Transfer Learning Based Adaptive Automated Negotiating Agent Framework

Ayan Sengupta[1] , **Shinji Nakadai**[1,2] , **Yasser Mohammad**[1,2,3]
[1]NEC Corporation, Japan
[2]National Institute of Advanced Industrial Science and Technology (AIST), Japan
[3]Assiut University, Egypt
{a-sengupta, nakadai, y.mohammad}@nec.com

Abstract

With the availability of domain specific historical negotiation data, the practical applications of machine learning techniques can prove to be increasingly effective in the field of automated negotiation. Yet a large portion of the literature focuses on domain independent negotiation and thus passes the possibility of leveraging any domain specific insights from historical data. Moreover, during sequential negotiation, utility functions may alter due to various reasons including market demand, partner agreements, weather conditions, etc. This poses a unique set of challenges and one can easily infer that one strategy that fits all is rather impossible in such scenarios. In this work, we present a simple yet effective method of learning an end-to-end negotiation strategy from historical negotiation data. Next, we show that transfer learning based solutions are effective in designing adaptive strategies when underlying utility functions of agents change. Additionally, we also propose an online method of detecting and measuring such changes in the utility functions. Combining all three contributions we propose an adaptive automated negotiating agent framework that enables the automatic creation of transfer learning based negotiating agents capable of adapting to changes in utility functions. Finally, we present the results of an agent generated using our framework in different ANAC domains with 88 different utility functions each and show that our agent outperforms the benchmark score by domain independent agents by 6%.

1 Introduction

Negotiation is a process of decision making between two or more parties who aim to reach a mutually beneficial agreement. Automated negotiation involves negotiation among agents acting on behalf of real-world parties to achieve win-win deals for all, while simultaneously reducing the time and effort. Automated negotiation is starting to play an important role in industrial applications like supply chain [Fiedler and Sackmann, 2021], smart grid [Etukudor *et al.*, 2020], e-commerce [Cao *et al.*, 2015], task allocation [Krainin *et al.*, 2007], and autonomous driving [Chater *et al.*, 2018].

With the availability of negotiation history between two parties in a particular negotiation domain, it is natural to assume that one can leverage some insights from the past data which might prove to be useful in subsequent negotiations. The question that arises is can we learn an end-to-end negotiation strategy from such data? If yes, how can we use that practically when utility functions of agents change depending on unknown factors that are difficult to model? In this work, we have tried to address these questions with the application of transfer learning to adapt trained negotiation strategies. The contributions of this work to existing research are three-fold. Firstly, we propose a simplified deep learning architecture for training an end-to-end negotiation strategy. Secondly, we propose a method of adapting to changes in the utility function using transfer learning technique. Finally, We present an approach to detect and measure such changes in utility functions. Altogether with these three components we propose a framework to create domain specific adaptive autonomous negotiating agents capable of automatically learning end-to-end bidding strategies via transfer learning amid changing market conditions.

The rest of the paper is organized as follows: Section 2 gives a sketch of related work in this domain, Section 3 provides the introduction to preliminaries whereas Section 4 describes the framework and its components in details. Section 5 outlines the experimental setup, Section 6 shows the evaluations of our framework and finally, we conclude with Section 7 by discussing the limitations and provide directions for future work.

2 Related Work

In automated negotiation, few attempts have been made to use historical data to create a negotiation strategy. Authors of [Liu *et al.*, 2020] have proposed a decision tree based negotiation assistant bot which can be used for price prediction and was trained on pricing data of a car trading platform. Here the authors predict a fair price of a car and use a handcrafted negotiation strategy for assisted negotiation. In contrast, we train the end-to-end negotiation strategy itself from the historical data of negotiations. In [Lewis *et al.*, 2017], the authors have introduced an end-to-end natural language based negotiation model. One major drawback of their technique is the requirement of huge datasets for a particular negotiation domain. Their data collection approach is to collect data for different random negotiation scenarios so that the model can adapt to different utility functions during testing, but this is infeasible in real-life applications. In contrast, we propose a transfer learning based approach to adapt to new negotiation scenarios even with less historical data. Several studies on the application of Reinforcement Learning [Chang, 2021; Sengupta *et al.*, 2021] have shown promising results in designing negotiation strategies under stationary conditions. However, one major drawback is the requirement for a large number of interactions with the environment for training the policy. This motivated us to design a Transfer Learning based approach that can adapt to changing utility functions with a small amount of negotiation data.

In the last decade, Transfer Learning has witnessed many successful applications including sentiment classification [Liu *et al.*, 2019], image classification [Kulis *et al.*, 2011; Zhu *et al.*, 2011], multi-language text classification [Zhou *et al.*, 2014], and many more. More recently, Transfer Learning has been successfully applied in opponent modeling in automated negotiations [Chen *et al.*, 2012; Chen *et al.*, 2016]. Although many recent works focused on the application of Deep Learning and Reinforcement Learning in the creation of negotiation strategies, this work, to the best of our knowledge is the first attempt at applying transfer learning in designing end-to-end automated negotiation strategies under changing utility functions.

