Document Agents with the Intelligent Negotiation Capability

Jerzy KACZOREK and Bogdan WISZNIEWSKI
Department of Intelligent Interactive Systems,
Faculty of Electronics, Telecommunications and Informatics, Gdansk University of Technology
80-233 Gdansk, Poland

ABSTRACT

The paper focus is on augmenting proactive document-agents with built-in intelligence to enable them to recognize execution contexts provided by devices visited during the business process, and to reach collaboration agreement despite of their conflicting requirements. We propose a solution based on intelligent bargaining based on neural networks to improve simple multi-issue negotiation between the document and the device, practically with no excessive cost to the agent with limited RAM and CPU resources.

Keywords: Proactive Document, eCollaboration, Mobile Computing.

1. INTRODUCTION

Individuals, who collaborate in a network organization, interact by exchanging electronic documents that constitute *units of information*, and at the same time *units of interaction*. This dichotomy has become apparent with the advent of active documents, often implemented as software agents. In particular, our mobile interactive document (MIND) [4] can migrate over the network and carry both, the content to be worked on and specification of its migration path with *activities* and *transitions*. Each activity represents a piece of work to be performed by the receiving user (knowledge worker) with the incoming document content, whereas transition indicates where the outgoing document, constituting a result of the activity, should migrate next.

Successful completion of each activity depends on the intentions of the document originator, operational characteristics of the device, as well as preferences of the knowledge worker responsible for the current processing step. A proactive MIND document can handle that in several ways: activity may be performed automatically by its *embedded* code – if allowed by the worker operating his/her device, may be performed manually by the worker using *local* services or tools installed on that device, or the device may call some *external* (third party) service requested by the document – if the Internet connection is available at the time of executing the activity. Owing to that, the agents – proactive MIND documents and execution devices alike – may exhibit specific beliefs and desires [3]; we represent them with *bargaining sets* and *policies*.

Attributes of execution contexts

When a proactive document and its execution device have different beliefs and desires with regard to what resources should be provided to perform the actual activity some consensus must be found. The conflict arises when the document wants to get from the device as much computational power and resources as possible, while the latter would like to complete the activity at the lowest possible cost, e.g., with regard to the battery power, network bandwidth consumption, and with the minimum risk of compromising its security, among others. In other words, the *execution context* must be found that satisfies both parties: the document and the device. When doing that the parties search the set of possible execution contexts for the one that may be agreed; we call that set the *bargaining set*. Elements of the bargaining set are *m*-vectors, called *offers*. Each offer $o = < v_1, \ldots, v_m >$, consists of values v_i, i=1,…,m, selected from the respective sets A_i, representing *attributes* of the execution context. In our model we distinguish five attributes: who shall be the actual *performer* of the activity (A_1), what is the current network *availability* (A_2), its *performance* characteristics (A_3), the current execution device *security* (A_4) and its possible *reliability* levels (A_5). They suffice to characterize a fairly wide range of classes of execution devices and proactive documents [5]. Moreover, *policy rules* define partial ordering of offers and specify which combinations of attribute values are allowed in the offers and which are not. We have formally defined these rules for specific classes of devices and documents.

There are five classes of devices in our model: *workstations* (x_1), *laptops* (x_2), *tablets* (x_3), *smartphones* (x_4) and *cellphones* (x_5). Each device class has a related subset of values that each respective attribute may assume. These subsets are denoted by unique labels that may appear in the respective levels of the option tree; combinations of these labels make the bargaining sets specific to the particular class of devices. Policy rules determine partial ordering of offers, and are of the form $X_i \rightarrow \{Y_{i+1}, \ldots, Z_{i+1}\}$, where $X_i \subset A_i$, $Y_{i+1}, \ldots, Z_{i+1} \subset A_{i+1}$ are subsets of the respective attribute value sets, and curly braces {} denote ordering of the respective attribute value labels. For example, Table 1 specifies the policy of laptop devices (x_2), implemented in our MIND system.

