Time Series Predictive Models for Opponent Behavior Modeling in Bilateral Negotiations

Gevher Yesevi$^{1[0000-0001-6992-519X]}$, M. Onur Keskin$^{1[0000-0002-8025-4304]}$, Anıl Doğru$^{1[0000-0002-9951-8912]}$, and Reyhan Aydoğan$^{1,2[0000-0002-5260-9999]}$

[1] Computer Science, Özyeğin University, Istanbul, Turkey
[2] Interactive Intelligence, TU Delft, Delft, Netherlands
{gevher.yesevi,onur.keskin,anil.dogru}@ozu.edu.tr,reyhan.aydogan@ozyegin.edu.tr

Abstract. In agent-based negotiations, it is crucial to understand the opponent's behavior and predict its bidding pattern to act strategically. Foreseeing the utility of the opponent's coming offer provides valuable insight to the agent so that it can decide its next move wisely. Accordingly, this paper addresses predicting the opponent's coming offers by employing two deep learning-based approaches: Long Short-Term Memory Networks and Transformers. The learning process has three different targets: estimating the agent's utility of the opponent's coming offer, estimating the agent's utility of that without using opponent-related variables, and estimating the opponent's utility of that by using opponent-related variables. This work reports the performances of these models that are evaluated in various negotiation scenarios. Our evaluation showed promising results regarding the prediction performance of the proposed methods.

Keywords: Automated Negotiation · Multi-agent Systems · Time-series Prediction · Utility Prediction

1 Introduction

Autonomous negotiating agents decide on their behaviors by considering various factors such as time pressure, the competitiveness of the underlying negotiation domain, the opponent's collaborative level, and so on [8, 14]. Sophisticated strategies aim to detect the opponent's behaviors and make reciprocating moves. Thus, understanding the fundamental causes of behavior is one of the essential research questions in agent-based negotiation systems. Once the underlying causes of their opponent's behaviors are observed, negotiating agents can act strategically to get better negotiation outcomes sooner. Consequently, agents can target to maximize their utility or increase social welfare depending on the context.

In autonomous negotiations, decisions on when to accept the opponent's counter-offer or what offer to make (i.e., acceptance and bidding) are made according to the employed strategies. These strategies are often defined based on the target utility calculations [2, 7, 11]. Therefore, predicting opponents' target utility for future rounds would be an excellent approach to grasp their behaviors. In the literature, some attempts aim to detect the opponent's moves/target utilities and act accordingly [4, 7]. For instance, Williams *et al.* present a Gaussian process to foresee the concession rate of the opponent [19]. Furthermore, researchers apply reinforcement learning to determine their target utility based on the exchanged offers and remaining negotiation time during the negotiation [1, 16]. Those studies implicitly take the opponent's behavior into account. Moreover, Chen *et al.* suggest applying transfer learning in negotiation to benefit from previous negotiation experiences [5].

With the growing need for decision-making processes in complex domains, there is an increasing interest in understanding the behavior of other agents such as logistics and transportation [6, 8, 16]. Some of them focus on learning the patterns and trends in an agent's behavior. For this type of prediction, time series analysis is a promising approach to build such predictors. In automated negotiation, Li *et al.* recently adopt such a time series analysis to recognize the opponent's strategy during the negotiation [12]. In their work, the agent aims to classify what strategy its opponent employs by analyzing the history of the offers made by the opponent. Inspired by that study and considering the importance of the target utility estimation in negotiation, we propose adopting time series predictors to guess the utility of the opponent's coming offers so that the agent can strategically make its decisions.

Accordingly, the goal of this study can be summarized twofold: (i) introducing two deep learning-based models, namely Long short-term memory (LSTM) [10] and Transformer models [18], to guess the utility of the opponent's following offers, and (ii) studying the effect of opponent's strategy and the size of the negotiation domain on the performance of the implemented predictors. This work could lead to a promising research direction toward recognizing the opponent's strategy. We believe that predicting the utility of the opponent's following offers (i.e., next-step utility prediction) helps the agent developers design resilient and robust negotiation strategies.

The following sections in this paper are as follows: Section 2 and 3 provide the information on the reviewed literature and the necessary background about automated negotiations. The proposed approach and the details of the prediction models are explained in Section 4 while its evaluation is elaborately reported in Section 5. Finally, Section 6 concludes this paper with future work.

2 Related Work

Various negotiation strategies have been proposed in the literature. Existing strategies usually calculate a target utility at each round and generate a bid with that utility. Time-based strategies such as Conceder and Boulware agents determine the target utility through a function of remaining negotiation time [7]. Opponent agents can straightforwardly exploit such strategies since they do not consider opponent's behaviors. Thus, strategies like Tit-for-Tat consider opponent's consecutive offers to determine their coming offers by mimicking their opponent to some extent. For a more robust strategy, Faratin *et al.* suggest adopting a hybrid strategy, which combines both time and behavior-dependent tactics similar to the strategy presented in [11]. Moreover, another strategy determines several negotiation states and proposes adopting a specific bidding tactic for each state while considering the opponent's consecutive offers [15].

The aforementioned strategies consider their opponent's behavior based on their previous offers during the negotiation. Besides, guessing an opponent's future moves may enable a negotiating agent to act strategically. Regarding predicting the opponent's future behavior, there is some literature work. For instance, Williams *et al.* present a Gaussian process to predict the utility of the opponent's next offer [19] by assuming that the opponent concedes over time. Consequently, the agent can estimate its opponent's future concession.

Another direction is to build a prediction model for the opponent's strategy. Accordingly, Li *et al.* introduce the idea of applying a time series prediction model to classify an opponent's strategy among a predefined set of strategies [12]. It is claimed that the proposed idea can be adopted independently from the domain. For this purpose, the authors use the agents of the negotiation platform called Genius [13]. Notably, they use the LSTM model for recognizing the opponent's strategy. In some cases, such classifiers may not perform well, especially when facing an opponent employing a sophisticated unknown strategy. Therefore, unlike that study, we propose adopting time series models such as LSTM [10] and Transformers [18] to predict the utility of the opponent's following offers. To our knowledge, Transformers have not been used yet in this context.