3 Preliminaries

A bilateral automated negotiation is a negotiation between two automated agents. Such repeated negotiation between two fixed parties is called a sequential negotiation. A negotiation scenario consists of the utility functions of each agent and the negotiation domain. The negotiation protocol used throughout this paper is the stacked alternating offers protocol [Aydoğan *et al.*, 2017].

A negotiation domain consists of one or more issues. The outcome space of a negotiation is the set of all possible negotiation outcomes and is defined as $\Omega = \{\omega_1, \cdots, \omega_n\}$ where ω_i is a possible outcome and n is the cardinality of outcome space. A utility function assigns a utility value to an outcome ω_i denoted by $U(\omega_i)$. Moreover, utility functions are private information and the agents have access to their own utility functions only. We define a partially ordered outcome set for an utility function U as $\Omega_U = \{\omega'_1, \cdots, \omega'_n \mid U(\omega'_1) \geq \cdots \geq U(\omega'_n) \quad \forall \quad \omega'_i \in \Omega, i \in [1, n]\}$. We will denote the final agreed offer as ω^* and the corresponding utility value as $U(\omega^*)$ for utility function U. We denote self utility function and opponent utility function as U^s and U^o respectively. Additionally, every agent also has a specified reservation value denoted by u_r which is the utility that the agent receives in case of no agreement. A trace of a negotiation denoted by $\mathcal{T}$ is a sequence of alternating bids by the agents for a single negotiation and is defined by $\mathcal{T} = \{\omega^o_1, \omega^s_1, \cdots, \omega^o_n, \omega^s_n\}$ for a bilateral negotiation where ω^o_i, ω^s_i are the offers by the opponent agent and self agent at step i respectively. $\mathcal{T}$ can be divided into two mutually exclusive and exhaustive sets

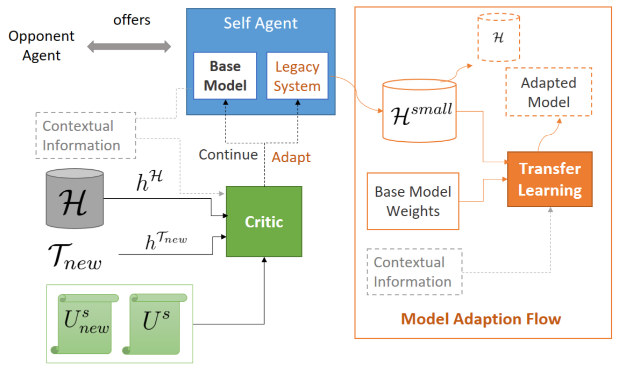


Figure 1: Block diagram of proposed Adaptive Automated Negotiating Agent Framework showing the following blocks: Critic (Green), Model Adaptation Flow (Orange Box), Base Model, Legacy System and extensions involving contextual information (gray dotted blocks).

$\mathcal{T}^s = \{\omega^s_1, \cdots \omega^s_n\}, \mathcal{T}^o = \{\omega^o_1, \cdots, \omega^o_n\}$ where $\mathcal{T}^s$ and $\mathcal{T}^o$ are traces containing self and opponent offers respectively. A negotiation history denoted by $\mathcal{H}$ is a set of negotiation traces in a particular negotiation domain D and is defined as $\mathcal{H}_D = \{\mathcal{T}_1, \cdots, \mathcal{T}_k\}$ where k is the number of traces in a negotiation history.

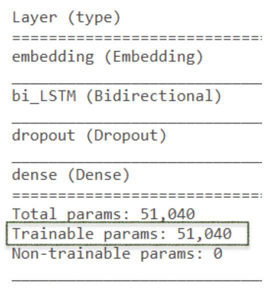
4 Framework

In this section, we explain each component of our Adaptive Automated Negotiating Agent Framework under changing utility functions. This framework enables automatic detection of changes in opponent utility functions and also measures the change in self utility functions. Additionally, it also provides a mechanism to automatically adapt to such changes by a novel adaption process. The proposed framework is comprised of three main components: the Critic, the Base Model, and the Model Adaptation Flow along with other auxiliary components like databases of negotiation traces, Legacy System, etc. Additionally, this framework can naturally handle contextual information and use it during both negotiations and training of a model. However, for showing the effectiveness of the framework as a whole, we will only focus on context-less negotiations for the rest of the paper. All components and auxiliary components are outlined in Figure 1 while each of them is discussed briefly in the remaining section.

