fit into a particular classification. Another way to improve AgentCommittee's success on predicting the user's rank would be to use agents that rank movies based on a factor that has a higher correlation with user ranks than genre. One such factor might be the average rating by critics or a demographic group.

5. Discussion

In this section, I discuss the effects of various aspects of the environment and implementation on the performance of AgentCommittee. Following this, I give suggestions for future work.

Section 5.1, Data Sets, examines how data set formats and distributions affect performance.

Section 5.2, Characteristics, discusses the effects of different characteristics on the performance of that agent and how the environment causes these characteristics to change.

Section 5.3, Interaction Mechanics, examines how the interaction system affects the system's output.

Section 5.4, Future Work, explores future uses and improvements for AgentCommittee.

5.1. Data Sets

AgentCommittee is better suited for making certain kinds of decisions on certain types of data than others. When it is possible to identify which factors contribute to the quality of an item and add agents that have these characteristics as their expertise, AgentCommittee will generate better answers than when it is not. Factors that do not have a correlation with the goodness of the output add noise to the system.

The current implementation of AgentCommittee only stores the relative ratings of the items in its opinion. It breaks ties arbitrarily and only knows that the lower ranked item is not better than one ranked higher; it may be equal. Once an agent forms its opinion list, it discards any other data about the items in the list.

5.2. Characteristics

5.2.1 Competitiveness

One of the most interesting characteristics in AgentCommittee turned out to be competitiveness. The system was designed so that agents that were satisfied with the quality of their one-on-one interactions reinforced the value of their competitiveness towards the nearest extreme. This causes agents with high confidence to gravitate towards being competitive due to the advantage high confidence conveys in interactions between two competitive agents. Agents with low confidence tend to become uncompetitive after having interactions with competitive agents with higher confidence.

The behavior of competitiveness in agents with a minority opinion was particularly interesting. These agents usually change their competitiveness to be the opposite of the agents that they interact with. The uncompetitive/uncompetitive interactions do not favor minority opinions because they cause the system's output to approximate the average combined opinion. This is usually bad for an agent with a very different opinion than the rest of the agents. The majority agents also tend to give the minority agent a lot of negative feedback when it has an interaction that favors the minority agent's opinion. Negative feedback increases fatigue, which penalizes agents in combative interactions. This makes competitive/uncompetitive, or coexisting,

interactions the most successful type for agents with a minority opinion. In these interactions, the minority agent has a good chance to promote its first ranked item to another agent in its first interaction. The chance of an agent (agent1) to promote its first ranked item is to another agent (agent2) in a coexisting interaction is:

$$P_{agent1} = (Extroversion_{agent1}/Extroversion_{agent2}) * Resistance_{agent2}$$

If the minority agent succeeds in doing this, there are two agents promoting the minority agent's top ranked item, which gives it significantly more influence on the system output.

5.2.2 Confidence

Confidence turned out to be interesting as well. The advantage that a more confident agent received in combative interactions caused agents with low confidence that have different opinions than the confident agent to become uncompetitive in order to avoid having to compete with confident agents. In future work, it might be worthwhile to alter the coexisting interactions to convey a greater advantage to more confident agents. Confidence changes solely as a result of user feedback and rises in agents whose original opinions are considered to be better than the combined output. One unintended result is that an output from AgentCommittee that is exactly the same as the user's feedback means that every agent's original opinion was equal to or worse than the output. This causes every agent to receive negative feedback and lose confidence. In future implementations, user feedback will probably also need to give some systemic feedback related to the quality of the system output in addition to the agent-specific feedback.

5.2.3 Resistance

Resistance affects an agent's ability to hold on to its own opinion in any given interaction. The agents' feedback mechanisms determine the amount of upwards and

downwards force on resistance due to the various interactions. These mechanisms can be configured to alter the natural ranges of resistance for a set of agents and data. The natural range of resistance for an agent depends on how conducive the agents' initial opinions are to proposals that the agent identifies as an improvement. If the agent rates most of the proposals from interactions as unfavorable, its resistance increases.

5.2.4 Extroversion

Extroversion was implemented in AgentCommittee, but was not linked to any of the feedback systems. I thought it was a promising concept, but wasn't sure how it should change in response to interactions and feedback. Perhaps an agent with a minority opinion would become introverted to minimize the number of chances the other agents had to push their opinions on it. Agents only know what they learn through interactions, however, which means an agent does not know whether it has a minority opinion until it has participated in a number of interactions. I chose to focus my efforts on the other aspects of the system and leave the complete integration of extroversion for future work.

5.2.5 Fatigue

Fatigue is fairly straightforward. Its main purpose is to give agents a way to diminish the influence of a very confident, resistant, or competitive agent. Negative feedback increases fatigue. Positive feedback from the user and successful interactions without negative feedback decreases it. Another way fatigue could be used is to temporarily mitigate an agent's resistance.