3 Automated Negotiation

In automated negotiation, agents negotiate over a finite set of n issues $\mathcal{I} = \{1, 2, \ldots, n\}$. Each issue $i \in \mathcal{I}$ has a range $\mathcal{D}_i$ of possible instantiations. An outcome, $o \in \Omega$, is a complete assignment to the set of issues where Ω is the Cartesian Product of the ranges of instantiations per issue. Formally, the set of all possible outcomes is defined as $\Omega = \mathcal{D}_1$ X $\mathcal{D}_2$ X... X $\mathcal{D}_n$. The assessment of each offer/outcome is done using a utility function mapping each negotiation outcome to a real number [0, 1], the desirability of that outcome. The utility function is a mathematical representation of the agent's preferences. As usual, additive utility functions are used for this purpose [8]. Equation 1 shows the function where w_i represents the importance of the negotiation issue I_i (i.e., issue weight), o_i represents the value for issue i in offer o, and V_i is the valuation function for issue i, which returns the desirability of the issue value. Without losing generality, it is assumed that $\sum_{i \in n} w_i = 1$ and the domain of V_i is (0, 1) for any i. An issue value is preferred when its valuation value V_i is higher. A negotiating agent utilizes its utility function to determine what to offer and when to accept.

$$\mathcal{U}(o) = \sum_{i=1}^{n} w_i \times V_i(o_i) \tag{1}$$

Stacked Alternating Offers Protocol (SAOP) governs the interaction among agents (i.e., what actions can be taken under which condition and when to stop the negotiation) [3]. As there is a deadline to reach an agreement, the interaction ends when agents find a consensus or reach the deadline. The interaction starts with an offer made by one of the agents. In each turn, the agent receiving an offer can (i) *accept* the current offer, (ii) make a *counteroffer*, or (iii) *end* the negotiation without an agreement. The interaction continues in a turn-taking fashion and ends until reaching an agreement or deadline. Agents generally do not know their opponent's preferences or negotiation strategies. However, they can try to learn those preferences/strategies over time by analyzing offer exchanges as we aim in this study.

To analyze the behavioral changes of negotiating agents during the negotiation, Hindriks *et al.* define six negotiation moves. A move is determined based on the utility difference of the negotiator's subsequent offers for both sides. These moves are defined as follows: *fortunate*, *nice*, *concession*, *selfish*, *unfortunate*, and *silent* [9]. Table 1 demonstrates the calculation of move types of a player where ΔU_A and ΔU_{Op} represent the utility difference for the negotiator itself and that for the opponent, respectively.

Table 1: Move Specification of a Negotiator [9]

	Self Difference	**Opponent Difference**
Silent	$\Delta U_A = 0$	$\Delta U_{Op} = 0$
Nice	$\Delta U_A = 0$	$\Delta U_{Op} > 0$
Concession	$\Delta U_A < 0$	$\Delta U_{Op} > 0$
Unfortunate	$\Delta U_A < 0$	$\Delta U_{Op} < 0$
Fortunate	$\Delta U_A > 0$	$\Delta U_{Op} > 0$
Selfish	$\Delta U_A > 0$	$\Delta U_{Op} < 0$

4 Proposed Prediction Approach

The main focus of this study is to build a prediction module that generates next-step utility value predictions during bilateral negotiations to improve the decision-making process of the agents. Consequently, agents may avoid making an offer whose utility is lower than the utility of its opponent's next offer (i.e., leaving money on the table). In particular, when the agent negotiates with the same opponent several times, it can foresee its behavior and act wisely to find a consensus sooner. Those predictions may play an essential role in capturing the trends of opponent behavior and help to recognize the opponent's strategies.

Accordingly, this study aims to predict the utility of the opponent's following offers to design sophisticated and robust negotiation strategies by adopting the three objectives listed below:

- ***Objective 1:*** Predicting the utility of its opponent's coming offer for itself, $U_A(o_{t+1})$, based on remaining time and bid exchanges so far.
- ***Objective 2:*** Improving the performance of the prediction gained in Objective 1 by considering additional features such as estimated opponent utility, Nash Distance, and opponent's moves.
- ***Objective 3:*** Predicting its opponent's estimated utility of its opponent's next offer, $\widehat{U}_{Op}(o_{t+1})$, based on remaining time, bid exchanges so far and additional features used in *Objective 2*.

Accordingly, Figure 1 illustrates the inputs used within our next-step utility prediction for achieving each objective mentioned above. For any kind of analysis, a negotiating agent can keep track of all offers made during the negotiation and calculate their utilities for itself by using its utility function ($< U_A(o_0), ..., U_A(o_t) >$) at time t. To achieve the first objective, we suggest adopting a time series predictor that can be fed by the remaining time (t_{remain}) and the agent's utility of a chunk (we will refer to it as 'window' for the rest of the paper) of previous consecutive offer exchanges ($< U_A(o_{t-k}), ..., U_A(o_t) >$). The designer can choose the size of the window k to be used for each training set.[3].