4.1 Base Model

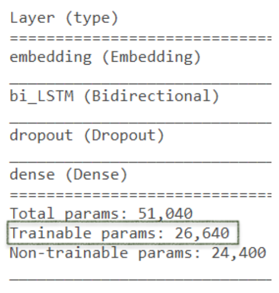
Base Model is a Deep Learning model that is being actively used by the agent for its bidding strategy. Initially, the Base Model is trained from the historical data, however, later it will be substituted by new adapted models whenever necessary. In this section, we propose a simple yet effective deep learning architecture for the Base Model where the model generates the next best offer for the agent given the current negotiation trace. The goal here is to provide a simple yet effective base architecture on which Model Adaptation by transfer learning techniques can be applied as described in Section 4.2.

For the model architecture, we have chosen a bidirectional Long short-term memory(LSTM) [Hochreiter and Schmidhu-



```
Layer (type)
============================
embedding (Embedding)
____________________________
bi_LSTM (Bidirectional)
____________________________
dropout (Dropout)
____________________________
dense (Dense)
============================
Total params: 51,040
Trainable params: 51,040
Non-trainable params: 0
____________________________
```

(a)



```
Layer (type)
=============================
embedding (Embedding)
_____________________________
bi_LSTM (Bidirectional)
_____________________________
dropout (Dropout)
_____________________________
dense (Dense)
=============================
Total params: 51,040
Trainable params: 26,640
Non-trainable params: 24,400
_____________________________
```

(b)

Figure 2: Figure showing (a) model summary for Base Model and (b) model summary for Adapted Model while using shared parameter based transfer learning with fewer trainable parameters.

ber, 1997] with an embedding layer at the beginning and a single Dense layer with *softmax* [Goodfellow *et al.*, 2016] activation at the end and the model summary is shown in Figure 2a. For utilizing past negotiation traces for training, first, from every $\mathcal{T} \in \mathcal{H}_D$ a set of input sequences and labels are generated where an input sequence I_t to the model is a sequence of offers till time t and is defined as $I_t = \{\omega_1^o, \omega_1^s, \cdots, \omega_t^o\}$ and the label $O_t = \{w_t^s\}$ is the next subsequent self offer in $\mathcal{T}$.We use weighted cross-entropy loss function with $(U^s(\omega_{\mathcal{T}}^*))^\kappa$ as loss weights to provide more weightage to traces with high agreement utility value, where κ is the exponent hyperparameter. Some preprocessing like mapping of offers to a numerical value and padding of each input sequence is done before providing them as input to the model. The effectiveness of this simple architecture comes from the fact that embedding layer successfully captures partial information about both U^s and U^o as illustrated in Figure 4 and described in Section 6.

4.2 Model Adaptation Flow and Adapted Model

In this section, we give an overview of the Model Adaptation Flow and the training of a new Adapted Model for the bidding strategy. Model Adaptation Flow comprises of two steps. Firstly, a Legacy System is temporarily used in-place of the Base Model for negotiations, that successfully negotiates with the opponent for a handful number of times to generate a set of small number of negotiation traces with agreements and we will denote that set by $\mathcal{H}^{small}$. Moreover, Legacy System can be comprised of single or multiple instances of any type of compatible automated negotiating agents.

In the second step, we adapt our Base Model using parameter sharing based transfer learning technique using the fresh dataset $\mathcal{H}^{small}$ for training. With changes in the utility function the performance of the Base Model declines majorly due to the use of stale embedding layer. Since the embedding layer captures partial information of both utility functions, it is essential to retrain the embedding layer when the utility function changes. To achieve this we propose to retrain both the embedding layer and the dense layer to recapture the new information about the modified opponent or self utility function. Assuming that the opponent strategy is fixed, we will reuse the weights and biases of bidirectional LSTM layer from the Base Model. This reduces the total number of trainable parameters enabling the model to learn faster and with less amount of data. In practice, as a final step of transfer learning one can fine tune the whole model with a very small learning rate and for small number of epochs. Though this results in better model accuracy, utilities of final agreements don't show changes of any statistical significance when compared with agreements reached with models without finetuning.

After the Adapted Model has been created through the Model Adaptation flow, this model is used as the Base Model from the next negotiation and the history of negotiation data $\mathcal{H}$ will be replaced by $\mathcal{H}^{small}$ as shown in Figure 1. The input and output of the model is same as described in Section 4.1. The model architecture for transfer learning with reduced number of training parameters is shown in Figure 2b.