5.3. Interaction Mechanics

Interaction mechanics in AgentCommittee were inspired by observed behavior of human interactions. Some aspects of the interaction mechanics that have a fairly substantial effect on the system's performance and output include:

- Quantitative comparison of two opinions
- Determination of how agents change their opinion
- Evaluation of what a successful interaction is

These aspects could be implemented in a number of ways. While creating and testing AgentCommittee, the metric for how agents compared two opinions changed several times. Originally, the design took the sum of the number of dissimilar items in two OpinionLists. The next method weighted differences in opinion by the size of the differences. Thus, when compared with the ranking [1, 2, 3, 4], the ranking [4, 3, 2, 1] has a greater difference than the ranking [2, 1, 4, 3], even though both rankings have 4 different items. The difference for the first ranking has a weighted sum of 8, compared to 4 for the second ranking. The comparison currently used adds an additional weighting factor based on the rank of the items being compared. The rationale for this is that higher ranked items are more important than low ranked ones. When compared to [1, 2, 3, 4, 5], an opinion of [1, 2, 5, 4, 3] should be more similar than an opinion of [3, 2, 1, 4, 5]. When AgentCommittee was changed to use this last metric, its performance on the third test (performance on the EachMovie data set) increased from being comparable to a random answer generator to roughly 20% better.

The way that agents change their opinion could also be done using various methods. Early designs had agents take turns unilaterally changing their opinion as a

result of observations of the environment. For example, an agent might notice another agent is especially confident and decide to make its own opinion more similar to the confident agent's opinion. I decided that interactions should be made less passive by allowing agents to communicate with each other and directly influence each other's characteristics. I think the current system could improve further by having agents keep a memory of their interactions and allowing a greater variety of behavior.

Currently, an agent determines if an interaction was successful by comparing the total difference from its original opinion at the beginning of an interaction to the total difference at the end of the interaction. The formula for this is:

$$Total\ Difference = difference(Opinion_{A1original}, Opinion_{A1current}) + difference(Opinion_{A1original}, Opinion_{A2current})$$

Difference(opinion1, opinion2) is the same formula that was discussed above. This is different from early versions of AgentCommittee, which used the agent's current opinion at the time the interaction started rather than its original opinion. This method caused the first term in the total difference to always be zero at the start of an interaction and resulted in agents having little attachment to their original opinions.

I think the framework of using one-on-one interactions to allow agents to modify characteristics and opinions has the potential for more interesting behavior and better performance. One step towards realizing this potential is the exploration of new and better ways for the agents to interact.

5.4. Future Work

AgentCommittee's framework can be applied to almost any type of decision-making task. With modifications to some of the interaction mechanics and agent behavior, it could be optimized for classification tasks in areas such as medical decision support.

One important aspect of AgentCommittee is that it can be easily broken into a few layers and subsystems. This makes it possible to change comparison algorithms, characteristic modification, and interaction methods without having to modify the entire system. There is also the potential to greatly increase the depth of the interaction system that agents in AgentCommittee participate in. Rather than having single sets of characteristics, agents could be modified to have a unique disposition for each agent. Agents could learn relationships between their own opinions and other agent's opinions and have a less self-centered viewpoint.

Another possibility involves increasing support for multiple users. Each agent could learn a particular state/disposition to use when acting on the behalf of a particular user. It could also learn relationships between different users in order to apply knowledge learned from one user to other users.

One new characteristic that could have a strong benefit on the system would identify how strongly an agent feels about its opinion. Currently, every opinion is considered to be of equal worth by its owner. It would be useful for agents to judge their opinions to differentiate between items which they feel strongly about and things which they are indifferent towards. This would allow an agent that thinks all of the possibilities it is presented with are of equal worth to have low resistance to changing its opinion

when interacting with agents that feel more strongly about their own opinion. This would be recognized when the agent processes feedback from the user, causing the agent to be less affected by the value of its original opinion for that particular decision.

6. Contributions

In this thesis, I introduced a framework for a multi-agent system in which agents with humanistic characteristics engage in one-on-one interactions to perform decision-making. This framework is inspired by multi-party negotiating behavior observed in a negotiating class, *Power and Negotiation*, taken at MIT's Sloan School.

I wrote a program, AgentCommittee, to implement this system using a set of characteristics and interaction mechanics I designed. AgentCommittee converts a data file into a set of lists, ordered by a particular attribute, which can be assigned to individual agents that promote their own list to other agents. After the agents engage in a user determined number of interactions, a central control agent collects each agent's updated list and performs a combination on these lists if a unanimous decision has not been reached. At this point, the user can give each a feedback opinion, which causes the agents to change their characteristics based on whether their original opinions were more similar to the feedback than the system's output.

I tested and analyzed AgentCommittee's performance on a section of data from Compaq's EachMovie database and found that it performed 20% better than an algorithm that randomly created output lists. This test involved 250 movie reviews performed by 6 different users. In this test, agents promoted movies based on whether they fit into

certain genre categories such as drama, horror, family, animated, action, etc. In addition, I ran several tests on data sets which determined that AgentCommittee's output was highly non-deterministic when and allowed the propagation of a minority opinion into one of the highest positions in the output 15% of the time when run on a set of four agents possessing a majority opinion and one agent possessing a very different minority opinion.

Appendix A: Input data for Non-deterministic Test

Input data:

Movie	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5	Average
Movie 1	1	6	2	1	5	**3**
Movie 2	2	1	3	4	6	**3.2**
Movie 3	3	5	4	5	2	**3.8**
Movie 4	4	2	6	3	1	**3.2**
Movie 5	5	4	1	6	3	**3.8**
Movie 6	6	3	5	2	4	**4**

Table 5: Input data for non-deterministic behavior test

*For the movies in the input data, a higher number is better. Agent 1 ranks movie 6 first and movie 1 last.

