NegoChat: A Chat-Based Negotiation Agent*

Avi Rosenfeld
Dept. of Industrial Engineering
Jerusalem College of Technology
Jerusalem, Israel
rosenfa@jct.ac.il

Inon Zuckerman
Dept. of Industrial Engineering and Management
Ariel University
Ariel, Israel
inonzu@ariel.ac.il

Erel Segal-Halevi, Osnat Drein, Sarit Kraus
Dept. of Computer Science
Bar-Ilan University
Ramat-Gan, Israel
erelsgl@gmail.com, osnatairy@gmail.com, sarit@cs.biu.ac.il

ABSTRACT

To date, a variety of automated negotiation agents have been created. While each of these agents has been shown to be effective in negotiating with people in specific environments, they lack natural language processing support required to enable real-world types of interactions. In this paper we present NegoChat, the first negotiation agent that successfully addresses this limitation. NegoChat contains several significant research contributions. First, we found that simply modifying existing agents to include an NLP module is insufficient to create these agents. Instead, the agents' strategies must be modified to address partial agreements and issue-by-issue interactions. Second, we present NegoChat's negotiation algorithm. This algorithm is based on bounded rationality, and specifically Aspiration Adaptation Theory (AAT). As per AAT, issues are addressed based on people's typical urgency, or order of importance. If an agreement cannot be reached based on the value the human partner demands, the agent retreats, or downwardly lowers the value of previously agreed upon issues so that a "good enough" agreement can be reached on all issues. This incremental approach is fundamentally different from all other negotiation agents, including the state-of-the-art KBAgent. Finally, we present a rigorous evaluation of NegoChat, showing its effectiveness.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Human Factors; Experimentation; Performance

Keywords

Human-Agent Systems; Negotiation; Chat Agent

1. INTRODUCTION

Negotiation is a basic task that forms a basic element in our daily lives. We often find ourselves in important situations, whether simple or complex, that require negotiations. Most negotiations are mundane, such as haggling over a price in the market, deciding on a meeting time, or even convincing our children to eat their vegetables. However, they can also have colossal effects on the lives of millions, such as negotiations involving inter-country disputes and nuclear disarmament [8].

To date, a variety of agents have been created to negotiate with people within a large spectrum of settings including: the number of parties, the number of interactions, and the number of issues to be negotiated [3, 9, 12, 11, 14]. As the next section details, two key common elements exist throughout all of these previous agents. First, these agents are all based on the assumption that the human negotiators use bounded rationality. People did not successfully reach agreements with agents based on notions of equilibrium or optimal methods, and thus alternatives needed to be found for all agents [11]. Second, all agents needed mechanisms for dealing with incomplete information. This is typically done through reasoning about the negotiating partners by learning their preferences and strategies [15]. Unfortunately, Natural Language Processing (NLP) abilities are lacking from current state-of-the-art negotiation agents – something that has been previously noted [11]. This inherent limitation requires these agents to "force" their human counterparts to interact via menus or other non-natural interfaces.

This paper presents NegoChat, an agent that contains the following three key contributions: First, NegoChat successfully incrementally builds agreements with people, something current automated negotiators do not do. Second, NegoChat integrates natural language into its agent, allowing people to practice his or her negotiation skills from anywhere, without installing any complicated software. Third, Negochat performs better than the current state-of-the art agent, achieving better agreements in less time. Users are also happier with NegoChat and think the agent is fairer.

This paper is structured as follows. We first provide background of previous automated negotiation agents and their limitations in Section 2. We also present an overview of Aspiration Adaptation Theory (AAT) [17] upon which several key elements of NegoChat are built. Section 3 presents an overview of the negotiation domain and the KBAgent [14]. NegoChat also integrates key elements of this state-of-the-art agent, but as Section 4 describes its negotiation algorithm builds agreements incrementally based on AAT. Section 5 presents how NegoChat generates and responds to natural language, allowing people to practice negotiation in this more intuitive format. Section 6 explores the strengths and weaknesses of NegoChat, including results supporting that it achieves significantly better agreements in less time than the current state-of-the-art KBAgent. Section 7 concludes and provides future directions.

*This work was supported in part by ERC grant #267523, ARO grants W911NF0910206, W911NF1110344 and U.S. Army Research Lab and Research Office MURI grant number W911NF-08-1-0144. Sarit Kraus is also affiliated with UMIACS.

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

2. RELATED WORK

This paper's main contribution lies in describing NegoChat, what we believe is the first negotiation agent successfully developed to use a natural chat interface while considering its impact on the agent's negotiation strategy. Extensive studies in the field of Human Computer Interactions (HCI) have noted that the goal of any system should be an intuitive interface with the stress being put on creating agents which operate in environments which are as real and natural as possible [6, 7]. Thus, following these approaches, it is critical to develop natural language support for negotiation agents to allow for these types of "normal" interactions [10]. This form of typing as natural interaction is referred to as Natural-Language Interaction (NLI) in the literature. There have been numerous informal tests of NLI systems, but few controlled experimental comparisons against some other design [18].

Automated Negotiation Agents

While automated negotiation agents have been developed for quite some time, even state-of-the-art negotiation agents unfortunately do not yet consider how natural language impacts the agent's strategy. Examples include Byde et al.'s *AutONA* automated negotiation agent [3]. Their problem domain involves multiple negotiations between buyers and sellers over the price and quantity of a given product. Jonker et al. [9] created an agent to handle multi-attribute negotiations which involve incomplete information. However, none of these agents support NLI, and a different interface might profoundly impact performance. In contrast, Traumet et al. [21] presented a model of conversation strategies which they implemented in a virtual human to help people learn negotiation skills. Their virtual human can negotiate in complex adversarial negotiation with a human trainee. Their main effort was toward the Dialogue Manager (DM) module and the virtual character module. Their strategy module used rule-based strategies. In contrast, we focus less on the DM and do not create a virtual character. However, we do study how the agent's strategy must be modified to incorporate NLP, and conduct an exhaustive experimental study how the strategy we develop is better than the state-of-the-art. As their focus in not on the agent's strategy, they do not report such a test. Another advantage of our agent is its ability to be used by anyone that wants to practice her negotiation skills, from anywhere, without installing any complicated software.