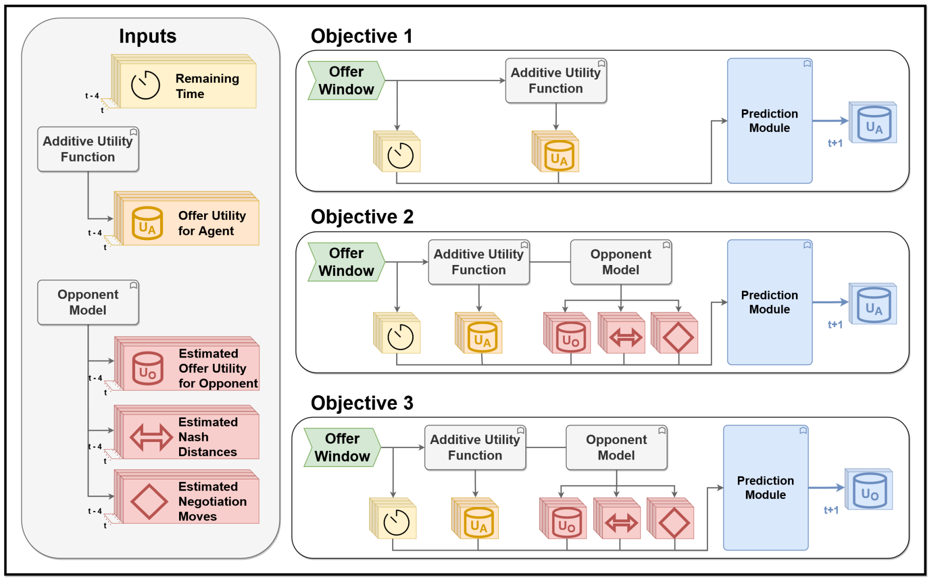


Fig. 1: Inputs and outputs for next-step utility prediction

For *Objective 2*, we suggest including the opponent's utilities of the offer exchanges, negotiation move analysis, and the Nash distances. Although the agent

[3] In our work, we consider five previous consecutive offer exchanges

has no access to the opponent's preferences, it could employ an opponent modeling for learning the opponent's preferences over time in terms of utilities. Any opponent modeling study from the literature [4] could be utilized for this purpose. For our work, a frequency-based opponent modeling mechanism is chosen [17]. Consequently, the agent can exploit additional estimated information such as the opponent's utility of a given offer, negotiation move types, and the Nash distances of the given offers. It is worth noting that the accuracy of the selected opponent model may affect the predictor's performance for the next-step utility value. To sum up, the following inputs feed the second predictor:

- Agent's utilities for the offer window, $< U_A(o_{t-k}), ..., U_A(o_t) >$,
- Estimated opponent utilities for the offer window, $< \widehat{U}_{Op}(o_{t-k}), ..., \widehat{U}_{Op}(o_t) >$,
- Nash distances for the offer window, $< \Delta N_(o_{t-k}), \Delta N_(o_t) >$,
- Estimated moves for the offer window, $< m(o_{t-k}, o_{t-k+1}), ..., m(o_{t-1}, o_t) >$,
- Remaining negotiation time, t_{remain}.

To accomplish the third object, we utilize the same inputs used for *Objective 2*, but the output of the model is the estimated opponent's utility of the opponent's coming offer, $\widehat{U}_{Op}(o_{t+1})$. All offer-related inputs can be kept as a time series during negotiation. This allows us to use supervised learning models to learn the sequential behaviors of negotiating agents and predict the next-step utility values accordingly. Due to the sequence-based characteristics of negotiation sessions, selecting a suitable multi-variate time series prediction method is essential. Considering these requirements, we decided on two deep learning architectures with sequence processing behaviors: LSTM and Transformer models.

4.1 Time Series Predictive Model Architectures

Next-step utility value prediction is essential for a negotiating agent for decision-making. With time series prediction methods, it becomes possible to observe the historical data patterns and foresee the behaviors in the following steps. Selected models, namely LSTM and Transformer models, have been used widely for this purpose. LSTM model is an extension of recurrent neural networks (RNN). This model has been selected due to the high predictive performance in similar competitive markets. After producing successful results on different sequence-based learning tasks like Natural Language Processing (NLP) and Computer Vision, LSTM models have become prevalent in regression tasks involving sequential data. When the Attention layers are introduced to replace RNNs, many use cases that prefer LSTM models have considered Transformer models in different learning tasks like NLP and Computer Vision. Therefore, we included Transformer models in our prediction module to test this assumption for next-step utility predictions.

Sequential information is fed to the aforementioned models with a sliding window approach. This approach first requires a number of time steps to decide the window size of data it will carry. Then, each iteration utilizes this window in the computations by sliding the window by one step. The inputs of each iteration not only depend on the current step but are also fed by the previous inputs received by the window, as can be seen in Figure 2.

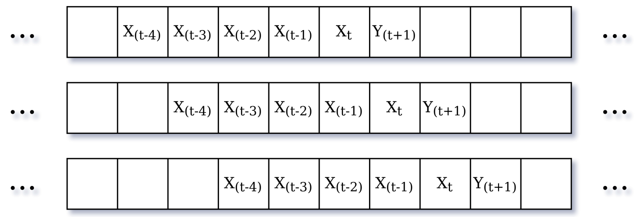


Fig. 2: Sliding window approach of model input and output

Long Short-Term Memory (LSTM) Model Architecture: Recurrent Neural Network (RNN) deep learning architectures are designed for processing sequential inputs. It is a widely used neural network since its ability to model the time dependency in its architecture. There are many variations of RNN to improve its efficiency. One of these variations is LSTM [10]. Mainly, deep neural networks suffer from the vanishing gradient problem, which occurs when the calculated gradient gets close to zero value during back-propagation operation. This situation causes the information to be lost during the model training. To resolve this problem, LSTM architecture introduces memory gates to control the input flow. This way, the information of long sequences can be preserved during the training and processed in a unidirectional way.

The LSTM architecture we employed contains one LSTM layer with 45 units and a fully connected layer to produce the predictions, as shown in Figure 3. The model takes last five inputs; $X_{(t-4)}$, $X_{(t-3)}$, $X_{(t-2)}$, $X_{(t-1)}$, $X_{(t)}$ as a sequence and predicts next value $Y_{(t+1)}$. The data is processed in the model with a batch size of 32. 'Mean Squared Error' is used as the loss function, whereas 'RMSProp' is selected as the optimizer since it is widely preferred for regression problems. The model is trained for 50 epochs with a 0.001 learning rate.