Appendix B: Input Data and Results from Survival of Weak Concepts Test

Input data:

Movie	Majority 1	Majority 2	Majority 3	Majority 4	Minority
Movie 1	1	1	1	1	5
Movie 2	2	2	2	2	6
Movie 3	3	3	3	3	2
Movie 4	4	4	4	4	1
Movie 5	5	5	5	5	3
Movie 6	6	6	6	6	4

Table 6: Input data for survival of weak concepts test

Results:

Instances in which the minority agent's top ranked item is ranked 1st or second in the output in bold.

Run	Output
1	[6, 5, 4, 2, 3, 1]
2	[6, 5, 2, 4, 3, 1]
3	[6, 5, 2, 1, 3, 4]
4	[6, 5, 2, 4, 3, 1]
5	[6, 5, 2, 4, 3, 1]
6	[6, 5, 2, 4, 3, 1]
7	[6, 5, 4, 2, 3, 1]
8	[6, 5, 2, 4, 3, 1]
9	[6, 5, 4, 2, 3, 1]
10	[6, 5, 2, 4, 3, 1]
11	[6, 5, 2, 4, 3, 1]
12	[6, 5, 2, 3, 4, 1]
13	[6, 5, 2, 4, 3, 1]
14	**[2, 6, 5, 1, 4, 3]**
15	**[6, 2, 5, 4, 3, 1]**
16	[6, 5, 2, 4, 1, 3]
17	[6, 5, 4, 3, 2, 1]
18	[6, 5, 4, 3, 2, 1]
19	**[6, 2, 5, 1, 4, 3]**
20	[6, 5, 2, 4, 1, 3]

Table 7: Output of survival of weak concepts test

Appendix C: Input data for Performance Test

Input data:

Table 8: Input data for performance test

User	Movie	Rating	User	Movie	Rating	User	Movie	Rating
1	13	0	23	800	5	27	736	5
1	17	4	23	132	1	27	783	4
1	18	1	23	1	4	27	785	0
1	44	0	23	288	1	27	174	0
1	34	4	23	662	0	27	586	1
1	162	5	23	36	5	71	110	5
1	66	0	23	39	4	71	688	1
1	31	1	23	44	0	71	19	1
1	39	4	23	95	2	71	14	2
1	104	0	23	650	1	71	141	5
1	95	1	23	300	5	71	34	1
1	45	4	23	661	2	71	150	5
1	55	4	23	163	1	71	203	5
1	62	1	23	162	5	71	158	1
1	157	0	23	208	0	71	160	2
1	160	0	23	208	0	71	296	5
1	111	4	23	209	2	71	300	4
1	112	1	23	224	5	71	344	0
1	149	0	23	105	0	71	317	1
1	150	4	23	232	5	71	364	1
1	152	1	23	253	1	71	356	5
1	162	5	23	1233	5	71	368	2
1	165	1	23	648	1	71	454	5
1	169	0	23	687	2	71	489	1
1	25	4	23	194	5	71	432	2
1	296	5	23	93	0	71	494	2
1	173	1	23	334	1	71	508	5
1	181	0	23	698	1	71	648	2
1	117	1	23	161	4	71	688	1
1	247	4	23	203	5	71	708	5

User	Movie	Rating
1	260	5
1	172	0
1	186	1
1	296	5
1	267	0
1	269	1
1	204	0
1	344	4
1	286	0
1	265	5
1	305	1
1	310	0
1	357	4
1	355	0
1	356	4
17	172	1
17	32	5
17	170	0
17	111	5
17	327	0
17	223	5
17	228	1
17	296	5
17	305	2
17	327	0
17	270	1
17	322	4
17	333	2
17	153	2
17	39	4
17	589	5
17	327	0
17	254	1
17	323	1
17	778	5

User	Movie	Rating
23	691	0
23	712	1
23	720	4
23	616	0
23	150	5
23	17	0
23	537	2
23	924	5
23	1094	5
23	35	0
27	12	1
27	1	5
27	17	5
27	18	1
27	3	0
27	87	0
27	5	4
27	7	5
27	151	1
27	54	0
27	10	4
27	11	5
27	157	0
27	250	0
27	253	1
27	281	1
27	282	4
27	288	0
27	289	3
27	292	2
27	470	0
27	471	1
27	474	4
27	539	5
27	543	0

User	Movie	Rating
71	539	5
71	541	1
71	169	1
71	172	2
71	349	5
71	420	2
71	590	5
71	185	1
71	350	5
71	288	1
71	593	5
71	185	1
71	350	5
71	288	1
71	616	0
119	412	5
119	413	1
119	202	1
119	25	5
119	122	2
119	124	5
119	206	1
119	288	2
119	290	0
119	296	5
119	216	2
119	223	5
119	243	1
119	373	5
119	377	1
119	339	1
119	344	5
119	26	5
119	443	0
119	444	1