The *QOAgent* [12] is a domain independent agent that can negotiate with people in environments of finite horizon bilateral negotiations with incomplete information. The negotiations consider a finite set of multi-attribute issues and time-constraints. The *KBAgent* currently represents the state-of-the-art automated negotiation agent. It also considers negotiations with a finite set of multi-attribute issues and time-constraints, and has been shown to be the most effective agent in achieving agreements with people in several domains involving multiple attributes [14]. As NegoChat is built on several elements of this agent, we provide an overview of how it operates, including how it learns from previous data and how it decides if it should accept proposals.

Please note that multi-attribute negotiation continues to be an important research challenge, with one active research avenue being the ANAC (Automated Negotiating Agents Competition) Workshop. However, please note that even to date, this competition focuses on agent-agent interactions and the interface supports only menu-based interactions between agents and people. Much research within this field studies exclusively agent to agent negotiations.

It was previously shown [23] that existing negotiation algorithms must be modified to support robust real-world interfaces with people, such as chat. Specifically, it was shown that simply adding chat support to the current state-of-the-art KBAgent was not effective. To address this limitation, we study what logical extensions are needed, if any, to make existing negotiation agents suitable for natural language. Previously economic and behavior research into people's negotiation would suggest that the current agent approach of attempting an agreement on all issues simultaneously will not be effective. For example, Bac and Raff [1] found that simultaneously negotiating a complete package might be too complex for individual buyers. Furthermore they show that, in the context of incomplete information with time discount, the more informed player ("strong" in their terminology) will push towards issue-by-issue negotiation. Busch and Horstmann [2] found that some people might like to decide all issues at once, while others prefer to decide one by one. Chen [5] studied issue-by-issue negotiation with opt-out factor, and argues that when the opt-out probability is low, agent prefer to negotiate a complete package because intuitively we know that the negotiations can last long enough so that agents can get to a "win-win" situation. However, with high opt-out probability, agents prefer issue-by-issue negotiation. Sofer et al. have shown that an alternative negotiation scheme which takes an issue-by-issue approach can theoretically form the best protocol within two self-interested rational agents [20]. Thus, one key contribution of this paper is its study as to how people react to agents that do not propose issue-by-issue agreements.

Aspiration Adaptation Theory

Given the realization that people prefer to negotiate issue-by-issue, we present NegoChat, a negotiation algorithm that successfully does this. NegoChat is built upon Aspiration Adaptation Theory (AAT) [17]. Following our previous work, we found that people use key elements from AAT in their negotiations, even when optimal solutions, machine learning techniques for solving multiple parameters, or bounded techniques other than AAT could have been implemented [16]. The premise of AAT is that certain decisions, and particularly our most complex decisions, are not readily modeled based on standard utility theory. For example, assume you need to relocate and choose a new house to live in. There are many factors that you need to consider, such as the price of each possible house, the distance from your work, the neighborhood and neighbors, and the schools in the area. How do you decide which house to buy? Theoretically, utility based models can be used. However, many of us do not create rigid formulas involving numerical values to weigh trade-offs between each of the search parameters. AAT is one way to model this and other similar complex problems.

Despite its lack of utility to quantify rational behavior, AAT is an approach by which bounded agents address a complex problem, $\mathcal{G}$. The complexity within $\mathcal{G}$ prevents the problem from being directly solved, and instead an agent creates m goal variables G_1, ..., G_m as means for solving $\mathcal{G}$. Agents decide how to set the goal variables as follows: The m goal variables are sorted in order of priority, or the variables' *urgency*. Accordingly, the order of G_1, ..., G_m refers to goals' urgency, or the priority by which a solution for the goal variables should be attempted. Each of the goal variables has a desired value, or its *aspiration level*, that the agent sets for the current period. Note that the agent may consider the variable "solved" even if it finds a sub-optimal, but yet sufficiently desired value. The agent's search starts with an initial aspiration level and is governed by its *local procedural preferences*. The local procedural preferences prescribe which aspiration level is most urgently adapted upward if possible, second most urgently adapted upward if possible, etc., and which partial aspiration level is *retreated from* or adapted downward if the current aspiration level is not feasible. Here, all variables except for the goal variable being addressed are assigned values based on ceteris paribus, or all other goals being equal, a better value is preferred to a worse one.

For example, referring to the above example of searching for a new home, one would like find the biggest home, for the cheapest price, in the best location. However, if this is found to not be possible, the agent may retreat from, or settle on some of its goals, say by selecting a smaller home, yet satisfy the other goals, such as the home's price and location. AAT provides a framework for quantifying this process. Accordingly, a person might form an *urgency* of the house's price as being the most important goal variable, its size as being the second most important variable, and its location between the third most important variable. Agents, including people, create an *aspiration level* for what constitutes a "good enough" value for these variables. Borrowing from Simon's terminology [19] there is only an attempt to "satisfice" the goal values, or achieve "good enough" values for these values instead of trying to optimize or truly solve them. If it is not possible to achieve all aspirations, the original values will need to be reexamined, or retreated from. Thus, the agent may retreat, or reconsider raising the budget they allocated for buying their house, the size they are willing to settle on, or its location. We now describe NegoChat's negotiation algorithm, and specify how AAT is used.

3. THE NEGOTIATION DOMAIN

The negotiation environment we consider can be formally described as follows: We study *bilateral* negotiation in which two agents negotiate to reach an agreement on conflicting issues. The negotiation can end either when (a) the negotiators reach a full or partial agreement, (b) one of the agents opts out (denoted as OPT), thus forcing the termination of the negotiation with a predefined opt-out outcome, or (c) a time limit is reached, that results in a predefined status-quo outcome (denoted as SQ).

The negotiations resolve around *multi-attribute* characteristics. There is a set of n issues, denoted as I, and a finite set of values, $domain(i)$ for each issue $i \in I$. An agreement is denoted as $\alpha \in O$, and O is a finite set of values for all issues. The negotiations are sensitive to the negotiation time that impacts the benefits of the negotiating parties. Formally, we define the negotiation rounds $Rounds = \{0, ..., dl\}$, where dl is the deadline limit and typically use R to denote the current round. The time duration of each round is fixed and known to both negotiators. Each agent is assigned a time cost which influences its benefits as time passes. The time effect may be negative or positive.