Table 1: Example of the laptop policy (x_2)

Attribute set		*Rules*
A_1	Performer	ε → {W[1-2],J[1-4],D[1-3]}
A_2	Availability	{W[1-2],J[1-4],D[1-3]}→{I[1-7],E[1-7]}
A_3	Performance	{I[1-7],E[1-7]} → {N[1-4], R[2,4], A[1-4],M[1-4]}
A_4	Security	{N[1-4], R[2,4], A[1-4],M[1-4]} → {C[1-4],T[1-4], K[1-4],P[1-4]}
A_5	Reliability	{C[1-4],T[1-4], K[1-4],P[1-4]} → {H[1-4],F[1-4], B[1-4],L[1-4]}

Regular expressions in the table denote a range of labels, e.g., D[1-3] denotes D1,..,D3, or a list of labels, e.g., R[2,4] denotes R2 and R4. Each single label represents a concrete subset of attribute values. In particular, for the 'performer' attribute, labels D[1-3] denote various subsets of values relevant to the autonomous execution of the activity by the document, without any help from the worker using the execution device, whereas J[1-4] refer to various execution contexts requiring interaction between the document and the worker. For the 'availability' attribute, labels I[1-4] and E[1-4] refer to various types of network connections (from inside of the host organization or external to it) that are available during execution. For the 'performance' attribute, label R2 denotes a concrete value indicating that a wireless network and excessive RAM may be provided, and so on. The rationale for partitioning the attribute value sets for the respective classes of devices has been explained in detail in [6], and is beyond the scope of this paper.

We have also defined four classes of documents, based on the following two characteristics. One is that the document content may be *protected* (P) or *open* (O) – depending on whether it does or does not need any secure connection to perform the activity, another is that it may impose a *heavy* (H) or *light* (L) load on the device – depending on whether it requires CPU power and the amount of RAM above some average level or not. For example, Table 2 specifies the policy of a proactive document which content is protected and light (d_{PL}).

Table 2: Example policy of d_{PL} proactive documents

Attribute set		*Rules*
A_1	Performer	ε → { J[1-4],D[1-3]}
A_2	Availability	{J[1-4],D [1-3]} → {I1,E1}
A_3	Performance	{I1,E1} → {N[1-4],R[1-4],A[1-4],M[1-4]}
A_4	Security	{N[1-4],R[1-4],A[1-4],M[1-4]} → {C[1-4],T[1-4], K[1-4]}
A_5	Reliability	{C[1-4],T[1-4], K[1-4]} → {H[3-4],B[3-4]}

The rest of the paper is structured as follows. Further in Section 1 the method of modeling preferences of negotiating parties is outlined. Section 2 introduces rules for the simple bargaining game and its basic implementation algorithms. In Section 3 the method for the intelligent selection of offers based on neural networks is proposed. Experiments evaluating the method are described in Section 4, while Section 5 provides a short survey of the related work in the area of intelligent negotiation. Section 6 concludes the paper.

Document-agent execution

The way the document activity is performed on the device where the mobile document is currently located, depends on the document functionality, operational characteristics of the device currently in use, and preferences of the worker (person) handling the device and expected to perform (or at least accept the outcome of) the current processing step. Combinations of these factors define specific execution contexts which may vary, since the same person may use different devices when performing activities of the same business process, e.g., using a workstation when in office, a smartphone during travel between office and home, and a laptop at home. User preferences for the same device may depend on its current location, often conflicting with document agents' policies.

Multi-item bargaining

In a loosely-coupled distributed system, like the multi-agent system described above, resolution of possible conflicts between document-agents and execution devices should rather not rely on any external Web service. This is because mobile devices may often be out of a trusted network (or any network at all) or the connection available to them may be too costly (if only a cellular network is possible), not mentioning the security and privacy problems, as activities of documents might be traced from outside of the business process. We propose to resolve conflicts between documents and devices by the means of negotiation [6]. The document-agent and its execution device exchange offers and counter-offers, and systematically search the space of possible contracts. However, this setting is non-cooperative: negotiating parties have mutually incomplete information on their preferences, as they keep them in secret and want to get individually as much wealth as possible. Such an alternating-offer protocol is modeled by us as a bilateral multi-stage simple bargaining game (SBG) [6]. Players exchange offers from the bargaining set they share.

Each player uses an *option tree* to model its individual preferences in the bargaining set, as shown in Figure 1. Attributes correspond to levels of the tree, with possible values of each attribute listed at its respective level. A path in the option tree defines a concrete offer combining these values, e.g., offers o_3=<D3,I1,N4,T4,H3> and o_4=<D3,E1,A4,C3,B4>. The bargaining set in Figure 1 is $C_B = \{o_1, o_2, o_3, o_4, o_5\}$. Each negotiating party has its own option tree, specifying the preferred order of all possible offers in C_B. In the example tree, offer o_3 is the most preferred, whereas o_4 is the least preferred one. Neither player reveals the ordering of offers in its tree to its opponent.