4.3 The Critic

The Critic can be modeled as a binary classifier that decides weather the agent should continue with the Base Model or should initiate the Model Adaptation Flow. We realistically assume that the utility functions of the opponent or self don't change significantly for a set of consecutive negotiations. For simplicity, we also assume that only one of the utility functions change at one instance. Let $h^{\mathcal{T}} = \{f^{\mathcal{T}}(\omega_1^o), \cdots, f^{\mathcal{T}}(\omega_n^o)\}$ be the the frequency distribution of opponent offers where $f^{\mathcal{T}}(.)$ calculates the number of occurrence of an offer in a negotiation trace $\mathcal{T}$. Now, $h^{\mathcal{H}} = \{\frac{1}{m}\sum_{j=1}^{m} f^{\mathcal{T}_j}(\omega_1^o), \cdots, \frac{1}{m}\sum_{j=1}^{m} f^{\mathcal{T}_j}(\omega_n^o) \mid \mathcal{T}_j \in \mathcal{H} \;\forall\; j \in [1, m]\}$ denotes the average frequency distribution of opponent offers in history of negotiation $\mathcal{H}$ with cardinality m. The input to the Critic is the tuple $\{h^{\mathcal{H}}, h^{\mathcal{T}_{new}}\}$ where $\mathcal{T}_{new}$ is the trace of latest negotiation and the output is binary with 0 meaning to continue the use of Base Model and 1 indicating to generate an Adapted Model. Note that, one can also use a set of last few traces instead of greedily using the last trace. For our experiments in section 5 we used *XGBoost*[Chen and Guestrin, 2016] based classifier for training the Critic where the training data is a synthetically generated dataset using different utility functions with $\{h^{\mathcal{H}_1}, h^{\mathcal{H}_2}\}$ as input and a binary label as output. The binary labels are generated by checking if Base Model trained with $h^{\mathcal{T}_1}$ is better than Adapted Model with $h^{\mathcal{T}_2}$. However, during cold start of the framework or when enough data is not available for the training of a classifier we can use a Wasserstein distance [Ramdas *et al.*, 2015] $\mathbb{W}(h_1, h_2)$ or Energy distance [Székely, 2003] $\mathbb{E}(h_1, h_2)$ based algorithm for the Critic as provided in Algorithm 1. It leans on the fact that offer distribution changes for dissimilar U^o and can get a measure of such a change by using any metric capable of measuring distance between two distributions.

An extension of this Critic can also accommodate the changes in self utility function U^s. One naive way is to always train a new model when self utility function changes, i.e. $U^s_{new} \neq U^s$. However, small changes in utility function may not effect the performance of the Base Model at all. This is due to the fact that a change in self utility function may not affect the utility values of offers in $\mathcal{T}^s$ significantly to alter the performance of the Base Model. Now, to measure the

Algorithm 1 Single metric based Critic algorithm for opponent utility function

```
Input: $\{h^{\mathcal{H}}, h^{\mathcal{T}}_{new}\}$
Parameter: metric: $\{M, \alpha_M\}$ where $M \in [\mathbb{W}, \mathbb{E}]$
Output: 0 (Base Model) or 1 (Model Adaptation Flow)
1: $m \longleftarrow M(h^{\mathcal{H}}, h^{\mathcal{T}}_{new})$
2: if $m < \alpha_M$ then
3:     return 0
4: else
5:     return 1
6: end if
```

Algorithm 2 Critic algorithm for self utility function

```
Input: $\{U^s_{new}, U^s\}$
Parameter: Threshold $\alpha_{\mathfrak{L}}$, maximum agreement utility in $\mathcal{H}$ denoted by $u^{\mathcal{H}}_{max}$ and hyperparameters $\gamma_1$ and $\gamma_2$ where $\gamma_1 > \gamma_2 > 0$.
Output: 0 (Base Model) or 1 (Model Adaptation Flow)
1: $i \longleftarrow \underset{k}{\text{argmin}}(\Omega_{U_s}[k] - \gamma_1 \times u^{\mathcal{H}}_{max})$
2: $j \longleftarrow \underset{k}{\text{argmin}}(\Omega_{U_s}[k] - \gamma_2 \times u^{\mathcal{H}}_{max})$
3: $l \longleftarrow \mathfrak{L}_{i:j}(U^s_{new}, U^s)$
4: if $l < \alpha_{\mathfrak{L}}$ then
5:     return 0
6: else
7:     return 1
8: end if
```

change between two utility functions U_1 and U_2 we introduce a new metric $\mathfrak{L}_{i:j}(U_1, U_2) \in [0, 1]$ based on the Levenshtein distance [Levenshtein and others, 1966]. Let $|o|$ denotes the length of a partially ordered set o and $o[k]$ denotes the k^{th} element of o where $k \in [0, |o| - 1]$. $o[k : l]$ is a subset of o and is defined as $o[k : l] := \{o[p] \ \forall \ k \leq p < l\}$. Then,

$$\mathfrak{L}_{i:j}(U_1, U_2) = 1 - \frac{L(\Omega_{U_1}[i : j], \Omega_{U_2}[i : j])}{|\Omega_U[i : j]|},$$

$$\text{where } |\Omega_U[i : j]| = |\Omega_{U_1}[i : j]| = |\Omega_{U_2}[i : j]|$$

and L is defined as

$$L(o_1, o_2) = \begin{cases} |o_1| & \text{if } |o_2| = 0, \\ |o_2| & \text{if } |o_1| = 0, \\ L(o_1[1 : |o_1|], o_2[1 : |o_2|]) & \text{if } o_1[0] = o_2[0], \\ 1 + min \begin{cases} L(o_1[1 : |o_1|], o_2), \\ L(o_1, o_2[1 : |o_2|]), \\ L(o_1[1 : |o_1|], o_2[1 : |o_2|]) \end{cases} & \text{otherwise.} \end{cases}$$