The negotiation *protocol* is fully flexible. As long as the negotiation has not yet ended, each side can propose a possible agreement, reject a previously offered agreement, opt-out of the negotiation, or communicate a threat, promise, or any general remark. In contrast to the model of alternating offers [13], each agent can perform up to $M > 0$ interactions with the opponent agent in each round.

Last, we consider environments with *incomplete information*. That is, agents are not fully aware of the scoring structure of their opponents. We assume that there is a finite set of scoring structures which will be referred to as agent types. For example, one type might model a long term orientation regarding the final agreement, while another might model a more constrained orientation. Formally, we denote the possible types of the agents $Types = \{1, ..., k\}$. Given $l \in Types$, we refer to the scoring of the agent of type l as S_l, and $S_l : \{O \cup \{SQ\} \cup \{OPT\}\} \times Rounds \rightarrow \mathcal{R}$. Each agent is fully aware of its own scoring function, but it does not know the exact type of its negotiating partner.

The KBAgent

The state-of-the-art automated negotiator for the above environment is the *KBAgent* [14]. It has been shown that the *KBAgent* negotiates efficiently with people and achieves better utility values than other automated negotiators. Moreover, the *KBAgent* achieves significantly better agreements, in terms of individual score, than the human counterparts playing the same role.

The main difference between the *KBAgent* and other agents is its inherent design, which builds a general opponent model. *KBAgent* utilizes past negotiation sessions of other agents as a knowledge base for the extraction of the likelihood of acceptance and offers which will be proposed by the other party. That data is used to determine which offers to propose and what offers to accept. One of its main advantages is that it can also work well with small databases of training data from previous negotiation sessions.

In order to generate an offer, the *KBAgent* creates a list ordered by the QOValue, which is an alternative to the Nash bargaining solution (see [12] for the exact definition). *KBAgent* first proposes the offer with the maximal QOValue. The subsequent offers are picked from the ordered list based on the concession rate the *KBAgent* applies and are chosen with a decreasing QOValue for the agent and an increasing utility value for the other party. To decide which offers to accept, the *KBAgent* determines a time dependent threshold to decide whether to accept or reject an offer. In order to decide on the optimal threshold, the probabilities learned from the database of past negotiations are used. The time dependent thresholds may depend on the *KBAgent* belief on its opponent's type. For space reasons we do not get into it in this paper, but see [14] for additional discussion. Given these thresholds, the KBAgent generates a list of concession offers to be made for every round of the negotiation. The concessions offers that are generated change with regard to the round number as the agent is more willing to reach less beneficial agreements as the round increases. There are two disadvantages to this approach. First, the KBAgent only proposes one concession offer per round, and if this offer is rejected, it does not attempt to generate a similar offer during the current round. For the domain they studied, this round was 2 minutes, which we found constituted lost opportunities to reach agreements. Thus, if the other side rejects the one offer the KBAgent provides during a given round, an agreement can only be reached if the other side takes the initiative to offer an alternate above the learned threshold. Second, the KBAgent never offers partial offers. We found this was a disadvantage as people found it difficult only to receive full offers.

4. THE NEGOTIATION ALGORITHM

The goal of the NegoChat agent is to reach an agreement, α, that maximizes its own score. Following AAT terminology, NegoChat creates n goal variables, $G_1, \ldots, G_n$, one for each issue in the negotiation issues set I. For each goal variable G_k we associate domain values $domain(G_k)$ setting it to $domain(k)$. We use V_k to denote a value in $domain(G_k)$, α_k to denote the value of G_k according to α and $S(\alpha, R)$ to denote the agent's score from α at round R.

Each value $V_1, \ldots, V_n$ is originally initialized to NULL to represent that no agreement initially exists on the values within G_k. As the two sides negotiate, values from $V_1, \ldots, V_n$ become non-null. An offer is defined as a *full offer* if it has values for $V_1, \ldots, V_n$. An offer is a *partial offer* if $V_1, \ldots, V_n$ contains between 1 and n -1 non-null values. We assume that an agreement must have non-null values for all values, $V_1, \ldots, V_n$. Note that while the agent's goal is to maximize its own score, an agreement will likely only be achieved through compromise with the person it is negotiating with as the person is self-interested and also wishes to maximize the score of her offer.

As per AAT, $G_1, \ldots, G_n$, are ordered by their aspiration scale, whereby the n issues that need to be negotiated are ordered by how important these issues are to agent. This ordering is pushed onto a stack, A, where the most aspired for issue (G_k) is at the top of the stack, the second most aspired for issue is the second position from the top, etc. In our approach, we choose this ordering based on previous historical data of what issues were typically discussed first, second, third, etc. However, if the aspiration scale is not known, the agent randomly chooses an aspiration scale.

Throughout the course of the negotiation process, the agent must generate offers and decide how to respond to offers. In both cases, as the agent and the person agree upon a value V_k, it pushes this value onto a stack B. In generating offers, the agent pops the last value from A to determine which goal variable G_k should be discussed next. An agreement is reached when A is empty as this indicates that no more issues need to be discussed. The agent must then search for an offer V_k, which represents a proposed value for G_k. Note that in this approach, only partial offers are given, as V_k is only one value. The search for V_k is executed as follows: Given the existing non-null values for V_1, ..., V_n, search for highest value for G_k within the **search cluster**, a concept that is based on the *KBAgent's* concession list [14].

While the output of the KBAgent's learning process is an ordered concession list that should be used during the negotiation session, the list itself contains only full offers, one for each round, with gradually decreasing QOValues. As NegoChat can make concessions of partial offers we define a search cluster based on all possible full offers within an epsilon deviance for the QOValue for this round that yields an equal or higher score to the agent. In particular, given the KBagent's offer for round R, α_R, we define the **search cluster**, $SC(R)$ for round R as all offers which provide NegoChat a score of at least $S(\alpha_R, R)$ but still yield its opponent a score not lower than $S_{opponent}(\alpha_R, R) - \epsilon$.