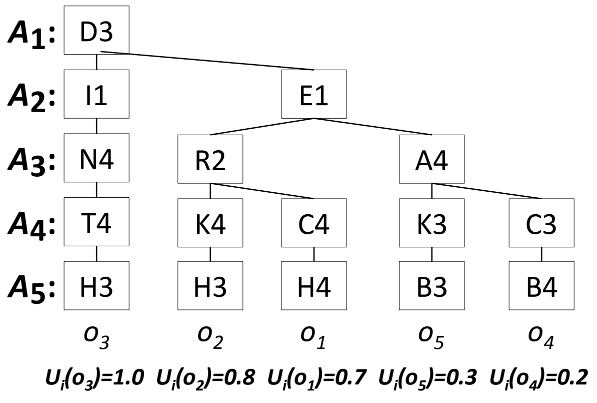

Figure 1: An example option tree and offers of player P_i

Each player P_i, i=1,2, determines the ordering of offers in its tree with its private utility function, $U_i: C_B \to (0,1]$. For example, in Figure 1 utility $U_i(o_3) = 1.0$, while $U_i(o_4) = 0.2$,

2. SIMPLE BARGAINING GAME

Players are rational, i.e. when making offers each one starts from the most valued offer in its respective option tree and concedes in successive rounds until offer $o_c \in C_B$ is agreed, such that:

$$o_c = \arg\max_{o \in C_B} U_1(o) U_2(o), \tag{1}$$

maximizes utilities of both players [6].

Formally, the objective of negotiating parties P_1 and P_2 is to find in the given bargaining set a solution to Eq. (1) under assumption that neither party knows its opponent's utility function. In order to do that, the execution device (player P_1) and the document-agent (player P_2) play the SBG game according to the following rules:

1) The game is started by P_1 (the execution device);
2) Both players, P_1 and P_2, keep in secret their private information, including their respective utility functions U_1 and U_2, and discount factors δ_1 and δ_2, but share knowledge on the bargaining set C_B;
3) Utility values of players' offers are discounted at each transition to the next stage;
4) Players exchange offers until the game is concluded, i.e., one of the players accepts an offer or quits the game;
5) The game may be concluded by any player P_1 or P_2, when:
    a. it repeats its opponent's offer received in any preceding stage, what implies accepting the offer as the agreed contract,

b. it repeats its own offer submitted in any preceding stage, what implies quitting the game, without agreeing any contract.

Implementation of the above rules is fairly simple: the game is sequential and the device alternate with the document by calling function *submitOffer*, as shown in the listing of Algorithm 1. The incoming MIND document-agent brings to the execution device its private implementation of the *submitOffer* function, embedded in its code. Another private implementation of this function is located on the device. Both implementations are called by the email client handling the document and executing function *playSBG* before allowing the document to perform its principal activity. Function *submitOffer* selects elements from the bargaining set $C_B = C_1 \cap C_2$, where C_1 and C_2 are sets of offers defined by the respective option trees of P_1 and P_2.

Algorithm 1: playSBG

```
1:  Data:
        $C_B$: bargainig set;
        $P_1.C_R$: offers received by the device;
        $P_1.C_S$: offers submitted by the device;
        $o_1$: current offer of the device;
        $P_2.C_R$: offers received by the document;
        $P_2.C_S$: offers submitted by the document;
        $o_2$: current offer of the device;
        k: current move number;
2:  Result: agreed contract and the final move number;
3:  k ← 0;
4:  $o_2$ ← null;
5:  while $o_1 \neq o_2$ do
6:      $o_1$ ← submitOffer($C_B$,$P_1.C_R$,$P_1.C_S$,$o_2$,k) ;
7:      if $o_1 = o_2$ then
8:          return [$o_1$,k];
9:      k ← k+1;
10:     $o_2$ ← submitOffer($C_B$,$P_2.C_R$,$P_2.C_S$,$o_1$,k) ;
11:     if $o_1 = o_2$ then
12:         return [$o_2$,k];
13:     k ← k+1;
```

Since any offer in C_B can be accepted by either player, the rule 5b listed above will never have to be applied and is skipped.