User	Movie	Rating
17	234	1
17	410	0
17	784	2
17	858	5
17	150	4
17	539	2
17	151	4
17	161	4
17	339	2
17	435	1
17	708	0
17	1461	1
17	427	0
17	162	4
17	246	4

User	Movie	Rating
27	546	1
27	365	2
27	367	5
27	368	4
27	374	0
27	527	5
27	536	2
27	552	4
27	569	0
27	747	0
27	780	5
27	798	0
27	799	1
27	802	5
27	177	1
27	168	4
27	174	0
27	1042	5
27	481	2
27	586	1

User	Movie	Rating
119	371	1
119	372	0
119	381	4
119	437	1
119	469	5
119	78	5
119	79	0
119	435	1
119	17	5
119	152	2
119	1099	5
119	1114	2
119	1126	1
119	501	5
119	510	1
119	1263	5
119	1331	1
119	1224	5
119	1227	0
119	1238	2

Appendix D: Code for Characteristic Change

User Feedback Processing:

```
// takes as input an OpinionList which corresponds to the actual
// opinion of the user. Compares both the agent's original opinion
// and the output of the system to the actual opinion.
// Modifies the agent's characteristics based on whether the Agent's original
// opinion was more or less similar to the actual opinion.

  public void processUserFeedback(OpinionList userlist) {
        int mydiff = CompareLists(userlist, this.original_opinion);        // compare to original
        int resdiff = CompareLists(userlist, this.current_opinion);        // compare to output
        if (mydiff < resdiff) {                                  // if original opinion was more similar
          if (resistance < 60) {resistance = resistance+6;}  // increase resistance unless it is high
          confidence++;                                          // increase confidence and fatigue
          fatigue =fatigue-3;
                                                  // if the agent is not competitive, make it more so
          if (competitiveness < 6) {competitiveness++;}          }
        else {
          if (mydiff > resdiff) {                    // if output was better than original opinion
                if (resistance > 40) {resistance = resistance-4;}
                SecureRandom gen = new SecureRandom();
                int g = gen.nextInt(10);
                if (g < confidence) { confidence--; } // decrease confidence and resistance
          }
        }
        // if the system is allowed to revert its opinion during maintanence, disable reverting
        // for now
        boolean ar = allow_revert;
        allow_revert = false;
        this.maintain();          // ensure none of the characteristic values is out of bounds
        allow_revert = ar;        // restore the allow_revert to its previous state
  }
```

Compare Lists:

```
        // compares how similar newList is to oldList using a method that weights
        // high ranked items on oldList and the size of the disparity between rankings
  public int CompareLists(OpinionList oldList, OpinionList newList) {
        int ans = 0;
        for (int j=0; j < oldList.size(); j++) {      // for each item in oldList
          String item = oldList.getElementAt(j);
          int diff = java.lang.Math.abs(j - newList.getRank(item));  // find the difference in rank
          ans = ans +((oldList.size()-j)*diff)^2; }          // take the sum of the squares of the
        return ans;                                          // difference in rank * the weight of that
        }                                                    // rank
```

Generate Feedback:

```
// measure the difference between the updated opinions of two agents and the OpinionList
// that represents the proposal generated by their interaction.  Returns a measurement of
// how much more the other agent's new opinion differs from the proposal than this
// agent's opinion does.  May want to use a different measurement that isn't zero-sum
// in future.

   public double genFeedback(OpinionList agreedList, OpinionList myNew, OpinionList
otherNew) {
        double ans;
        int listsize = myNew.size();       // number of items in opinion
                // average own difference
        double mysumsq = CompareLists(agreedList, myNew);
                // average other agents difference
        double othersumsq = CompareLists(agreedList, otherNew);
        double myave = java.lang.Math.sqrt(mysumsq);
        double otherave = java.lang.Math.sqrt(othersumsq);
        ans = (otherave - myave)/listsize;
        return ans;
   }
```

Update Characteristics from an Interaction:

```
// compares the states of this agent and the other agent before and after the interaction
// updates the agent's characteristics based on this comparison and feedback from the
// other agent.

   public void updateCharacteristics(OpinionList myNew, OpinionList otherOld,
                                     OpinionList otherNew, double feedback) {
        OpinionList myOld = this.original_opinion;       // myOld is the reference opinion
                                                         // that this agent wants to promote
        double feedbackmax = .5;         // maximum feedback to not get penalized
        double compchangespeed = 0;      // determines how fast competitiveness
                                         // can change.  0 is fast, 10 is never
        SecureRandom gen = new SecureRandom();
                // how similar were both opinions to myOld before this interaction?
        int oldSumSq = (CompareLists(myOld, otherOld) +
                        CompareLists(myOld, this.current_opinion));
                // how similar are the two opinions to myOld after this interaction?
        int newSumSq = (CompareLists(myOld, otherNew) + CompareLists(myOld, myNew));
        if (newSumSq == oldSumSq) { }          // if no change, don't change based on that
        else {
          if (newSumSq < oldSumSq) {           // if new opinions are more similar to myOld
                this.fatigue--;                // decrease fatigue
                int q = gen.nextInt(10);
                if (q>compchangespeed) {       // give a chance to change competitiveness
                                               // currently a 100% chance, increase the number
                                               // q is compare to in order to slow down the
```

```
                                        // rate that competitiveness changes
            if (this.competitiveness > 5) {         // reinforce competitive style
                this.competitiveness = competitiveness+1; }
            else {
                this.competitiveness = competitiveness-1; }
        }
    }
    else {          // unsuccessful interaction
        this.fatigue++;
        int y = gen.nextInt(300);           //
        if (y > this.resistance) {this.resistance++; }
        int q = gen.nextInt(10);
        if (q>compchangespeed) {
            if (this.competitiveness > 5) {         // negatively reinforce competitiveness
                this.competitiveness = competitiveness-1; }
            else {
                this.competitiveness = competitiveness+1; }
        }
    }
}
//      process feedback
if (feedback > feedbackmax) {   // if bad feedback
    this.fatigue =this.fatigue++;
    this.resistance =this.resistance-2;
    if (competitiveness > 8) { competitiveness--; }
}
if (feedback < 0) {         // if feedback is good, decrease fatigue
    this.fatigue--;
}
}
```