Algorithm 1 NegoChat

```
1: Create stacks A and B, A ← aspiration scale, B ← ∅
2: while $R$ <= DEADLINE do
3:     SC ← SearchCluster($R$)
4:     while NOT(AGREEMENT) do
5:         if AGREEMENT OR OPT-OUT then
6:             EXIT
7:         if Received-Offer($V_{k_1}$, ..., $V_{k_m}$) then
8:             HandleOffer($V_{k_1}$, ..., $V_{k_m}$)
9:         else
10:            ProposeOffer(1, Top(A))
```

We present three algorithms that comprise NegoChat's strategy, which we now explain in greater detail. Algorithm 1 provides the base of the agent, and controls how the agent handles offers it receives and generates offers. This is done by keeping two stacks, A and B. Stack A contains the aspiration scale of $G_1, \ldots, G_n$ and represents the order by which the agent should negotiate issues. Stack B is initially set to be empty and represents issues that have been resolved (line 1). The remainder of the agent (lines 2–10) controls the timing of the agent and how it creates and responds to offers. While the negotiation deadline has not been reached, the agent first creates SC, its search cluster. Every round, the agent recalculates the same search cluster until the negotiation deadline is reached (lines 2 or 3). This loop is terminated before this point, either by the sides reaching an agreement or the other side opting out (lines 4-6). As our agent never opts out, this can only happen from the person's initiative. As opposed to automated agents which can easily handle thousands if not millions of request every time negotiation round, we found by trial-and-error that people do not appreciate more than 1 offer more than every 25 seconds. Thus, while the agent might immediately propose an offer for the most aspirated for goal value (line 10), it will wait for counter-offers or other proposals for another 25 seconds. During this time, Algorithm 2 handles these offers (lines 7–8).

Algorithm 2 HandleOffer(V_{k_1}, ..., V_{k_m})

```
1: if ∃ ($\alpha \in$ SC s.t. $\forall k_i, \alpha_{k_i} = V_{k_i}$ AND $\forall V_i \in B$, $\alpha_i = V_i$) then
2:     Send-Accept($V_{k_1}, \ldots, V_{k_m}$)
3:     Push($V_{k_1}, \ldots, V_{k_m}$, B)
4:     Remove($G_{k_1}, \ldots, G_{k_m}$, A)
5: else
6:     Send-Reject($V_{k_1}$, ..., $V_{k_m}$)
7:     ProposeOffer(m, $G_{k_1}, \ldots, G_{k_m}$)
```

Algorithm 2 receives as its input an offer α for m goals variables, $G_1, \ldots, G_m$, which have the corresponding values V_{k_1}, ..., V_{k_m}. Note that m can be the trivial case of 1 issue or n representing all issues. Assuming that every value for every goal issue is contained within the search cluster (line 1), this is an acceptable offer, and the agent responds with an accept message. The NLP module described in the next section is responsible for generating a natural language response. Once an agreement is reached, the agreed upon values, V_{k_1}, ..., V_{k_m}, are added to the stack B (line 3), and the agreed upon issues, G_{k_1}, ..., G_{k_m} are removed from stack A (line 4).

Algorithm 3 ProposeOffer(m, $G_{k_1}, \ldots, G_{k_m}$)

```
1: if SC ≠ ∅ then
2:     $\alpha \leftarrow \arg\max \{S(\alpha', R)\}$ | $\alpha' \in$ SC, $\forall V_i \in B$, $\alpha_i = V_i\}$
3:     if $\alpha \neq$ NULL then
4:         Send-Offer($\alpha_{k_1}, \ldots, \alpha_{k_m}$)
5:         if Received-Accept($\alpha_{k_1}, \ldots, \alpha_{k_m}$) then
6:             Push($\alpha_{k_1}, \ldots, \alpha_{k_m}$, B)
7:             Remove($G_{k_1}, \ldots, G_{k_m}$, A)
8:         if Received-Reject($\alpha_{k_1}, \ldots, \alpha_{k_m}$) then
9:             SC = SC \ {$\alpha' \in$ SC | $\forall k_i$ $\alpha'_{k_i}$=$\alpha_{k_i}$ AND $\forall V_i \in B$, $\alpha_i$ = $V_i$}
10:    else
11:        $V_{last}$ ← Pop(B)
12:        Push($G_{last}$, A)
```

Algorithm 3 creates the agent's offers. Note that when this algorithm is typically called by the agent (line 8 of Algorithm 1), it only proposes one issue at time ($m = 1$). The agent will only propose offers with multiple issues in response to the person's offer (line 7 of Algorithm 2). In either case, assuming an offer can be found (line 3), then the proposal is sent as natural language (line 4). Specifics of this module are again described in the next section. Assuming this search cluster is empty, there is nothing for the agent to do, except wait for the next time R, during which the search cluster will be recalculated. Assuming the search cluster contains different possibilities, it offers the agreement with the highest score from within the search cluster. If this offer is accepted, the agent adds these values to B (line 6) and removes these issues from A (line 7). Otherwise, the agent removes all offers with this value (line 9). For example, again assume you wish to buy a house and are in negotiations to purchase a specific house. Your goal variables are: price, payment schedule and occupancy date, and your agent offers a price of 200,000 Euro for a house. If the other side rejects your offer, the agent assumes that all agreements with the same price will also be rejected. Thus, it removes these similar agreements from the search cluster (line 9). If in another example, if you already agreed on the price of 180,000 EURO and your agent offered the seller a payment plan with 4 payments over the next 4 months, and the offer was rejected, NegoChat will remove from the search clusters only the agreements that includes both the price of 180,000 and 4 payments. Next, NegoChat will look for a different payment schedule to offer consistent with 180,000 EURO (back at line 2). If there is no additional agreement in the search cluster that includes 180,000 EURO for the price, it will revisit the last agreed upon issue (line 11) and then explore it again as the next issue to pursue (line 12). Referring back to our example, if the person accepted a bid of 180,000 Euro for a house and with this price the agent agrees to a payment plan with 4 payments, once the offer of 4 payments is rejected, NegoChat will renegotiate the price.

5. NEGOCHAT'S NATURAL LANGUAGE

Our natural language system has a standard dialog system architecture, described in Figure 1. We illustrate the system with a running example from our experimental domain, where the human is an employer and the agent is a job-candidate, and they negotiate over the candidate's job conditions (described further in the next section). Nonetheless, the system itself is general and can be applied to support chat in any system.