Simple selection of offers

When systematically searching the bargaining set C_B for the solution of Eq. (1), each player has to decide whether to accept one of the offers already received from its opponent, kept in its set C_R, or to select the offer not yet received from its opponent (nor submitted earlier to it, as it would imply quitting the game), kept in the set $C_N = C_B - (C_R \cup C_S)$. The decision is made in line 19 of Algorithm 2, implementing a simple selection of offers.

Algorithm 2: submitOffer (simple)

```
1:  Data:
        $C_B$: bargainig set;
        $C_R$: offers received by the player;
        $C_S$: offers submitted by the player;
        $C_N$: offers not submitted nor received yet;
        o: offer submitted by the player's opponent;
        δ: discount factor of the player;
2:  Result: if the opponent's offer is accepted it is returned as
        the agreed contract, otherwise a counter-offer not yet
        submitted is returned;
3:  if o = null then
4:      {start from the best offer}
5:      o ← max($C_B$);
6:      return o;
7:  else
8:      if $o \in C_S$ then
9:          {last offer accepted by the opponent}
10:         return o;
11:     $C_R \leftarrow C_R \cup \{o\}$;
12:     o' ← max($C_R$);
13:     $C_N \leftarrow C_B - (C_R \cup C_S)$;
14:     if $C_N = \varnothing$ then
15:         {accept the best received offer}
16:         return o';
17:     else
18:         o'' ← max($C_N$);
19:         if $U(o') \geq \delta \cdot U(o'')$ then
20:             {the incoming offer is better than the
                discounted future one}
21:             return o';
22:         else
23:             $C_S \leftarrow C_S \cup \{o''\}$;
24:             return o'';
```

Notice that the simple selection implemented by Algorithm 2 may often search the entire C_B, to the last possible element, before making a decision on what offer to accept. If the bargaining set is large, and the players option trees are sorted in the opposite order to each another, the game may have as many as $\lceil |C_B|/2 \rceil$ stages. The question is whether the number of stages can be reduced.

Prospect estimation

Note that in Algorithm 2, further exploration of the C_N subset of C_B, thus considering continuation of the game, may not be worthwhile if the utility function of the player making that decision strongly decreases in the C_N part of its option tree. In such a case it may be reasonable to accept the best offer from the ones already submitted by the opponent and kept in C_R. This idea is implemented by Algorithm 3.

Algorithm 3: submitOffer (estimated)

```
1:  Data:
        $C_B$: bargaining set;
        $C_R$: offers received by the player;
        $C_S$: offers submitted by the player;
        $C_N$: offers not submitted nor received yet;
        o: offer submitted by the opponent (initially empty);
        $\delta_1$: discount factor of the device player;
        $\delta_2$: discount factor of the document player;
        k: current step number;
        λ: cumulative discount factor;
2:  Result: if the opponent's offer is accepted it is returned as the
        agreed contract, otherwise a counter-offer not yet submitted is
        returned;
3:  if o = null then
4:      {start from the best offer}
5:      o ← max($C_B$);
6:      return o;
7:  else
8:      $C_N \leftarrow C_B - (C_R \cup C_S)$;
9:      o' ← max($C_N$);
10:     if o' = o then
11:         { accept the opponent's offer}
12:         return o;
13:     else
14:         {consider utility of all remaining offers }
15:         $C_N \leftarrow C_N - \{o, o'\}$;
16:         $C_R \leftarrow C_R \cup \{o\}$;
17:         o'' ← max($C_R$);
18:         $U_{sum} \leftarrow 0$;
19:         λ ← discount($|C_N|$,k+1, $\delta_1$,$\delta_2$);
20:         foreach $o_c \in C_N$ do
21:             $U_{sum} \leftarrow U_{sum} + \delta_1 \cdot (\lambda \cdot U(o_c) + (1 - \lambda) \cdot U(o))$;
22:         if $U_{sum} / |C_N| > U(o'')$ then
23:             {if there are still valuable offers submit the new
                one}
24:             return o';
```

```
25:         else
26:             {accept the best received offer};
27:             return o";
```