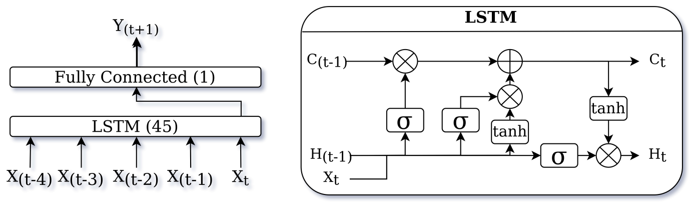


Fig. 3: The left-hand side demonstrates the proposed LSTM model, while the right-hand side demonstrates the LSTM architecture

Transformer Architecture: Transformer models stand out with the self-attention mechanism they employ to draw dependencies between inputs and outputs instead of focusing on recurrence in neural networks (i.e., allowing data to be processed regardless of a direction) [18]. This attention mechanism takes the positions of input and output sequences, connects them to each other, and traverses among them to decide on what to pay attention to the most. While connecting the sequences, the architecture limits the process with a constant number of sequentially executed operations, whereas recurrent layers require $O(n)$ sequential operations. The advantage of having lower complexity pays off when the tasks involve very long sequences.

Recent studies show that Transformer-based architectures can be used to predict time series [21]. Wu *et al.* use an encoder-decoder based approach for time series prediction in their study [20]. Their model takes the time series inputs with a window size of five, and the output as a sequence with the shifted input indices. After experimenting with various Transformer-based architectures for our task, we propose an encoder-decoder model with shifted inputs. Figure 4 shows the Transformer block and input-output design of this architecture that is used in our study.

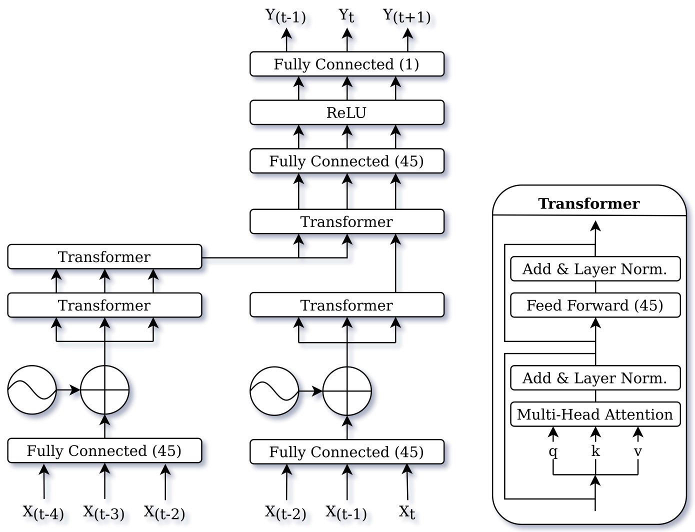


Fig. 4: Proposed Transformer based encoder-decoder model

Encoder: The model's encoder is shown on the left-hand side of Figure 4. It takes the first 3 inputs of sequence ($X_{(t-4)}$, $X_{(t-3)}$, $X_{(t-2)}$). In the study introducing attention layers, Vaswani et al. use input embedding for the 'token2vector' operation [18]. However, we prefer to use a fully connected layer instead of input embedding to adapt the original structure of Transformer models to time series prediction tasks, similar to the work of Wu et al. [20]. Note that the other components of the original Transformer structure remained the same. The encoder of the model has two Transformer blocks.

Decoder: The model's decoder is shown on the right-hand side of Figure 4. It takes the last 3 inputs of sequence ($X_{(t-2)}$, $X_{(t-1)}$, X_t), and predicts the corresponding shifted outputs ($Y_{(t-1)}$, Y_t, $Y_{(t+1)}$). Similar to the encoder, the decoder uses a fully connected layer instead of input embedding and contains two Transformer blocks. The inputs used in both Transformer blocks are the same, except the inputs obtained from the encoder and fed into the decoder. The decoder part of the architecture contains two fully connected layers right after the second Transformer block.

Similar to the LSTM architecture, we build the Transformer model with a window size of five and batch size of 32, then train it for 50 epochs. 'Mean Squared Error' is selected as the loss function while 'RMSProp' optimizer is used with a 0.001 learning rate.

5 Evaluation

The objectives of this study have been evaluated elaborately in varying negotiation settings (i.e., different negotiation scenarios and opponent strategies). We pre-trained the selected model architectures with historical negotiation sessions to feed the prediction models with the defined input and target variables. This section reports the actions taken for model training and evaluations of three different objectives. Accordingly, Section 5.1 describes our experimental setting where the results are analyzed and discussed in Section 5.2.

5.1 Experimental Setup

Defined model architectures must be fed with historical negotiation sessions to capture the opponent agent's behaviors during the negotiation. For this reason, multiple negotiations with different settings have been conducted using the GeniusWeb negotiation framework [13] whose collected sessions are used for model training purposes. Note that GeniusWeb is the Web version of Genius, which allows agent developers to design their agents in Python. It supports machine learning libraries that are available in Python for time series predictions. Unfortunately, we cannot directly use agents available in Genius on this platform. There are a few agent strategies available at this moment. For the baseline agent that performs the learning process, we implemented an additional strategy – a hybrid bidding strategy [11] with a frequency-based opponent model [17]. The negotiations have been conducted between this baseline agent and five available agents, which employ Boulware, Conceder, behavior-based, hybrid, and another hybrid strategy with opponent modeling. The brief information about those agents is given below:

– ***Time-based strategies***: These strategies calculate a target utility considering the remaining time and generate an offer with that utility [7]. The agents adopt a function of normalized negotiation time to determine the utility of their coming offer. They tend to concede on their target utility over time. The main difference between the Conceder and Boulware agents is their concession curve. While the Conceder agent concedes quickly over time, the Boulware agent tends to concede only when the deadline is approaching.