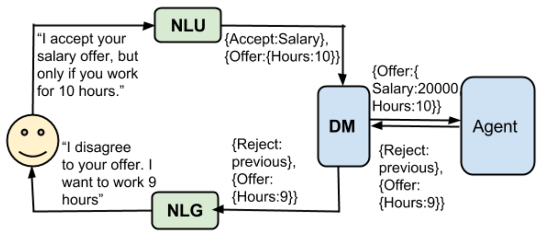
The natural language system is composed of several components. The **Natural Language Understander** (NLU) translates the human sentences from natural language to a set of *dialog acts* that represents the user intentions. We represent our dialog acts in the standard JSON format (www.json.org).For example, the human utterance "I accept your salary offer, but only if you work for 10 hours", is translated to a set of two dialog acts: [[{Accept:Salary}, {Offer:{Hours:10}}]]. The NLU is described in detail in Subsections 5.1 and 5.2.

The **Dialog Manager** (DM) has several responsibilities: First, it interprets the human dialog acts based on the current dialog state. For example, it interpretes the dialog act {Accept:Salary} based on the salary value in the most recent offer made by the agent, and converts it to an explicit Offer. Second, it responds to human dialog acts that are not directly related to negotiation, such as greetings and questions. Third, it notifies the agent when the human dialog acts are related to negotiation. For example, if one of the human's dialog acts in an offer, then the DM sends a "Received-Offer" notification (see Algorithm 1), and if the human has accepted a full offer, the DM sends a "Received-Accept" notification (see Algorithm 3). Fourth, it controls the timing of conversation. For example, if the human hasn't done anything in a pre-specified time interval (e.g. 25 seconds), then the DM asks the agent to make an action, e.g., repeat the previous offer or make a new offer. Fifth, it receives commands from the agent, and translates them to dialog acts. For example, if the agent issues a Send-Reject command (from Algorithm 2), then the DM creates the dialog act {Reject:previous}. Finally, it combines several dialog acts to a single set. For example, if the agent issues a Send-Reject command, and shortly after that, a Send-Offer command (from Algorithm 3), then the DM creates a set with two dialog acts [{Reject:previous}, {Offer:...}].

The **Natural Language Generator** (NLG) translates the set of dialog acts, created by the DM, to a single natural language sentence, that is sent to the human. Our NLG works in cooperation with our NLU in order to create human-like sentences, as we describe in detail in Subsection 5.3.

5.1 Natural Language Understander (NLU)

Our NLU component is a **multi-label classifier** (MLC) - a classifier that returns a set of zero or more labels for each input sample.



NLU
"I accept your salary offer, but only if you work for 10 hours."
{Accept:Salary}, {Offer:{Hours:10}}
{Offer:{ Salary:20000 Hours:10}}
DM
Agent
"I disagree to your offer. I want to work 9 hours"
{Reject: previous}, {Offer: {Hours:9}}
{Reject: previous}, {Offer: {Hours:9}}
NLG

Figure 1: Dialog System Architecture. Example starts at the top-left corner.

The set of possible labels is the set of dialog acts recognized by our DM, whose total number is 58. They have a hierarchical structure, for example: {Offer:{Salary:20000}} and {Offer:{Hours:9}} are two different dialog acts. The top level of the hierarchy contains 8 different labels: {Offer, Accept, Reject, Append, Insist, Query, Quit, Greet}. In order to take advantage of the hierarchical structure of the dialog acts, we used the HOMER approach (Hierarchy Of Multi-label classifiERs, [22]). In this approach, there is a different MLC for each level of the hierarchy. The input sentence is first sent to the top-level MLC, which returns a subset of the top-level labels, e.g., {Offer, Query}. Then the sentence is sent in parallel to all relevant second-level MLCs, e.g., the Offer MLC and the Query MLC. The Offer MLC returns a set of second-level labels from the set relevant to Offer (i.e. Salary, Hours, etc.), and the MLC for Query returns a set of second-level labels from the set relevant to Query. This process continues until the leaves of the hierarchy are reached. Then, the replies of all MLCs are combined to produce the final set of dialog acts.

For the MLCs in each node of the HOMER, we used the One-versus-All approach: each MLC is a collection of binary classifiers, one for each label. For each input sentence, it runs each binary classifier in turn, and returns the set of labels whose classifier returned "true". As the base binary classifier, we used Modified Balanced Winnow [4] - a classifier that supports online training and real-time classification. In preliminary experiments we tried several other state-of-the-art MLCs, and several other binary classifiers, and the configuration described here performed best.

An input sentence goes through several pre-processing components before it arrives at the MLC. The **normalizer** converts numbers and other common phrases in the input sentence to canonical format. The **splitter** splits the sentence around punctuation marks and creates several sub-sentences. We found out that this simple heuristic greatly improves the performance of the MLC. The **feature extractor** creates a feature vector from each sub-sentence. As features, we use unigrams and bigrams (pairs of adjacent words).[1] As feature values we use the standard TF/IDF metric. The resulting feature vectors are the inputs to the MLC.

5.2 Development and Training

As a first step in adding natural language capabilities, we manually wrote a single natural language sentence for each dialog act supported by the agent. This facilitated the coordination between the team working on the agent and the team tagging the training data, and made sure they both understand the negotiation acts in the same way. We also used these sentences as an initial training set for the Multi-Label Classifier (MLC).

Using this initial NLU component, we let our agent speak with students and Amazon Mechanical Turk workers. During these preliminary experiments, one of the developers acted as a "Wizard-of-

[1]We tried more sophisticated features, such as pairs of non-adjacent words, but this didn't improve performance.