Function *discount*, called in line 19, evaluates the cumulative discount if the player decides to submit offers from C_N in future rounds, used next to calculate the average payoff of all such prospective offers. The loop in line 20, which calculates the formula in line 21, is the iterative version of the recursive formula implementing the regressive outguessing aimed at evaluating all possible outcomes of SBG when continued from the step following the one currently performed [7]. Algorithm 3 was used by us to simulate SBG when comparing it with the intelligent bargaining utilizing knowledge on the opponent's preferences, learned during previous encounters (see Algorithm 4). However, the cost of reducing the number of stages in Algorithm 3 is that the estimation of prospect offers by the player requires knowing the discount factor of its opponent. It may be either assumed to be the same, or the additional procedure to guess it during negotiation should be applied. The latter is beyond the scope of this paper, but is realistic to implement [7].

3. INTELLIGENT BARGAINING GAME

We propose a further improvement of our SBG game [6] by utilizing machine learning mechanism based on neural networks; the latter have been demonstrated in the literature to require quite moderate computational and storage resources [8]. In that research, however, it was assumed that no a-priori knowledge on the opponent existed, thus only on-line training during the actual encounter was considered. In consequence, the learning capability of negotiation agents concentrated on predicting better offers, i.e., offers that might faster lead to the contract based on the offers submitted by the opponent during the initial series of rounds of the actual encounter [8]. In contrast to that, we advocate the approach based on policies, which enable generation of training sets for the neural network before the encounter takes place; these sets could be much richer than those available to the negotiating document-agent during its actual encounter with the device. In consequence, our intelligent negotiation agent may be expected to perform far better than the one that learns to negotiate after the negotiation has started – especially when it lacks any significant computational resources.

Recognition of execution devices

Execution devices are recognized based on their bargaining sets, modeled as *occurrence vectors* – binary bit-words indicating the presence or absence of each possible attribute value in any offer recorded in the negotiation history during the training phase. This idea is illustrated in Figure 2.

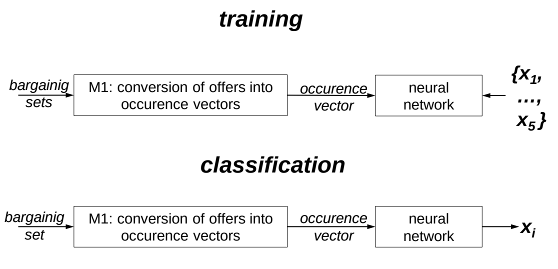

Figure 2: Device class recognition

Negotiation histories for the training set are generated based on the policies for each specific device class, and enable training the agent before starting its mission. Module M1 builds a bit-word of the form $\beta = \omega_1 \ldots \omega_5$, where each respective bit-field ω_i consists of flags $\omega_i[1] \ldots \omega_i[|A_i|]$, indicating whether the respective attribute value v_i^j occurred in offers submitted by the device. The network is trained with thus obtained occurrence vectors to recognize classes of devices, based on characteristic combinations of attribute values in bargaining sets of each possible class of devices.

Prediction of contracts

Once the execution device class is recognized the neural network may be set up to recognize the sequence leading to the contract. We base this analysis on recognizing binary relations between each pair of offers in the sequence. Each device class requires a separate neural network, of the same number of layers and neurons but with different weights. This is the key asset of our method, to which we will return in the next section. Figure 3 illustrates a general scheme for training the network to predict contracts based on input sequences.

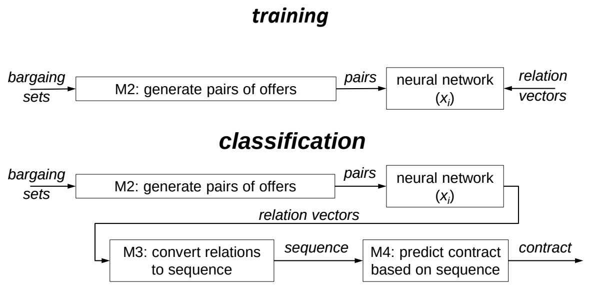

Figure 3: Contract prediction

Module M2 generates all possible pairs of offers used to train the network to recognize their precedence relations. Offers in each pair (o', o'') have the respective labels of their attribute values coded with the unique natural numbers, so each pair is represented as the 10-element vector of natural numbers. Two complete sets of such vectors are generated based on the set of sequences recorded when playing SBG over the particular bargaining set: one for vectors where o' precedes o'' $(o' < o'')$, and another where o' succeeds o'' $(o' > o'')$. When trained with pairs from the bargaining set and precedence relations from the negotiation history, the network is able to guess precedence relations based on the bargaining set under test. Module M3 uses the recognized relation vectors to reconstruct the predicted sequence of offers. The sequence can be used next by the player to choose offers in the more informed way than in the case of simple bargaining.