– ***Behaviour-based strategy***: The opponents that employ this strategy are expected to be opponent-aware and behave accordingly. The agents that adopt a behavior-based strategy mimic the behavior of the opponent. To do so, the agent calculates the utility difference between the opponent's two consecutive offers and applies the concession amount proportionally [7].
– ***Hybrid strategy***: The agents employing the hybrid strategy make their decisions by considering both remaining time and the behaviors of the agent they negotiate with. The target utility is a linear combination of the target utilities calculated by time-based and behavior-based strategies as shown in Equation 2 [11]. The principal intuition is that when time is not crucial (e.g., at the beginning of the negotiation), the agent pays more attention to its opponent's behavior while deciding on its next bid's target utility. As the deadline approaches, it tends to find an agreement urgently; therefore, it cares more about the remaining time. We use two versions of this agent with and without opponent modeling in our work.

$$TU_{Hybrid} = (t^2) \times TU_{Times} + (1 - t^2) \times TU_{Behavior} \tag{2}$$

To assess the performance of the proposed approaches, we run negotiation tournaments with varying negotiation scenarios (i.e., domain with two preference profiles for bilateral negotiation). For this purpose, we selected nine negotiation domains available in the GeniusWeb negotiation platform. Table 2 shows the selected domain information of negotiation sessions that are used for training and testing purposes. In order to evaluate the effect of domain sizes, two large, two medium, and two small-sized domains have been selected for training sessions, whereas one domain for each size group has been selected for testing purposes. The baseline agent negotiates with all opponents for each domain and preference profile. The negotiations are repeated ten times for each negotiation configuration with a session deadline of three minutes. After completing the negotiation sessions with five opponents, two preference profiles, and six domains ten times, we obtained 600 different negotiation sessions for training purposes. Other than that, 60 negotiation sessions are collected for testing purposes after negotiating with five opponents, two profiles, and three different domains twice.

In addition to feeding all collected data into the models, the filtered sessions of the small, medium, and large domains have been trained separately. Therefore, collected negotiation sessions have been grouped by domain sizes before training models. When the training data becomes ready, training sessions for each objective have been conducted with prepared data sets.

Table 2: Domain details for both training and test data

	DomainID	Category	Issue & Value List	Bid Space	Opposition
Train	1	Small	[2, 3, 2, 2, 2, 3, 3]	432	0.189
	2	Small	[8, 2, 2, 8, 2]	512	0.096
	3	Medium	[5, 11, 8, 9]	3,960	0.260
	4	Medium	[17, 4, 5, 3, 2]	2,040	0.056
	5	Large	[26, 20, 8, 4]	16,640	0.281
	6	Large	[26, 22, 10, 8]	45,760	0.255
Test	7	Small	[2, 3, 2, 2, 11, 2]	528	0.080
	8	Medium	[4, 3, 2, 2, 5, 5, 3]	3,600	0.191
	9	Large	[26, 8, 26, 2]	10,816	0.280

5.2 Results

After training the neural networks (LSTM and Transformer) with 600 negotiation sessions with given configurations, we generate the predictions of the opponent's next-step utility values on the test data for all rounds after the fifth round. Predicted values are compared with the actual values by using two evaluation metrics; the root-mean-square error (RMSE) and the mean absolute percentage error (MAPE). Recall that we make predictions for 60 negotiation sessions in total. For the final results, we take the average of calculated RMSE and MAPE metrics. Additionally, we analyze the effect of the domain size per each objective. Remind that *Objective* 1 and *Objective* 2 have a shared target variable which is the agent's utility value of the opponent's next offer, whereas *Objective* 3 makes use of the same opponent-related input variables as *Objective* 2 during training. However, its target is to guess the opponent's utility. Therefore, the evaluations are formed with the consideration of these differences.

We first evaluate the performance of the predictions regarding to *Objective* 1 and *Objective* 2. Figure 5 shows the MAPE scores of each model on each domain size for both objectives. The blue bars show their prediction performance on all test scenarios. It can be said that the *Objective* 1 achieved prediction with 17% and 18% MAPE by Transformer and LSTM, respectively. Here, the performance of the Transformer is slightly better than the LSTM. Moreover, it is obvious that the prediction error decreases when the models consider the opponent-related features (i.e., estimated opponent's utility of its previous offers, their Nash distances, and estimated opponent moves described in Table 1). Therefore, we can say that *Objective* 2 is achieved.

To investigate the effect of the domain size on Transformer and LSTM models, we train separate predictors for different sized scenarios (i.e., small, medium, and large), which are shown by orange, grey, and yellow bars, respectively. We observed that predicting agents' utilities on large domains have the highest error rates regardless of the chosen model. This situation may mainly stem from the high complexity of possible offer distributions in large negotiation domains. Compared to other domain sizes, a regular negotiation session in large domains covers a small portion of the bid space, which might decrease the model's gen-

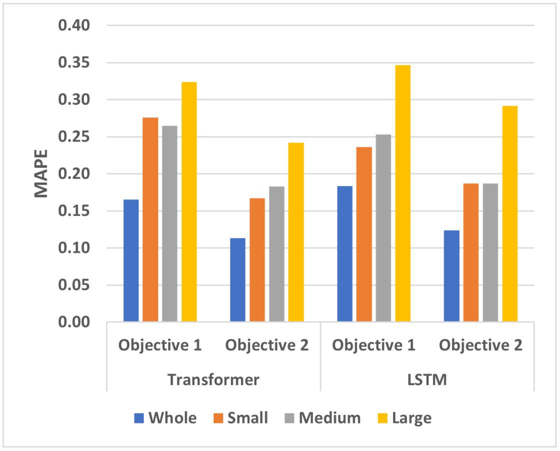
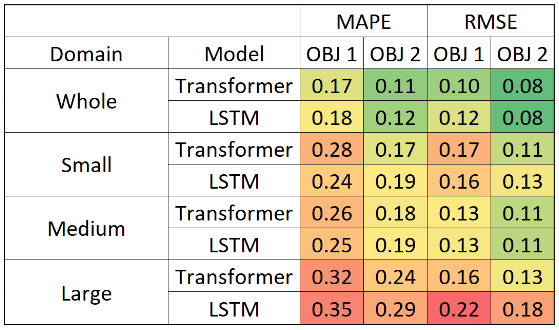