Oz": through a web-based GUI, he viewed each set of dialog acts produced by the NLU component, and could edit it before it is sent to the DM. He could also immediately train the classifier with each new sentence, thanks to its fast training abilities. During the online learning process, the sentence-level accuracy of the NLU componenet improved from 18% (with only the initial 58 manually-written sentences) to 72% (with 775 tagged sentences).[2]

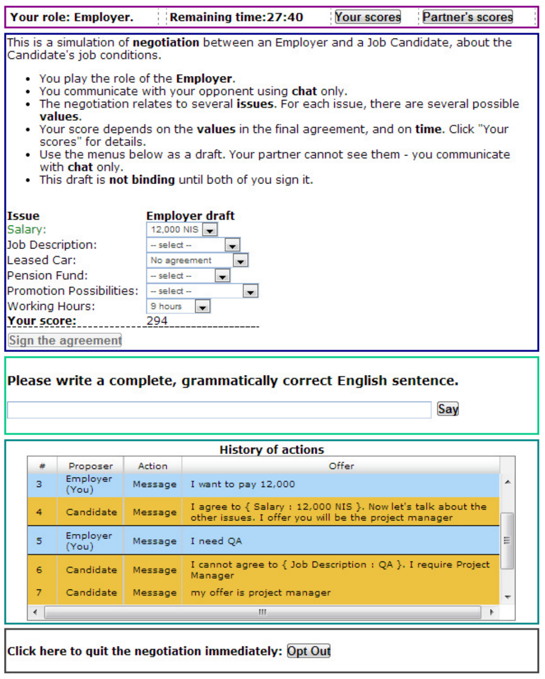
5.3 Natural Language Generator (NLG)

The NLG takes as input a set of dialog acts produced by the DM, and returns a natural language sentence that is sent to the human. Usually, NLGs are based on manually-written templates. In contrast, our NLG uses the NLU's training data in the reverse direction. For each dialog act, the NLG asks the NLU for a sentence tagged with exactly this dialog act, and combines the received sentences to a single output sentence. This approach has several advantages, which we exemplify with several actual examples from our experiments: First, the agent's replies are versatile, even when the strategy demands that it repeats the same offer again and again. For example: the agent says "I would like to work for 20,000", and 25 seconds later, "I need to make 20,000". Second, the agent's replies are human-like. They even contain spelling and grammar mistakes that occur naturally in chat conversations between humans. Third, some of the agent's replies contain reasoning and argumentation. For example: "i would like a 20000 salary. this is mandatory to me to have a good salary as i believe working conditions affect directly my effectiveness" (sic). Last, the agent continuously learns new ways to express itself during the NLU's online learning process.

6. EXPERIMENT DESIGN AND ANALYSIS

The main goal of this research was to push the envelope of automated negotiators research by moving from menu-driven interfaces to chat based environments. As this work transitions from the fruitful work of previously developed agents, we intentionally chose to base ourselves on these agents and the complex environments they had studied. Thus, we shied away from dealing with overly simplified settings, such as those with full information, single issues, or alternating turn based offers, and instead considered a complex problem with partial information, multi-attribute negotiations, and an unconstrained interaction protocol.

In order to properly evaluate the influence of natural language input on automated negotiation agents, we picked the *job candidate* domain used in previous research [12, 14]. In this domain, a negotiation takes place after a successful job interview between an employer and a job candidate. In the negotiation both the employer and the job candidate wish to formalize the hiring terms and conditions of the applicant. Below are the issues under negotiation: **[Salary]**:This issue dictates the total net salary the applicant will receive per month. The possible values are {7000, 12000, 20000}. **[Job description]**:This issue describes the job description and responsibilities given to the job applicant. The possible values are {QA, programmer, team manager, project manager}. **[Social benefits]**:The social benefits are divided into two categories: company car and the percentage of the salary allocated, by the employer, to the candidate's pension funds. The possible values for a company car are {leased car, no leased car, no agreement}. The possible value for the percentage of the salary deposited in pension funds are {0%, 10%, 20%, no agreement}. **[Promotion possibilities]**: This issue describes the commitment by the employer regarding the fast track for promotion for the job candidate. The possible values are {fast promotion track (2 years), slow promotion track (4 years), no agreement} **[Working hours]**: This issue describes the number of working hours required by the employee per day (not including over-time). The possible values are {8 hours, 9 hours, 10 hours}.



Your role: Employer. | **Remaining time:27:40** | Your scores | Partner's scores

This is a simulation of **negotiation** between an Employer and a Job Candidate, about the Candidate's job conditions.

- You play the role of the **Employer**.
- You communicate with your opponent using **chat** only.
- The negotiation relates to several **issues**. For each issue, there are several possible **values**.
- Your score depends on the **values** in the final agreement, and on **time**. Click "Your scores" for details.
- Use the menus below as a draft. Your partner cannot see them - you communicate with **chat** only.
- This draft is **not binding** until both of you sign it.

Issue	**Employer draft**
Salary:	12,000 NIS
Job Description:	-- select --
Leased Car:	No agreement
Pension Fund:	-- select --
Promotion Possibilities:	-- select --
Working Hours:	9 hours
Your score:	294

Sign the agreement

Please write a complete, grammatically correct English sentence.

Say

History of actions

#	Proposer	Action	Offer
3	Employer (You)	Message	I want to pay 12,000
4	Candidate	Message	I agree to { Salary : 12,000 NIS }. Now let's talk about the other issues. I offer you will be the project manager
5	Employer (You)	Message	I need QA
6	Candidate	Message	I cannot agree to { Job Description : QA }. I require Project Manager
7	Candidate	Message	my offer is project manager

Click here to quit the negotiation immediately: Opt Out

Figure 2: The negotiation system's interface.

In this scenario, a total of 1296 possible agreements exist ($3 \times 4 \times 12 \times\ 3 \times 3 = 1296$). Each turn in the scenario equates to two minutes of the negotiation, and the negotiation is limited to 15 rounds of 2 minutes each (30 minutes total). If the sides do not reach an agreement by the end of the allocated time, the job interview ends with the candidate being hired with a standard contract, which cannot be renegotiated during the first year. This outcome is modeled for both agents as the status quo outcome. Each side can also opt-out of the negotiation if it feels that the prospects of reaching an agreement with the opponent are slim and it is impossible to negotiate anymore. Opting out by the employer entails the postponement of the project for which the candidate was interviewing, with the possible prospect of its cancelation and a considerable amount of expenses. Opting-out by the job candidate will make it very difficult for him to find another job, as the employer will spread his/her negative impression of the candidate to other CEOs of large companies. Time also has an impact on the negotiation. As time advances the candidate's score decreases, as the employer's good impression has of the job candidate decreases. The employer's score also decreases as the candidate becomes less motivated to work for the company. To facilitate incomplete information there are 3 possible score structures for each side, which models a long term candidate, short term candidate and compromising candidate.