Illustrating using a simple example, $\mathfrak{L}_{0:n}(U_1, U_2)$ with $n < |\Omega_{U_1}|$ compares the top n outcomes for both utility functions by calculating the number of editing operations (insertions, deletions or substitutions) required to transform $\Omega_{U_1}[0 : n]$ to $\Omega_{U_2}[0 : n]$. The detailed algorithm is given in Algorithm 2.

5 Experimental Setup

In this section we briefly describe the experimental setup for our evaluations. We analyzed our proposed framework using 3 ANAC 2015 domains with 88 opponent utility functions and 88 self utility functions for each domain. The details of the domain including the opposition range and median $\mathfrak{L}$ values are provided in Table 1. Our experiments are split in two parts, first set of experiments aim to measure performance of overall framework with changes in U^o while the second set of experiments aim to test the overall framework with changes in U^s.

For experiments considering changes in U^o, the U^s and opponent strategy is kept fixed throughout the experiment. Initially for training the Base Model, $\mathcal{H}_D$ is populated by simulating negotiations between agents from our Legacy System and the fixed opponent strategy in a round robin fashion till $|\mathcal{H}^{pre}_D| = 100$ for each domain. Our Legacy System consists of 5 popular GENIUS [Lin *et al.*, 2014] agents namely Atlas3, YXAgent, PonpokoAgent, NiceTitforTat and ParsCat[1], whereas we use Atlas3 for the opponent strategy. Following the data generation a Base Model is trained. Next, using the Model Adaptation Flow we trained 88 adapted models for each domain and compared the results of all the adapted models to that of the Legacy System. During training we used weighted crossentropy loss function with $\kappa = 4$. Additionally, all used hyperparameters remained same for each training. Moreover, to truly gauge the potential of transfer learning based bidding strategies, we did not use any acceptance strategy for our agent while evaluating, that is, our agent never accepts. Moreover, for all utility functions, we assumed $u_r = 0$ and discount factor to be 1. For testing the overall framework in a more realistic way, we changed the opponent utility function after every 10 rounds of negotiation. For testing the performance of our framework with changes in U^s, we assume that the U^o and strategy are not changing. The model creation, experimental setup and evaluation is otherwise similar.

For the training of the Critic in the condition when U^o changes, first we created a dataset with two opponent offer distribution as features and binary labels with 0 meaning that Base Model will be used and 1 meaning the activation of Model Adaptation Flow. After that we used XGBoost to train a classifier. Our experiments also include another version of Critic where we use Algorithm 1 with Wasserstein Distance and α_M as 3, 82 and 538 for the domains *Bank Robbery*, *car domain* and *Tram* respectively. For our experiments optimal α_M was chosen after plotting the Wasserstein Distance with respect to labels but in practice it may need to be tuned after certain intervals. Next, for the condition when self utility functions are changing we used the Algorithm 2 with $\gamma_1 = 0.7, \gamma_2 = 1$ and $\alpha_{\mathfrak{L}} = 0.2$. Furthermore, all simulations regarding negotiations are done using NegMAS [Mohammad *et al.*, 2021] library, where as all deep learning models were created using Tensorflow [Abadi *et al.*, 2015].

6 Results

In this section we present the results obtained by following the experimental setup mentioned in Section 5. In the first

[1]ANAC winners: Atlas3 (2015 winner), ParsCat (2016 2nd position), AgentYX (2016 2nd position) and PonpokoAgent (2017 winner)

Domain	Opposition	$\|\Omega\|$	Median $\mathfrak{L}$
Bank Robbery	0 - 0.256	18	0.273
Car Domain	0.002 - 0.222	240	0.079
Tram	0 - 0.066	972	0.054

Table 1: Overview of domains with opposition values when U^o is changing, cardinality of outcome space and median $\mathfrak{L}$ values when U^s is changing.

part, we will compare the performance of our Adapted Models with the Legacy System in three separate ANAC 2015 domains under changing U^o. Following that we present a full comparative study of our framework when U^o is changing. Finally, we present another comparative study of our framework when U^s is changing.