Appendix E: Interaction Scenario

Scenario:

The user wants to find a few good comedies that her family can enjoy.

Step 1:
She selects agents with expertise in:

- comedy genre
- pg-13 or less maturity rating
- a high rating by the 13-18 year old demographic
- a high rating by the female 30-45 year old demographic.

Step 2:
The user downloads a file with a list of the available movies, useful data on each movie, and rankings of those movies by various demographics. Then the user imports the data set into the program.

Step 3:
Inside the program, the central control program sends the data set to each of the agents (who have been picked by the user based on their expertise). Each agent looks at each of the movies in the list and its data and creates a list in which movies are ordered by that agent's expertise.

Step 4:
The central control program needs to have the agents interact and exchange opinions so that they can try to promote their own opinions to each other. It knows introverted agents would rather interact less and extroverted agents would prefer to interact more. To accommodate this, the central control program rolls a biased die in which sides corresponding to extroverted agents have a higher chance to come up. It rolls the die until it comes up with two different agents and then asks the two of them to discuss their opinions and suggests that each make some concessions so they can come closer to an agreement.

Step 5.
Agents Ben and Alfred are meeting to exchange opinions and see if they can agree to each reorder their own opinion list in a way to increase their success in the final combination. Ben himself isn't very competitive with a competition value of 3/10, however Alfred is (with a value of 6/10). They decide to use a coexisting interaction method. They take turns picking which movie will go in each spot on the list that they will agree on. They flip a coin to see who will choose the first position and Ben wins. Ben chooses the highest movie on his list to be the highest one on the combined list. Next, Alfred chooses his highest ranked movie to be second on the combined list. Ben's second ranked movie happens to have been the one that Alfred just added, so Ben takes

his next highest ranked movie and puts it in the third position. They continue until the new list includes all of the movies.

Step 5 v2.
Agents Alfred and Mary are selected to have an interaction. They are both competitive with competition values of 6.0 and 7.0 respectively. Therefore, they decide to have a combative interaction. They flip a biased coin for each slot on the combined list to decide who gets to pick which movie goes in that slot. The coin is biased such that the probability of heads coming up =
$(confidence_{Alfred} - fatigue_{Alfred}) / ((confidence_{Alfred} - fatigue_{Alfred}) + (confidence_{Mary} - fatigue_{Mary}))$
$= 8\text{-}2/(8\text{-}2 + 7 - 1) = \frac{1}{2}$
probability of tails =
$(confidence_{Mary} - fatigue_{Mary}) / ((confidence_{Alfred} - fatigue_{Alfred}) + (confidence_{Mary} - fatigue_{Mary}))$
$= \frac{1}{2}$

Step 6.
Ben and Alfred just interacted and created a new list with input from each of them. However, Ben's resistance to changing his opinion is fairly high (60/100), so he decides that he should only make half of the changes (by flipping a biased coin that has a 6/10 chance to come up heads and a 40/100 chance to come up tails for each potential change) that would be necessary to change his list into the one he and Alfred agreed upon.

Step 7.
For each item on the list, he flips the coin. If it comes up heads, his list will be updated with the highest item from his original list in that slot. If it comes up tails, his list will be updated with the highest item from the new combined list in that slot.

List A0 – Ben's original list:
1. Gone with the Wind
2. Gladiator
3. Muppets in Manhattan
4. Othello

List B0 – Alfred's original list:
1. Gladiator
2. Gone with the Wind
3. Muppets in Manhattan
4. Othello

List A1 – Ben's old list:
1. Gone with the Wind
2. Gladiator
3. Muppets in Manhattan
4. Othello

List B1 – Alfred's old list:
1.. Gladiator

2. Othello
3. Gone with the Wind
4. Muppets in Manhattan

List C - the combined list:

1. Gladiator
2. Gone with the wind
3. Othello
4. Muppets in Manhattan

Ben's die rolls were: heads, heads, tails, tails so his new list is:
List A2 – Ben's new list:

1. Gone with the wind
2. Gladiator
3. Othello
4. Muppets in Manhattan

Alfred's die rolls were: tails, tails, tails, tails so his new list is:
List B2 – Alfred's new list:

1. Gladiator
2. Gone with the wind
3. Othello
4. Muppets in Manhattan

List A0 – Ben's original list:

5. Gone with the Wind
6. Gladiator
7. Muppets in Manhattan
8. Othello

List B0 – Alfred's original list:

5. Gladiator
6. Gone with the Wind
7. Muppets in Manhattan
8. Othello

Step 8.