6.1 Experiment design

For our experiment we developed a negotiation system with a chat interface and natural language processing mechanism (see Figure 2). We have implemented both agents in this system: the current state-of-the-art *KBAgent* and the newly developed *NegoChat*

[2]Sentence-level accuracy is the number of sentences whose classification was exactly correct (i.e. the set of dialog acts returned by the MLC is identical to the correct set), divided by the total number of sentences. The 72% accuracy was calculated using 5-fold cross-validation on the set of 775 tagged sentences.

agent.[3] To decide upon the aspiration scale we first ran 16 trials of two people negotiating with each other within the system (32 people total). We noted that on average people discussed issues in the order of: Salary, Job Description, Pension, Working Hours, Car Benefit, and Promotion Possibilities.

We then ran the agent with 46 human participants, most of whom were university students in Israel. These students were motivated through receiving bonus points to their course grade as a function of their final score in the session. The groups were divided randomly to play against either the KB or NegoChat agents and 27 participants played against the KBAgent and 19 against the NegoChat agent. All people played the role of the employer, while the agents played the role of the job candidate.

Before starting the negotiation task, the subjects were given a full tutorial about the task at hand, the interface and the possible score functions. A short test was issued to verify that the participants understood the instructions and task at hand. The participants did not know any details regarding the automated agent with which they were matched, or the fact that it was not a human player. The outcome of each negotiation task was either reaching a full agreement, opting out, or reaching the deadline.

In addition, following each session of the experiments for both interfaces we conducted a post-experiment questionnaire to evaluate the users' satisfaction with both agents. This questionnaire had participants answer the following questions using a scale of 1 (lowest) to 5 (highest): How happy are you with the negotiation's end result? Do you consider the end result to be fair?

6.2 Experimental results

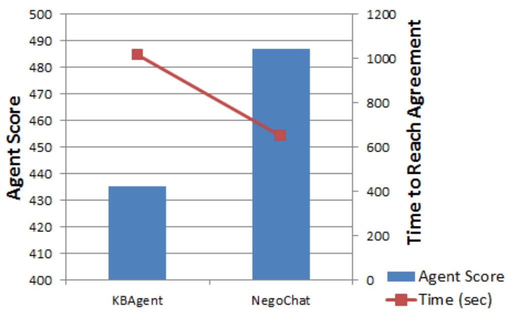


Figure 3: A summary of NegoChat vs. KBAgent results.

The main result, found in Figure 3, is that the NegoChat agent significantly outperformed the state-of-the-art KBAgent. NegoChat achieved on average significantly better scores (p-score 0.036 in a two-tailed t-test, left side Y-axis) and reaches agreements in less time (p-score 0.005 in a two-tailed t-test, right side Y-axis, less time is better). However, in addition to the scores of the agreements themselves, we also analyzed two additional factors, the score of all outcomes including those cases when the person chose to opt-out of the negotiation and the results of the post-experiment user evaluation, the results of which are found in Table 1.

Please note that the NegoChat algorithm does better than the KBAgent in all categories that we analyzed. Since the results for the agent scores including the opting out cases were not normally distributed, here we performed the Mann-Whitney's significance test and still found a score 0.026 for single tailed test implying significance. Not only does NegoChat reach better agreements, but it still performs better when all outcomes are considered including those when the person opted-out. Similarly, people felt that the NegoChat agent was fairer and were happier with the overall agent. We attribute these results to the more intuitive chat interface and its ability to handle partial offers. We posit that NegoChat's more natural interface made its results seem superior, even to people who had no means of comparison with other agents' performance. Additionally, while negotiation is primarily a zero-sum interaction, we found that while subjects' score was lower when interacting with NegoChat than when negotiating with the KBAgent, NegoChat's agreements had a higher social welfare than that of the KBAgent.

We then proceeded to evaluate the NegoChat's NLU (see Section 5.1). To evaluate this component, we employed a human expert that tagged each human sentence from our experiments with its correct set of dialog acts. We then compared the correct set to the set returned by the NLU during the experiments, and calculated the sentence-level accuracy. Some sentences could not have been translated correctly, because they are out-of-domain - their correct meaning is currently not handled by the agent, for example: "What is your work experience?" or "If you will be good you will get better conditions". For each of the two agents, we calculated two accuracy figures: one for only those sentences that could be handled ("In domain"), and once for the entire set of sentences ("All").

Table 2 summarizes the results of this analysis. As expected, the KBAgent required less interaction from the user, as it only works with full offers, full accepts or full rejects. In contrast, NegoChat also adds counter-offers to rejections, accepts partial offers and suggests new issues to discuss, making its language richer. KBAgent typically elicited a smaller range of language, resulting in a smaller average number of utterances in this group (11.3 vs. 28.42 for in-domain utterances, 12.48 vs. 30 overall). The NLU unit was more accurate in the KBAgent games (by either 8 or 4 percent), because of the more limited type of language used by the average participant. These results highlight the centrality of NegoChat's success. Not only did this agent perform better in all categories, but it did so despite a less accurate NLU unit, something that inherently should have **hurt** its performance.

Table 2: The NLU sentence-level accuracy (#correct/#all) in KBAgent games vs. NegoChat games

	Agent	Average #instances	Accuracy
In domain:	**KB**	11.3	195/305=64%
	NegoChat	28.42	305/540=56%
All:	**KB**	12.48	195/337=58%
	NegoChat	30	305/570=54%

To further analyze the impact of the NLU in general, we studied how the KBAgent's performance was impacted by its NLU. To aid in the collection of data, we hired 42 workers from Amazon Turk to participate in this experiments. 21 participants interacted with the KBAgent with the NLU, and 21 people used a "Wizard-of-Oz" approach where an expert manually edited the translations generated by the NLU and corrected them before they were sent to the agent. These results, shown in Table 3, show that when a Wizard of Oz was active, the agent was more successful in all categories: it achieved a higher score, in less time, and people were happier with this agent and thought it fairer. This result implies that if an improved NLU could be generated, the results of NegoChat, and a new generation of Chat Agents we hope it will help create, will be even more significant.

7. CONCLUSIONS AND FUTURE WORK

This paper presents NegoChat, the first negotiation agent that

[3]A copy of the KBAgent and NegoChat agents are found at: `http://u.cs.biu.ac.il:5555/demo`.