6.1 Performance of Adaptive Models

In this section, we compare our transfer learning based Adapted Models with the Legacy System. Figure 3 shows the box plots of average utility values reached by agents using Adapted Models and the agents from the Legacy System for each domain. Note that though the Model Adaptation Flow actually uses the negotiation traces generated by the Legacy System for training, yet the median utility value of Adapted Model is higher than that of Legacy System for each domain, and overall approximately 6% higher in average. This shows that parameter share based transfer learning is effective in adapting to changes in the utility function. The successful performance of our simple architecture is due to the fact that embedding layer successfully captures partial information about the utility functions of both the agents. Figure 4 shows the PCA plot of the embedding layer of one such model trained on the car domain. In Figure 4, the offers with highest utilities for the opponent utility function is encircled in region A while region B encircles the offers with highest utilities for the self utility function. We can notice the distinct separation between the two regions and since this is found to be true for other embedding layers for all domains as well, we can empirically state that the embedding layer of our models learns a partial representation of both the utility functions. Therefore, it is evident that any changes in utility function demands a retraining of the embedding layer to recapture new representation for the new pair of utility functions. Moreover, the long whiskers in Figure 3 for Adapted Model are partially due to the absence of acceptance condition and partially due to unoptimized hyperparameters during transfer learning.

6.2 Performance of Proposed Framework with Changes in Opponent Utility Function

In this section, we will present the comparison results of the whole framework with different Critic settings and cross-compare all the variations of the framework including the Legacy System. As mentioned in Section 5, in this experiment the opponent utility functions are kept fixed for certain negotiation rounds before being changed. Now in Figure 5 we have shown the median scores achieved by our framework in different configurations along with the median score of the Legacy System. First comparison is with Critic output fixed

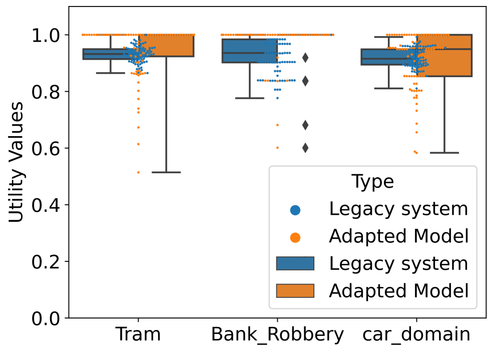


Figure 3: Box plots showing the average agreement utility values for Transfer Learning based bidding strategies and the benchmark scores by the agents of Legacy System in three separate domains.

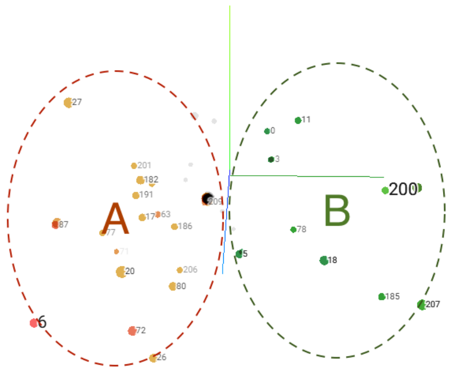


Figure 4: PCA plot of embedding layer for a single model highlighting offers of opponent agent with high utilities (Region A) and offers with high utilities for self agent (Region B) for a single pair of utility functions in Car domain.

at 0, that is, we use the Base Model for all the negotiations while opponent utility function changes. Next, we plot the median average utility obtained by our Legacy System. Following that, we compare the performance of our framework with Critic output fixed at 1, that is, we always use Model Adaptation Flow for each change in U^o. Next, we compare our framework's performance once with Wasserstein distance based Critic and again with *XGBoost* based Critic. The first thing to note in all three domains is that the performance of the overall framework with both the Critic algorithms is similar to the performance of Critic output fixed at 1. This indicates that, the Critic is not depreciating the performance when in fact it saves valuable computation resource and time by intelligently choosing to continue with Base Model. Considering the percentage of initialization of Model Adaptation Flow in Critic output fixed at 1 to be 100%, the average initialization for our Critic implementation is approximately 33%, 50% and 7% for the domains *Bank Robbery*, *car domain* and *Tram* respectively. It is to be noted that though Wasserstein distance based Critic is quite effective, and works in par with *XGBoost* based Critic, the threshold parameter is domain dependent and needs to be optimized on a regular in-

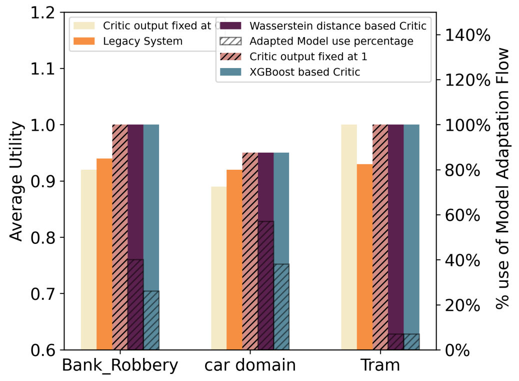


Figure 5: Figure comparing the median average utility achieved by agent using the proposed framework with Critic output fixed at 0 (Base Model), Critic Output fixed at 1 (Adapted Model), Wasserstein distance and *XGBoost* based Critic and Legacy System