After Ben finishes making the changes to his list, he waits for Alfred to do the same. Then, they exchange their new updated opinions. Alfred's resistance happened to be low (3/10) and he changed his list to exactly what they had agreed upon. Ben notices that his and Alfred's updated opinions are more similar to Ben's original opinion than the opinions were before the interaction. He gives Alfred positive feedback. Alfred isn't as happy. He observes that the set of new lists is less similar to his original opinion than the

two lists were before the interaction started using the CompareLists method in Appendix D.

Step 9.
He complains to Ben, giving him negative feedback of 1.4, which makes Ben's fatigue increase from 5 to 7 and resistance drop from 60 to 58. This increased fatigue causes Ben to become less effective in combative interactions.

Ben's feedback is less than zero, which results in no changes in Alfred's characteristics.

Step 10.
Weighted sums of squares of the differences in rank:
Ben's new vs Alfred's original: 3*1+2*1+1*1+0*1=6.
Alfred's new vs. Alfred's original = 3*0+2*0+1*1+0*1=1.
Alfred's new vs. Ben's original = 3*1+2*1+1*1+0*1=6.
Ben's new vs. Ben's original = 3*0+2*0+1*1+0*1=1.

Ben's old vs Alfred's original: 3*1+2*1+1*0+0*0=5.
Alfred's old vs. Alfred's original = 3*0+2*1+1*1+0*2=3.
Alfred's old vs. Ben's original = 3*2+2*2+1*1+0*1=11.
Ben's old vs. Ben's original = 3*0+2*0+1*0+0*0=0.

old vs Ben's original = 11+0=11
old vs Alfred's original = 5+3=8
new vs Ben's original = 6+1=7
new vs Alfred's original = 6+1=7
By this measure, both agents made out well in the negotiation – the new lists have a smaller combined sum of the squares of differences than the old lists for both agents. Therefore, each agent considers it a successful interaction for himself.

Ben decides that he'll continue to do well by keeping an uncompetitive attitude so he decreases his competitiveness from 3 to 2.

Alfred increases his competitiveness from 6 to 7.

Step 11.
After twenty interactions, the central program realizes that it has met the maximum time limit for interactions that was set by the user. It then does a final combination by figuring out the average weighted rank by multiplying each item's rank value by the confidence of the agent who ranked it.

***Note: The tests in Section 4 were done using equal weights for each agent unlike the example below, in which weights are based on confidence.

A rank of one would be worth 10, a rank of two worth 8, three worth 6, etc.

	Agent 1	Agent 2	Agent 3	Agent 4
Agent confidence	5	7	3	4
First ranked movie	Gladiator	Othello	Muppets	Muppets
Second ranked movie	Othello	Gladiator	Gladiator	Othello
Third ranked movie	Muppets	Muppets	Othello	Gladiator

Gladiator = 10*5 + 8*7 + 8*3 + 6*4= 154
Othello = 8*5 + 10*7 + 6*3 + 8*4= 160
Muppets = 6*5 + 6*7 + 10*3 + 10*4= 142

Final list:
1. Othello
2. Gladiator
3. Muppets

Appendix F: Detailed Interaction Data

Data from a run of Survival of Weak Concepts test in which the minority agent successfully promoted its highest ranked item into the top spot in the output.

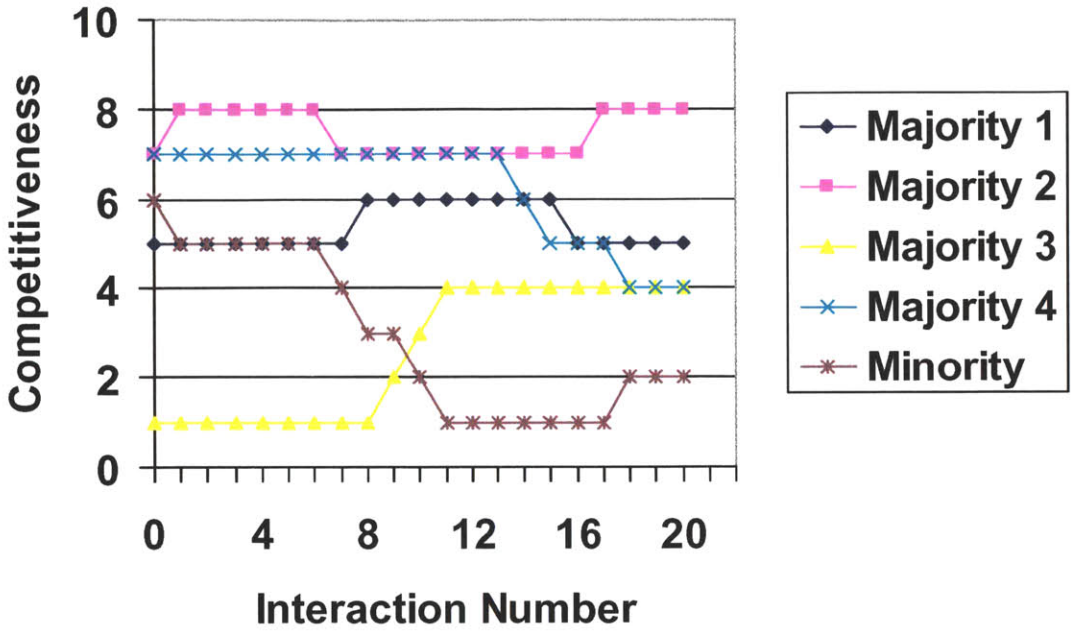


Figure 12: Competitiveness after each interaction in a run in which the minority agent was successful