Informed selection

Module M4 in Figure 3 implements the following algorithm:

Algorithm 4: submitOffer (informed)

```
1:  Data:
        $C_B$: bargaining set;
        $C_R$: offers received by the player;
        $C_S$: offers submitted by the player;
        $C_N$: offers not submitted nor received yet;
        o: offer submitted by the opponent (initially empty);
        k: current step number;
        nnSeq: sequence predicted by the neural network;
2:  Result: if the opponent's offer is accepted it is returned as
        the agreed contract, otherwise a counter-offer predicted by
        the supporting neural network is returned.
3:  if o = null then
4:      {start from the best offer}
5:      o ← max($C_B$);
6:      return o;
```

```
7:  else
8:      $C_N \leftarrow C_B - (C_R \cup C_S)$;
9:      $C_N \leftarrow C_N - \{o\}$;
10:     $C_R \leftarrow C_R \cup \{o\}$;
11:     $o'' \leftarrow max(C_R)$;
12:     $U_{max} \leftarrow 0$;
13:     foreach $o_c \in C_N$ do
14:         {use the predicted sequence to find the offer leading
            to the maximum payoff}
15:         $o_{tmp} \leftarrow simulateSBG(o_c, k+1, nnResp)$;
16:         if $U(o_{tmp}) > U_{max}$ then
17:             $U_{max} \leftarrow U(o_{tmp})$;
18:             $o' \leftarrow o_c$;
19:     if $U_{max} > U(o'')$ then
20:         $C_N \leftarrow C_N - \{o'\}$;
21:         {submit the predicted counter-offer}
22:         return $o'$;
23:     else
24:         {accept the opponent's offer}
25:         return $o''$;
```

When comparing Algorithms 3 and 4, it may be seen that instead of estimating the average payoff of all prospective offers from the unexplored part C_N of the bargaining set, the prospective offer with the maximum payoff is searched (the loop in line 13), based on the sequence of the opponent's offers predicted by the network.

4. SIMULATION EXPERIMENTS

In the experiments we assumed that when playing SBG, as specified by Algorithm 1, the document-agent implemented its *submitOffer* function using Algorithm 4, while the device implemented Algorithm 3. In order to do that, the document-agent needed one neural network to recognize device classes and five neural networks (one for each device class) to recognize sequences. In practice, all these six classification tasks were accomplished effectively with just two networks, one for recognizing classes of devices, and another for recognizing sequences. The latter network required a dynamically changed set of weights to adjust it to recognize sequences of the specific device class, once it has been recognized by the document. In consequence, only the "knowledge" on how to make informed selection of offers had to be brought by the document to the device, along with the content to work on, whereas the network execution facility was provided by the device, running the specialized email client responsible for handling MIND documents [4]. The network required to classify devices (see Figure 2) consisted of the input layer of 20 neurons, one hidden layer of also 20 neurons, and the output layer of nine neurons, with the total of 2269 weights and biases to set it up. The network capable of classifying sequences (see Figure 3) consisted of the input layer of 10 neurons and the output layer with one neuron, requiring the total of 1766 weights and biases to set it up. Given the Java single-precision 32-bit IEEE 754 floating point representation the document had to carry $4 \cdot 1766 = 7064$ bytes per device, or for all five device classes about 35KB of data. If the MIND documents were equipped also with the "knowledge" required to recognize device classes (instead of email clients handling them), the total extra load was less than 45KB – a negligible amount when compared to the typical size of attachments allowed by the present-day email systems.

In the experiments, we have concentrated on three characteristics of the intelligent document agent, namely its *fitness level*, measuring the ratio of devices correctly recognized by it, *payoff* of the contract guessed by the trained network compared to the one negotiated by the agent without any machine learning, and the *cost* measured as the time of training the network and the length of negotiation measured as the round number where the contract was agreed. Networks of various sizes have been tested to find the minimal number of layers and neurons to get the 100% fitness level for recognizing device classes and sequences within each class device.