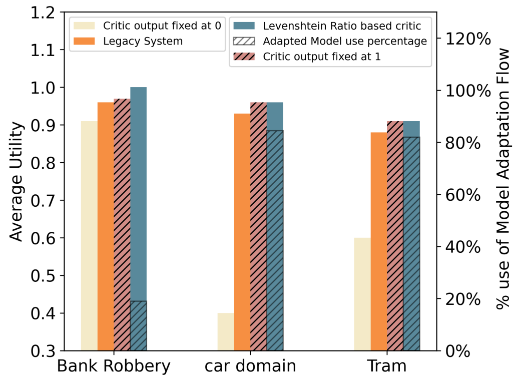


Figure 6: Figure comparing the median average utility achieved by agent using the proposed framework with Critic output fixed at 0 (Base Model), Critic Output fixed at 1 (Adapted Model), Levenshtein ratio based Critic and Legacy System

terval when enough data points are not available. Moreover, the median average utility of the overall framework with an intelligent Critic is more than the Legacy System by approximately 6% on average. This shows the effectiveness of our Transfer Learning based Adaptive Autonomous Negotiating Agent Framework over standard domain dependent agents.

Moving on, it is quite natural to assume that the Base Model will not perform adequately when opponent utility functions change over time, and can also be verified from Figure 5 for the domains *Bank Robbery* and *Car domain*. However, in the case of domain *Tram*, the score of Base Model is not only better than Legacy System but also in par with the overall framework and this is because of very low opposition scores compared to other domains as shown in Table 1. This in turn is also the reason behind low percentages of activation of Model Adaptation Flow while using our framework for *Tram* domain.

6.3 Performance of Full Framework with Changes in Self Utility Function

In this section we briefly provide the results of the whole framework with the Critic with Algorithm 2 while U^s is changing. Figure 6 compares the median score of our framework with different Critic settings along with the median score of the Legacy System. One can see that the Critic has on average performed 2% better than the Legacy System. Additionally, when Critic output is fixed at 0, i.e., when only Base Model is used of all experiments then performance will suffer over time if the U^s changes too much as we can see for the case of *Tram* and *car domain*. However, the median score of the Base Model didn't suffer in case of domain *Bank Robbery* because the changes in the utility function is low as measured by metric $\mathfrak{L}$ and shown in Table 1. We can note that when Critic output is fixed at 1, i.e. when only Adapted Model is used for evaluation the median score is more than the Base Model and Legacy system from all 3 domains. Considering the percentage of initialization of Model Adaptation Flow in Critic output fixed at 1 to be 100%, the initialization in the case of our Critic implementation is approximately 19%, 88% and 90% for the domains *Bank Robbery*, *car domain* and *Tram* respectively. This aligns with the intuition that when Base Model itself will perform well on a set of utility functions, then initialization of Model Adaption flow is not required and can hence conclude that the Levenshtein Ratio based Critic can switch intelligently. Also note that unlike Wasserstein distance based Critic proposed in Algorithm 1, the threshold $\alpha_{\mathfrak{L}}$ is domain independent.

7 Conclusion and Future Work

In this work we proposed a Transfer Learning based adaptive autonomous negotiating agent framework which is capable of not only learning bidding strategies from historical negotiation data but also adapting to changes in opponent or self utility function. For such adaptation we introduced a novel Model Adaptation Flow which automatically generates Adapted Strategies by parameter sharing based transfer learning. Additionally, to detect such changes in opponent utility function, we proposed a Critic and presented two different algorithms. Furthermore, for detecting significant changes in self utility function, we introduced a new Levenshtein based metric for measuring a change in self utility function.

This work has shown promising results in the application of transfer learning in automated negotiation. We also showed evidence that embedding layers learn partial representation of utility functions. As part of the future work, it would be interesting to test other methods of representation learning in automated negotiation. Apart from that, while designing this framework we did not consider changes in opponent strategy, but in reality strategies may change over time. Methodologies for adapting to such changes or designing components that are inherently inert to changes in opponent strategy is a real challenge and would be the next line of focus as an extension to our current work.

References

[Abadi *et al.*, 2015] Martín Abadi, Ashish Agarwal, Paul Barham, and ET al. TensorFlow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/, 2015. Accessed: 2022-05-17.

[Aydoğan *et al.*, 2017] Reyhan Aydoğan, David Festen, Koen V Hindriks, and Catholijn M Jonker. Alternating offers protocols for multilateral negotiation. In *Modern Approaches to Agent-based Complex Automated Negotiation*, pages 153–167. Springer, Berlin, Germany, 2017.

[Cao *et al.*, 2015] Mukun Cao, Xudong Luo, Xin Robert Luo, and Xiaopei Dai. Automated negotiation for e-commerce decision making: a goal deliberated agent architecture for multi-strategy selection. *Decision Support Systems*, 73:1–14, 2015.