Fig. 5: Model comparison among different domain sizes for Objectives 1 & 2

eralization ability. On the other hand, the models trained with whole data show the best performances among others since they are introduced to more diverse examples in the training data. For more detailed analysis, Figure 6 elaborately summarizes both MAPE and RMSE scores per each setting for *Objective* 1 and *Objective* 2. According to the RMSE and MAPE scores of *Objective* 1 and *Objective* 2, it can be observed that both metric scores of *Objective* 2 are lower than of *Objective* 1. This observation shows that both models are improved as they are enriched with additional opponent-related variables, as expected.

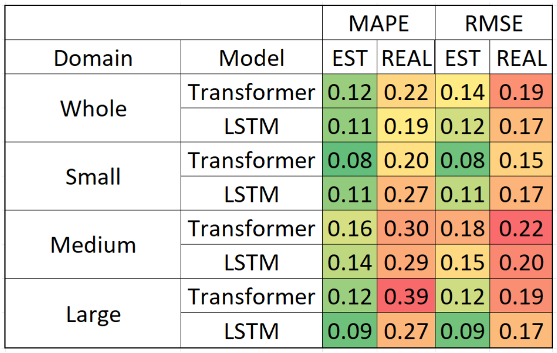


		MAPE		RMSE	
Domain	Model	OBJ 1	OBJ 2	OBJ 1	OBJ 2
Whole	Transformer	0.17	0.11	0.10	0.08
	LSTM	0.18	0.12	0.12	0.08
Small	Transformer	0.28	0.17	0.17	0.11
	LSTM	0.24	0.19	0.16	0.13
Medium	Transformer	0.26	0.18	0.13	0.11
	LSTM	0.25	0.19	0.13	0.11
Large	Transformer	0.32	0.24	0.16	0.13
	LSTM	0.35	0.29	0.22	0.18

Fig. 6: Model comparison among different domain sizes for Objectives 1 & 2

The evaluation of *Objectives* 1 and 2 is important to understand the opponent's attitude from the agent's utility change perspective. Besides, it is crucial to predict the opponent's utilities of their coming offers since most of the opponents generate their offers by considering their utility. Accordingly, *Objective* 3 aims to learn opponent's utility. Note that the agent does not know the opponent's utility function. Therefore, we first assess the predictor's performance by comparing the predictions with the estimated utilities by relying on the current opponent model. Then, we compare these predictions with actual utilities calculated by the opponent's utility function in the system. It is worth noting that these actual utilities are not accessible by the agent during the training process and the adopted opponent modeling approach may affect the performance of this comparison between predicted opponent utilities and the actual opponent utilities since the models are trained with the estimations. Implicitly, this comparison can give some insights about the performance of the opponent modeling technique employed by the agent.

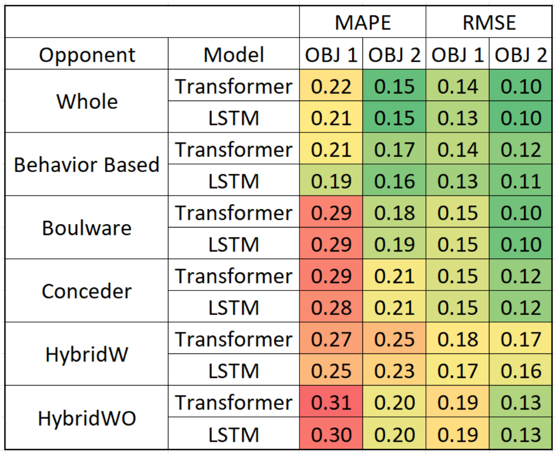
Figure 7 compares the predictions with estimated and actual opponent utilities. For both models, the predictions have higher error rates when compared to the actual opponent utilities as expected. However, the performance of predicting the estimated values shows the positive impact of utilizing opponent-related inputs to predict opponent utility. Inherently, this shows that improving the employed opponent model's accuracy might help obtaining lower error rates while predicting actual opponent utilities, which can lead to an interesting research direction. In general, the performance of the LSTM models seems to be slightly better than Transformer in all domain categories except small domains.



		MAPE		RMSE	
Domain	Model	EST	REAL	EST	REAL
Whole	Transformer	0.12	0.22	0.14	0.19
	LSTM	0.11	0.19	0.12	0.17
Small	Transformer	0.08	0.20	0.08	0.15
	LSTM	0.11	0.27	0.11	0.17
Medium	Transformer	0.16	0.30	0.18	0.22
	LSTM	0.14	0.29	0.15	0.20
Large	Transformer	0.12	0.39	0.12	0.19
	LSTM	0.09	0.27	0.09	0.17

Fig. 7: Model comparison among different domain sizes for Objective 3 according to estimated & real values

We also analyze how well the general models work against different opponents as far as the different characteristics of the opponents are concerned. Figure 8 shows the error distribution of the general models among different opponent strategies regarding the *Objectives* 1 and 2. We might think that learning the utilities of the opponent's next offers would be easier when they employ a time-based concession strategy since they are generated through a function of normalized negotiation. However, the *Objectives* 1 and 2 aim to learn the agent's utility, not the opponent's utility. Therefore, there may not be a regular pattern as far as the agent's utility is concerned due to varying opposition of the domains as shown in Table 2. In other words, the received utilities by the agent may have some fluctuations. On the other hand, we expect behavior-based strategies to show a more regular pattern since they consider the other side's utilities. Therefore, the predictors' performance against such agents is lower than the behavior-based agents. The performance of the predictors is better for the *Objective* 2 because of considering opponent-related features. While the models perform best against behavior-based opponents, their performance drastically drops due to the stochastic behavior shown by the hybrid agents.