Figure 4 indicates that the 3-layer network mentioned before, when trained with at least 50 examples, was able to reach 100% (perfect) fitness level, i.e., the intelligent MIND document could properly recognize any of the five device classes during the encounter. Moreover, training the document to reach the perfect fitness level was not very demanding, as the network needed for that just a few seconds on a moderately equipped laptop. This observation is very important for any realistic implementation of the MIND document-agent system: training of documents may be performed directly on regular execution devices, mainly workstations or laptops, used by the collaborators during the implemented business process.

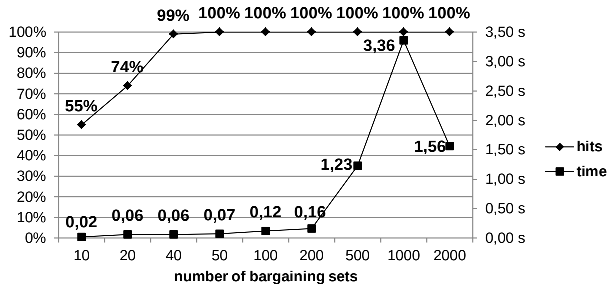

Figure 4: Training for device recognition

In the second phase of the experiment, when the contract prediction capability was tested, only sequences of offers from the perfectly recognized devices were used. Figure 5 illustrates results of recognizing sequences leading to the contract based on the initial subsequence of offers returned by the laptop device.

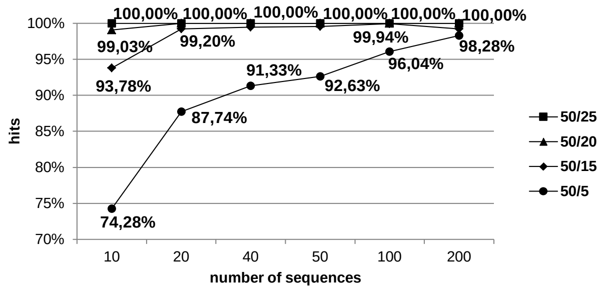

Figure 5: Recognition of offer sequences from the laptop device

In this particular experiment the bargaining set included 50 offers, and sequences of offers from that bargaining set of various lengths were used to generate the training examples; sequence used for training included respectively 5, 15, 20 and 25 offers out of the total 50. It may be seen that in order to train the network to properly recognize sequences leading to the contract, sequences including only a subset of offers from the bargaining set had to be used. While for the smallest fraction of offers used (only five out of 50) a relatively large set of 200 training sequences was needed – all other fractions required a significantly smaller set of about 20 training sequences. In the experiments with sequences of offers from the bargaining set of other device classes (not reported in the paper) we got similar results. They all indicate that the neural network may be effectively trained to recognize sequences.

The ultimate objective of our experiments was to evaluate how much the simple bargaining game may be improved by utilizing properly trained neural networks to recognize classes of devices and sequences of their offers. Results for all five classes of devices playing with the intelligent d_{PL} document (see Table 2) are presented in Figure 6.

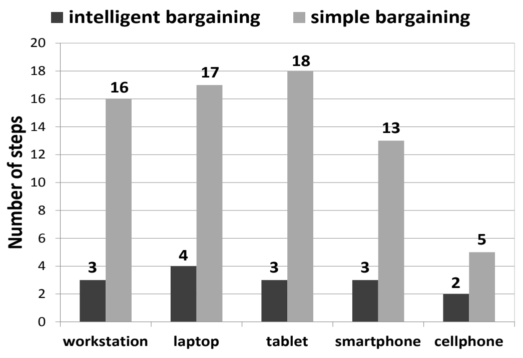

Figure 6: Intelligent vs. simple bargaining

We have first solved Eq. (1) by assuming both utility functions, U_1 and U_2, to be known. This solution is called in the literature the *fair* result. Next we performed a significant number of simulated encounters between documents and devices, with their options trees generated at random, but in accordance to the policy rules of each respective device and document class. In one part of the experiment parties were negotiating contracts using simple (estimated) bargaining, and in another, intelligent bargaining with documents utilizing the appropriately trained networks. In each simulated encounter option trees of the negotiating parties were sorted in the opposite order to each other to ensure the maximum possible number of steps. It may be seen that for each class of devices, documents capable of predicting sequences of offers returned by devices were able to get close to the fair result in a significantly smaller number of steps. The number of steps observed for various classes of devices is related to the number of combinations of attribute value options: cellphones had a significantly smaller number of value labels in their policy rules, therefore Algorithm 3 performed quite well compared to Algorithm 4.