[Chang, 2021] Ho-Chun Herbert Chang. Multi-issue negotiation with deep reinforcement learning. *Knowledge-Based Systems*, 211:106544, 2021.

[Chater *et al.*, 2018] Nick Chater, Jennifer Misyak, Derrick Watson, Nathan Griffiths, and Alex Mouzakitis. Negotiating the traffic: Can cognitive science help make autonomous vehicles a reality? *Trends in Cognitive Sciences*, 22(2):93–95, 2018.

[Chen and Guestrin, 2016] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[Chen *et al.*, 2012] Siqi Chen, Haitham Bou Ammar, Karl Tuyls, and Gerhard Weiss. Transfer learning for bilateral multi-issue negotiation. In *Proceedings of the 24th Benelux Conference on Artificial Intelligence (BNAIC)*, pages 59–66, 2012.

[Chen *et al.*, 2016] Siqi Chen, Shuang Zhou, Gerhard Weiss, and Karl Tuyls. Using transfer learning to model unknown opponents in automated negotiations. In *Recent Advances in Agent-based Complex Automated Negotiation*, pages 175–192. Springer, 2016.

[Etukudor *et al.*, 2020] Christie Etukudor, Benoit Couraud, Valentin Robu, Wolf-Gerrit Früh, David Flynn, and Chinonso Okereke. Automated negotiation for peer-to-peer electricity trading in local energy markets. *Energies*, 13(4):920, 2020.

[Fiedler and Sackmann, 2021] Alexandra Fiedler and Dirk Sackmann. Automated negotiation for supply chain finance. In Martijn Mes, Eduardo Lalla-Ruiz, and Stefan Voß, editors, *Computational Logistics*, pages 130–141, Cham, 2021. Springer International Publishing.

[Goodfellow *et al.*, 2016] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Krainin *et al.*, 2007] Michael Krainin, Bo An, and Victor Lesser. An application of automated negotiation to distributed task allocation. In *2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'07)*, pages 138–145. IEEE, 2007.

[Kulis *et al.*, 2011] Brian Kulis, Kate Saenko, and Trevor Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *CVPR 2011*, pages 1785–1792. IEEE, 2011.

[Levenshtein and others, 1966] Vladimir I Levenshtein et al. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.

[Lewis *et al.*, 2017] Mike Lewis, Denis Yarats, Yann N Dauphin, Devi Parikh, and Dhruv Batra. Deal or no deal? end-to-end learning for negotiation dialogues. *arXiv preprint arXiv:1706.05125*, 2017.

[Lin *et al.*, 2014] Raz Lin, Sarit Kraus, Tim Baarslag, Dmytro Tykhonov, Koen Hindriks, and Catholijn M. Jonker. Genius: An integrated environment for supporting the design of generic automated negotiators. *Computational Intelligence*, 30(1):48–70, 2014.

[Liu *et al.*, 2019] Ruijun Liu, Yuqian Shi, Changjiang Ji, and Ming Jia. A survey of sentiment analysis based on transfer learning. *IEEE Access*, 7:85401–85412, 2019.

[Liu *et al.*, 2020] Tingwei Liu, Zheng Zheng, et al. Negotiation assistant bot of pricing prediction based on machine learning. *International Journal of Intelligence Science*, 10(02):9, 2020.

[Mohammad *et al.*, 2021] Yasser Mohammad, Shinji Nakadai, and Amy Greenwald. Negmas: A platform for automated negotiations. In *PRIMA 2020: Principles and Practice of Multi-Agent Systems: 23rd International Conference, Nagoya, Japan, November 18–20, 2020, Proceedings 23*, pages 343–351. Springer International Publishing, 2021.

[Ramdas *et al.*, 2015] Aaditya Ramdas, Nicolas Garcia, and Marco Cuturi. On wasserstein two sample testing and related families of nonparametric tests. https://arxiv.org/abs/1509.02237, 2015.

[Sengupta *et al.*, 2021] Ayan Sengupta, Yasser Mohammad, and Shinji Nakadai. An autonomous negotiating agent framework with reinforcement learning based strategies and adaptive strategy switching mechanism. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, page 1163–1172, Richland, SC, 2021. International Foundation for Autonomous Agents and Multiagent Systems.

[Székely, 2003] Gábor J Székely. E-statistics: The energy of statistical samples. *Bowling Green State University, Department of Mathematics and Statistics Technical Report*, 3(05):1–18, 2003.

[Zhou *et al.*, 2014] Joey Tianyi Zhou, Sinno Jialin Pan, Ivor W Tsang, and Yan Yan. Hybrid heterogeneous transfer learning through deep learning. In *Twenty-eighth AAAI conference on artificial intelligence*, 2014.

[Zhu *et al.*, 2011] Yin Zhu, Yuqiang Chen, Zhongqi Lu, Sinno Jialin Pan, Gui-Rong Xue, Yong Yu, and Qiang Yang. Heterogeneous transfer learning for image classification. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.