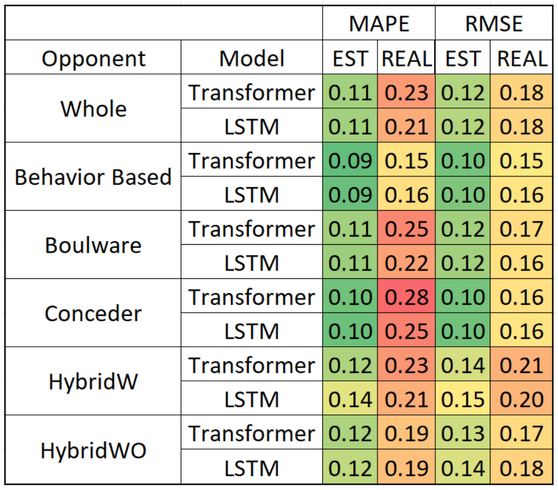


		MAPE		RMSE	
Opponent	Model	OBJ 1	OBJ 2	OBJ 1	OBJ 2
Whole	Transformer	0.22	0.15	0.14	0.10
	LSTM	0.21	0.15	0.13	0.10
Behavior Based	Transformer	0.21	0.17	0.14	0.12
	LSTM	0.19	0.16	0.13	0.11
Boulware	Transformer	0.29	0.18	0.15	0.10
	LSTM	0.29	0.19	0.15	0.10
Conceder	Transformer	0.29	0.21	0.15	0.12
	LSTM	0.28	0.21	0.15	0.12
HybridW	Transformer	0.27	0.25	0.18	0.17
	LSTM	0.25	0.23	0.17	0.16
HybridWO	Transformer	0.31	0.20	0.19	0.13
	LSTM	0.30	0.20	0.19	0.13

Fig. 8: Model comparison among different opponents for Objectives 1 & 2

Figure 9 shows the model comparison against each opponent strategy for *Objective* 3 and demonstrates the metrics for estimated and actual opponent utilities. In general, next-step utility predictions against Hybrid strategies demonstrate lower performance than time-based and behavior-based strategies, as expected, since they have higher complexity for involving time-based and behavior-based strategies. However, predictions against Hybrid strategies only perform better while comparing Objective 3 predictions with the real opponent utilities in terms of MAPE. This can stem from estimating the opponent's offers by the employed opponent model.

To sum up, it is observed that there is no significant difference between LSTM and Transformer results. RMSE and MAPE values were observed similarly for each objective, as shown in Figure 10. The similarity of the model performances can also be interpreted by the prediction examples shown in Figure 11. Although the models can be refined by adding more complexity and potentially achieving higher success, this study used basic architectures to focus on the prediction generalization capability on different domain sizes and employed opponent strategies.



		MAPE		RMSE	
Opponent	Model	EST	REAL	EST	REAL
Whole	Transformer	0.11	0.23	0.12	0.18
	LSTM	0.11	0.21	0.12	0.18
Behavior Based	Transformer	0.09	0.15	0.10	0.15
	LSTM	0.09	0.16	0.10	0.16
Boulware	Transformer	0.11	0.25	0.12	0.17
	LSTM	0.11	0.22	0.12	0.16
Conceder	Transformer	0.10	0.28	0.10	0.16
	LSTM	0.10	0.25	0.10	0.16
HybridW	Transformer	0.12	0.23	0.14	0.21
	LSTM	0.14	0.21	0.15	0.20
HybridWO	Transformer	0.12	0.19	0.13	0.17
	LSTM	0.12	0.19	0.14	0.18

Fig. 9: Model comparison among different opponents for Objective 3 according to estimated & real values

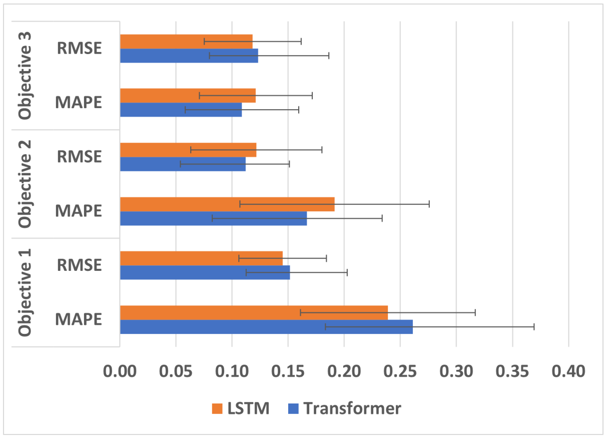
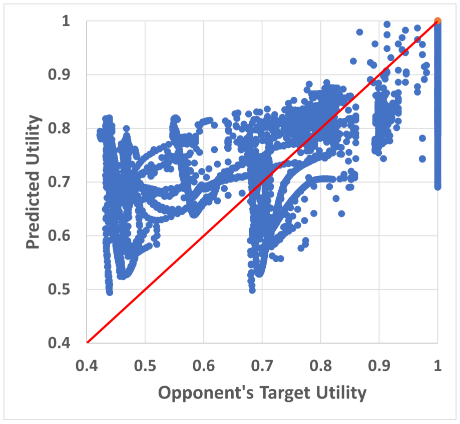


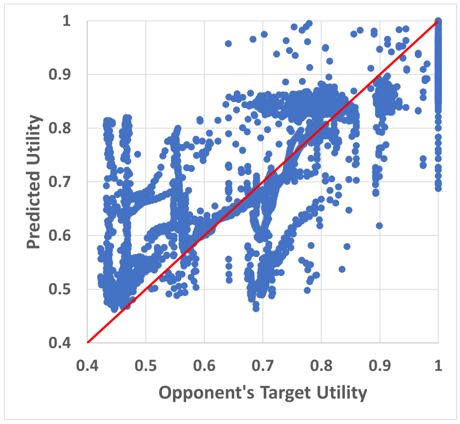
Fig. 10: Averaged RMSE and MAPE results models for different objectives