5. RELATED WORK

The player attempting to model characteristics of its opponent in order to generate offers leading to the agreement faster may be represented as the optimization problem. For example, in [9] the best offer in each round is searched in the set of possible offers using the particle swarm optimization (PSO) method. Negotiation may also be modeled as a multi-state control process, in which the intelligent agent has to determine a sequence of optimal controls, i.e., to predict future offers [2]. Agents may also learn in the initial series of rounds of the actual encounter to predict offers that that may faster lead to the contract [1], [8]. However, negotiation-as-a-service (NaaS) proposed in [1] may work only when the document agent has (or is allowed to) access the network from its current execution device. Compared to the above, our approach also relies on neural networks, as they have been demonstrated to require moderate computational and storage resources, but prefers training of document agents before rather than during the actual encounter with the device.

6. CONCLUSION

This idea of a proactive document negotiating with its execution device is novel to the field of document engineering. We have shown that such a document can effectively interact with a non-cooperative device to agree on the execution context that may satisfy both parties when using the economic agent model augmented with a relatively simple machine learning mechanism. Our experiments show that these mechanisms may be effectively implemented using even less powerful mobile devices. In our view there will be a growing demand for investigating mechanisms proposed in this paper, in particular to develop methods for recognizing device classes not yet known to agents – as the advances in the mobile device technology will push forward the evolution of new computational 'life forms' in the Web.

Acknowledgment

This work was supported in part by the National Science Center, Poland, under grant DEC1-2011/01/B/ST6/06500.

7. REFERENCES

[1] M. I. Bala, S. Vij and D. Mukhopadhyay, "Intelligent Agent for Prediction in E- Negotiation: An Approach", in **Proc. Int. Conf. on Cloud Ubiquitous Computing Emerging Technologies CUBE'13**, IEEE, pp. 183–187.

[2] J. Brzostowski and R. Kowalczyk, "Predicting Partner's Behaviour in Agent Negotiation", in **Proc. 5th Int. Joint Conf. on Autonomous Agents and Multi-agent Systems AAMAS'06**, ACM, 2006, pp. 355–361

[3] M. Georgeff, B. Pell, M. Pollack, M. Tambe and M. Wooldridge, "The Belief-Desire-Intention Model of Agency", in **Proc. 5th Int. Workshop ATAL'98**, LNCS, Vol. 1555, J. Müller, A. Rao and M. Singh, Eds., Springer, pp. 1-10.

[4] M. Godlewska and B. Wiszniewski, "Smart Email: Almost an Agent Platform", **Innovations and Advances in Computing, Informatics, Systems Sciences, Networking and Engineering,** LNEE, Vol. 313, T. Sobh and K. Elleithy, Eds., Springer, 2015, pp. 581–589.

[5] J. Kaczorek and B. Wiszniewski, "Augmenting Digital Documents with Negotiation Capability", in **Proc. ACM Symp. on Document Engineering DocEng'13**, ACM, 2013, pp.95-98.

[6] J. Kaczorek and B. Wiszniewski, "Bilateral Multi-Issue Negotiation Between Active Documents and Execution Devices", in **Proc. 9th International Conference on Digital Society ICDS'15**, IARIA, 2015, in press.

[7] J. Kaczorek, "Automated Negotiations over Collaboration Protocol Agreements", **Ph.D. Thesis**, Gdansk University of Technology, Gdansk, Poland, 2015.

[8] I.V. Papaioannou, I. Roussaki, and M.E. Anagnostou, "Using Neural Networks to minimize the Duration of Automated Negotiation Threads for Hybrid Opponents", **Journal of Circuits, Systems, and Computers**, Vol. 19, No. 1, 2010, pp. 59–74.

[9] Z. Wang and L. Wang, "Adaptive Negotiation Agent for Facilitating Bi-Directional Energy Trading Between Smart Building and Utility Grid", **IEEE Trans. Smart Grid**, Vol. 4, No. 2, 2013, pp. 702– 710.