Table 1: Results of NegoChat vs. KBAgent Negotiation Experiments

AGREEMENTS	AgentScore	Participants	Time to Reach Agreement	Fairness Standard Deviation	Happiness
KBAgent	435.12, $\sigma = 65.92$	25	1017.4, $\sigma = 401.79$	3.58, $\sigma = 0.88$	3.25, $\sigma = 0.89$
NegoChat	**487**, $\sigma = 64.4$	17	**652.94**, $\sigma = 376.56$	**3.78**, $\sigma = \mathbf{0.97}$	**3.1**, $\sigma = \mathbf{0.99}$
WITH OPT OUTS					
KBAgent	409.56, $\sigma = 111.92$	27 (2)	1001.67, $\sigma = 399.23$	3.56, $\sigma = 0.86$	3.16, $\sigma = 0.98$
NegoChat	**446.05**, $\sigma = 137.02$	19 (2)	**664.74**, $\sigma = 370.08$	**3.76**, $\sigma = \mathbf{0.94}$	**3.19**, $\sigma = \mathbf{1.03}$

Table 3: The NLU Impact on the KBAgent

Interface	AgentScore	Participants	Time to Reach Agreement	Fairness	Happiness
NLU	468.71, $\sigma = 62.68$	21	767.62, $\sigma =$, 326.62	3, $\sigma = 1.21$	2.7, $\sigma = 1.08$
Wizard of Oz	**481.47**, $\sigma = 59.21$	21	**669.76**, $\sigma = 343.614$	**3.3**, $\sigma = 1.28$	**3.3**, $\sigma = 0.86$

considers a natural language interface and its impact on the agent's strategy. In creating NegoChat, we present several novel contributions. First, we describe a new negotiation algorithm based on bounded rationality that facilitates incremental agreements crucial for interacting with people. Second, we present a Natural Language module for interacting with this agent, and describe its originality. Last, we describe extensive experiments highlighting NegoChat's ability to reach significantly better agreements, in less time than the current state-of-the-art KBAgent. We also present results from a user satisfaction survey showing how people were happier with this agent and thought it to be more fair– something we attribute to the agent's more natural interface and ability to generate partial offers. Last, we calculated the accuracy of our natural language understanding unit. We show that it was more difficult to understand humans speaking with NegoChat than humans speaking with the KBAgent. We conjecture that it is because NegoChat itself uses a more versatile natural language than the KBAgent. The success of NegoChat over KBAgent, even when considering the greater difficulty of the task, highlights the challenge NegoChat faced, and further emphasizes its success.

For future work, we hope to study how NegoChat can be applied to additional domains and cultures. In this paper we used participants in Israel who participated in an employer / employee negotiation scenario. An open question which we have begun to study is what extensions, if any, are needed to apply NegoChat to these problems. Furthermore, we hope to analyze if other bounded rationality theories, in addition to Aspiration Adaptation Theory, can be used by agents with natural language interfaces.

8. REFERENCES

[1] M. Bac and H. Raff. Issue-by-issue negotiations: The role of information and time preference. *Games and Economic Behavior*, 13(1):125–134, March 1996.

[2] L.-A. Busch and I. Horstmann. A comment on issue-by-issue negotiations. *Games and Economic Behavior*, 19(1):144–148, April 1997.

[3] A. Byde, M. Yearworth, K.-Y. Chen, and C. Bartolini. AutONA: A system for automated multiple 1-1 negotiation. In *International Conference on Electronic Commerce*, 2003.

[4] V. R. Carvalho and W. W. Cohen. In *KDD*, 2006.

[5] M. K. Chen. Agendas in multi-issue bargaining: When to sweat the small stuff. Technical report, Harvard Department of Economics, Cambridge, November 2002.

[6] M. H. Coen. Design principles for intelligent environments. In *AAAI/IAAI*, 1998.

[7] P. R. Cohen. The role of natural language in a multimodal interface. In *UIST*, 1992.

[8] P. T. Hoppman. *The Negotiation Process and the Resolution of International Conflicts*. University of South Carolina Press, May 1996.

[9] C. M. Jonker, V. Robu, and J. Treur. An agent architecture for multi-attribute negotiation using incomplete preference information. *Autonomous Agents and Multi-Agent Systems*, 15(2):221–252, 2007.

[10] P. Kenny, A. Hartholt, J. Gratch, W. Swartout, D. Traum, S. Marsella, and D. Piepol. Building interactive virtual humans for training environments. In *I/ITSEC*, 2007.

[11] R. Lin and S. Kraus. Can automated agents proficiently negotiate with humans? *CACM*, 53(1):78–88, January 2010.

[12] R. Lin, S. Kraus, J. Wilkenfeld, and J. Barry. Negotiating with bounded rational agents in environments with incomplete information using an automated agent. *Artificial Intelligence*, 172(6-7):823–851, 2008.

[13] M. J. Osborne and A. Rubinstein. *A Course In Game Theory*. MIT Press, Cambridge MA, 1994.

[14] Y. Oshrat, R. Lin, and S. Kraus. Facing the challenge of human-agent negotiations via effective general opponent modeling. In *AAMAS*, 2009.

[15] N. Peled, Y. Gal, and S. Kraus. A study of computational and human strategies in revelation games. In *AAMAS*, 2011.

[16] A. Rosenfeld and S. Kraus. Modeling agents based on aspiration adaptation theory. *Autonomous Agents and Multi-Agent Systems*, 24(2):221–254, 2012.

[17] R. Selten. Aspiration adaptation theory. *Journal of Mathematical Psychology*, 42:1910–214, 1998.

[18] B. Shneiderman and C. Plaisant. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. 2004.

[19] H. A. Simon. *Models of Man*. John Wiley & Sons, New York, 1957.

[20] I. Sofer, D. Sarne, and A. Hassidim. Negotiation in exploration-based environment. In *AAAI 2012*.

[21] D. Traum, S. C. Marsella, J. Gratch, J. Lee, and A. Hartholt. Multi-party, multi-issue, multi-strategy negotiation for multi-modal virtual agents. In *Intelligent Virtual Agents*.

[22] G. Tsoumakas, I. Katakis, and I. Vlahavas. Effective and Efficient Multilabel Classification in Domains with Large Number of Labels. In *MMD'08*, 2008.

[23] I. Zuckerman, A. Rosenfeld, S. Kraus, and E. Segal-Halevi. Towards automated negotiation agents that use chat interfaces. In *ANAC 2013*.