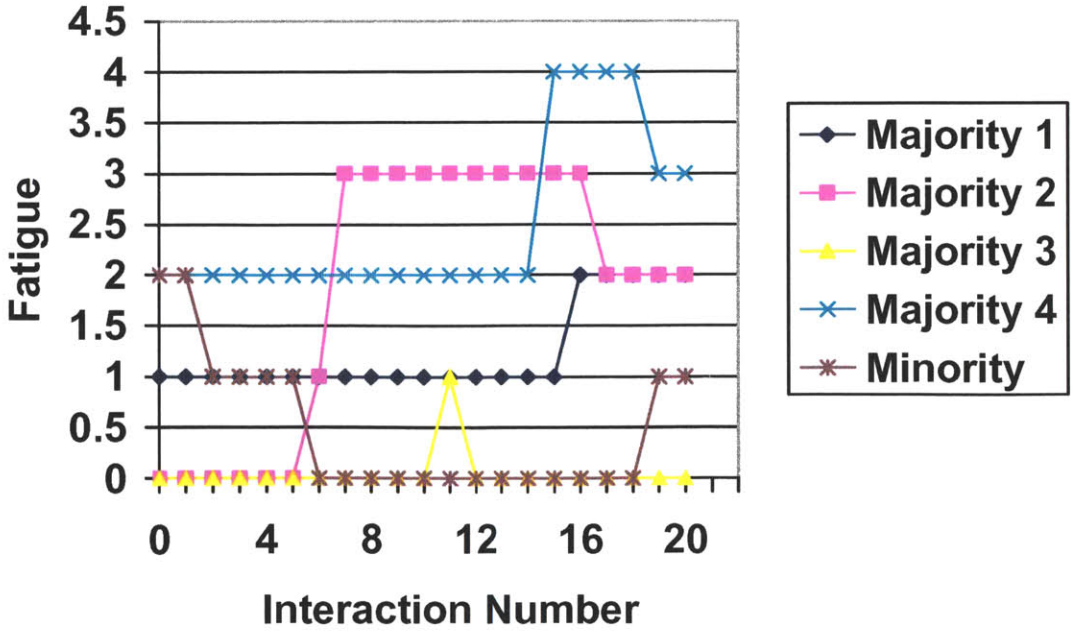


Figure 13: Fatigue after each interaction in a run in which the minority agent was successful

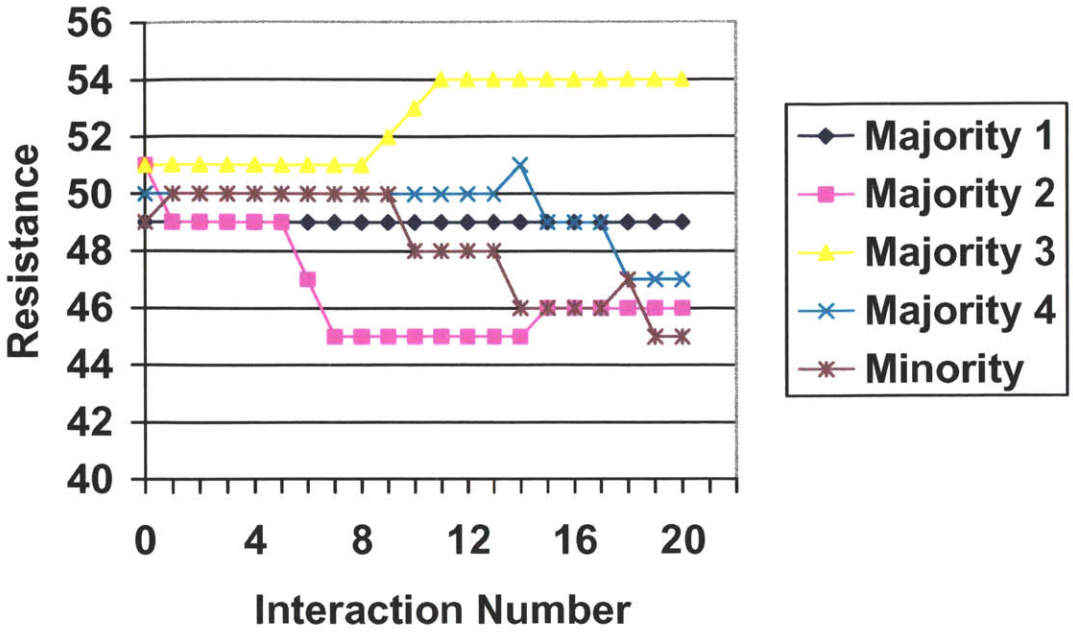


Figure 14: Resistance after each interaction in a run in which the minority agent was successful