6 Conclusion

This study proposes time series predictive models, namely LSTM and Transformers, to guess the utility values of the opponent's coming offers during the bilateral negotiation. Three different objectives have been introduced with the motivation to foresee the utility of the opponent's next offer from the agent's and opponent's perspectives and investigate the effect of the opponent-related features. To assess the performance of the proposed approaches, we trained LSTM and Transformer models with data obtained from 600 negotiation sessions. The



(a) LSTM (RMSE: 0.135)



(b) Transformer (RMSE: 0.104)

Fig. 11: Comparing predicted utilities in Objective 3 for a large domain

results are analyzed elaborately by considering the effect of domain size and the opponent's strategy. The results support that the models can learn the utilities of the opponent's following offers to a certain extent. Since the opponent modeling significantly influences the estimated utilities, different opponent modeling approaches could be applied and compared in future work. Moreover, other time series prediction models could be used and evaluated.

The models used in the study are selected based on their sequence processing behaviors. LSTM models process the sequential utility values unidirectionally, whereas Transformers process the data in a bidirectional manner. The bidirectional behavior of Transformer models outperforms the LSTM models in some tasks such as Natural Language Processing (NLP) and Computer Vision. However, we observed that it does not significantly exceed the unidirectional processing behavior of LSTM models in next-utility value predictions. The situation might stem from the fact that most agents make decisions by only considering their previous and/or current rounds. In our opinion, if they act by considering their future steps, Transformers might perform better.

Bibliography

1. Arslan, F., Aydogan, R.: An actor-critic reinforcement learning approach for bilateral negotiation. Turkish Journal of Electrical Engineering and Computer Sciences pp. 1–20 (2022)
2. Aydoğan, R., Baarslag, T., Fujita, K., Mell, J., Gratch, J., de Jonge, D., Mohammad, Y., Nakadai, S., Morinaga, S., Osawa, H., Aranha, C., Jonker, C.M.: Challenges and main results of the automated negotiating agents competition (anac) 2019. In: Multi-Agent Systems and Agreement Technologies. pp. 366–381. Springer International Publishing (2020)

3. Aydoğan, R., Festen, D., Hindriks, K.V., Jonker, C.M.: Alternating offers protocols for multilateral negotiation. In: Modern Approaches to Agent-based Complex Automated Negotiation, pp. 153–167. Springer (2017)
4. Baarslag, T., Hendrikx, M., Hindriks, K.V., Jonker, C.M.: Learning about the opponent in automated bilateral negotiation: a comprehensive survey of opponent modeling techniques. Autonomous Agents and Multi-Agent Systems (2016)
5. Chen, S., Ammar, H., Tuyls, K., Weiss, G.: Transfer learning for bilateral multi-issue negotiation. In: Proceedings of the BNAIC 2012. pp. 59–66 (2012)
6. Eran, C., Keskin, M.O., Cantürk, F., Aydoğan, R.: A decentralized token-based negotiation approach for multi-agent path finding. In: Multi-Agent Systems. pp. 264–280. Springer International Publishing (2021)
7. Faratin, P., Sierra, C., Jennings, N.R.: Negotiation decision functions for autonomous agents. Robotics and Autonomous Systems **24**(3), 159–182 (1998)
8. Fatima, S., Kraus, S., Wooldridge, M.: Principles of automated negotiation. Cambridge University Press (2014)
9. Hindriks, K., Jonker, C., Tykhonov, D.: Let's dans! an analytic framework of negotiation dynamics and strategies. Web Intelligence and Agent Systems **9**, 319–335 (01 2011)
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation (1997)
11. Keskin, M.O., Çakan, U., Aydoğan, R.: Solver agent: Towards emotional and opponent-aware agent for human-robot negotiation. In: Proceedings of the AAMAS 2021. p. 1557–1559. AAMAS '21, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2021)
12. Li, M., Murukannaiah, P.K., Jonker, C.M.: A data-driven method for recognizing automated negotiation strategies. ArXiv **abs/2107.01496** (2021)
13. Lin, R., Kraus, S., Baarslag, T., Tykhonov, D., Hindriks, K.V., Jonker, C.M.: Genius: An integrated environment for supporting the design of generic automated negotiators. Computational Intelligence (2014)
14. Marsa-Maestre, I., Klein, M., Jonker, C.M., Aydoğan, R.: From problems to protocols: Towards a negotiation handbook. Decision Support Systems **60**, 39–54 (2014)
15. Sanchez-Anguix, V., Tunali, O., Aydoğan, R., Julián, V.: Can social agents efficiently perform in automated negotiation. Applied Sciences (2021)
16. Sengupta, A., Mohammad, Y., Nakadai, S.: An autonomous negotiating agent framework with reinforcement learning based strategies and adaptive strategy switching mechanism. CoRR **abs/2102.03588** (2021)
17. Tunalı, O., Aydoğan, R., Sanchez-Anguix, V.: Rethinking frequency opponent modeling in automated negotiation. In: PRIMA 2017: Principles and Practice of Multi-Agent Systems. pp. 263–279. Springer International Publishing (2017)
18. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. CoRR **abs/1706.03762** (2017)
19. Williams, C.R., Robu, V., Gerding, E.H., Jennings, N.R.: Using gaussian processes to optimise concession in complex negotiations against unknown opponents. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (15/07/11 - 21/07/11). pp. 432–438 (2011)
20. Wu, N., Green, B., Ben, X., O'Banion, S.: Deep transformer models for time series forecasting: The influenza prevalence case. CoRR (2020)
21. Zeng, A., Chen, M., Zhang, L., Xu, Q.: Are transformers effective for time series forecasting? (2022)