When possible to identify the agent(s) involved in an interaction, the current opinion of that agent is in bold font

Agent:	**Interaction Number**																				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Majority Agent 1	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	**[6,** **2,** **5,** **4,** **3,** **1]**	[6, 2, 5, 4, 3, 1]	[6, 2, 5, 4, 3, 1]	[6, 2, 5, 4, 3, 1]	[6, 2, 5, 4, 3, 1]	[6, 2, 5, 4, 3, 1]	[6, 2, 5, 4, 3, 1]	[6, 2, 5, 4, 3, 1]	**[2,** **6,** **5,** **4,** **3,** **1]**	[2, 6, 5, 4, 3, 1]	[2, 6, 5, 4, 3, 1]	[2, 6, 5, 4, 3, 1]	[2, 6, 5, 4, 3, 1]
Majority Agent 2	[6, 5, 4, 3, 2, 1]	**[6,** **5,** **4,** **3,** **2,** **1]**	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	**[6,** **5,** **4,** **3,** **2,** **1]**	**[2,** **6,** **5,** **4,** **3]**	[2, 6, 5, 4, 3]	[2, 6, 5, 4, 3]	[2, 6, 5, 4, 3]	[2, 6, 5, 4, 3]	[2, 6, 5, 4, 3]	[2, 6, 5, 4, 3]	[2, 6, 5, 4, 3]	**[2,** **6,** **5,** **4,** **3]**	[2, 6, 5, 4, 3]	**[2,** **6,** **5,** **4,** **1]**	[2, 6, 5, 4, 3]	[2, 6, 5, 4, 3]	[2, 6, 5, 4, 3]
Majority Agent 3	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 2, 3, 1]	**[6,** **5,** **4,** **2,** **3,** **1]**	**[6,** **2,** **5,** **4,** **3,** **1]**	**[2,** **6,** **5,** **4,** **3,** **1]**	**[2,** **6,** **5,** **4,** **3,** **1]**	[2, 6, 5, 4, 3]	[2, 6, 5, 4, 3]	[2, 6, 5, 4, 3]	[2, 6, 5, 4, 3]	[2, 6, 5, 4, 3]	[2, 6, 5, 4, 3]	[2, 6, 5, 4, 3]	[2, 6, 5, 4, 3]
Majority Agent 4	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	[6, 5, 4, 3, 2, 1]	**[6,** **2,** **5,** **4,** **3,** **1]**	**[6,** **2,** **5,** **4,** **1,** **3]**	[6, 2, 5, 4, 1, 3]	[6, 2, 5, 4, 1, 3]	**[6,** **2,** **5,** **4,** **3,** **1]**	**[6,** **2,** **5,** **4,** **3,** **1]**	[2, 6, 5, 4, 3, 1]
Minority Agent	[2, 1, 6, 5, 3, 4]	**[2,** **6,** **1,** **5,** **4,** **3]**	**[2,** **6,** **1,** **5,** **4,** **3]**	[2, 6, 1, 5, 4, 3]	[2, 6, 1, 5, 4, 3]	[2, 6, 1, 5, 4, 3]	**[2,** **6,** **1,** **5,** **4,** **3]**	**[2,** **6,** **1,** **5,** **4,** **3]**	**[2,** **6,** **5,** **4,** **1,** **3]**	[2, 6, 5, 4, 3, 1]	**[2,** **6,** **5,** **4,** **3,** **1]**	**[2,** **6,** **5,** **4,** **3,** **1]**	[2, 6, 5, 4, 3, 1]	[2, 6, 5, 4, 3, 1]	[2, 6, 5, 4, 3, 1]	[2, 6, 5, 4, 3, 1]	[2, 6, 5, 4, 3, 1]	[2, 6, 5, 4, 3, 1]	[2, 6, 5, 4, 3, 1]	**[2,** **6,** **5,** **4,** **3,** **1]**	**[2,** **6,** **5,** **4,** **3,** **1]**	[2, 6, 5, 4, 3, 1]

Table 9: Agent Opinions after each interaction in a single test run in which the minority agent was successful.

Bibliography:

Ferber, Jacques. *Multi-agent Systems: An Introduction to Distributed Artificial Intelligence.* Addison-Wesley Pub. Co. 1999

M. Yuasa, Y. Yasumura, K. Nitta "Negotiation Support Tool Using Emotional Factors", IFSA-NAFIPS 2001 Conference Proceedings, 2001

S. S. Nemani, V. H. Allan. "Agents and the algebra of emotion", AAMAS 2003

McCauley, L., S. Franklin, and M. Bogner. "An Emotion-Based 'Conscious' Software Agent Architecture". In Affective Interactions, Lecture Notes on Artificial Intelligence ed., vol. 1814, ed. A. Paiva. Berlin: Springer, 2000.

M. Prietula, K. Carley, and L. Gasser ed. *Simulating Organizations: Computational Models of Institutions and Groups.* The MIT Press. Cambridge, MA, 1998

Winston, Patrick H. *Artificial Intelligence.* Addison Wesley. Reading, MA, 1993

Russel and Norvig. *Artificial Intelligence, A Modern Approach.* Prentice Hall. Upper Saddle River, NJ, 1995