Online learning and transfer for user adaptation in dialogue systems

Nicolas Carrara
Orange Labs - NaDia
Univ. Lille, CNRS,
Centrale Lille,
INRIA UMR 9189 - CRIStAL
F-59000 Lille, France
nicolas.carrara@orange.com

Romain Laroche
Microsoft Maluuba
Montréal, Canada
romain.laroche@microsoft.com

Olivier Pietquin*
Univ. Lille, CNRS, Centrale Lille,
INRIA UMR 9189 - CRIStAL
F-59000 Lille, France
olivier.pietquin@univ-lille1.fr

Abstract

We address the problem of user adaptation in Spoken Dialogue Systems. The goal is to quickly adapt online to a new user given a large amount of dialogues collected with other users. Previous works using Transfer for Reinforcement Learning tackled this problem when the number of source users remains limited. In this paper, we overcome this constraint by clustering the source users: each user cluster, represented by its centroid, is used as a potential source in the state-of-the-art Transfer Reinforcement Learning algorithm. Our benchmark compares several clustering approaches, including one based on a novel metric. All experiments are led on a negotiation dialogue task, and their results show significant improvements over baselines.

1 Introduction

Most industrial dialogue systems use a generic management strategy without accounting for diversity in user behaviours. Yet, inter-user variability is one of the most important issues preventing adoption of voice-based interfaces by a large public. We address the problem of Transfer Learning (Taylor and Stone, 2009; Lazaric, 2012) in Spoken Dialogue Systems (SDS) (Gašić et al., 2013; Casanueva et al., 2015; Genevay and Laroche, 2016), and especially the problem of fast optimisation of user-adapted dialogue strategies (Levin and Pieraccini, 1997) by means of Reinforcement Learning (RL) (Sutton and Barto, 1998). The main goal of this paper is to improve cold start (also called jumpstart in the literature) learning of RL-based dialogue management strategies when facing new users, by transferring data collected from similar users (Lazaric et al., 2008). To do so, we consider the setting in which a large amount of dialogues has been collected for a several users, and a new user connects to the service (Genevay and Laroche, 2016). Our solution combines techniques from the multi-armed bandit (Auer et al., 2002), batch RL (Li et al., 2009; Chandramohan et al., 2010; Pietquin et al., 2011) and policy/MDP clustering (Chandramohan et al., 2012; Mahmud et al., 2013) literatures.

Instead of clustering user behaviours as in (Chandramohan et al., 2012), we propose to cluster the policies that are trained on the user dialogue datasets. To do so, we define a novel policy-based distance, called PD-DISTANCE. Then, we investigate several clustering methods: k-medoids and k-means, which enable the identification of source representatives for the transfer learning. Once clusters representatives have been selected, they are plugged into a multi-armed bandit algorithm, as proposed in Genevay and Laroche (2016).

Following previous work where user adaptation (Janarthanam and Lemon, 2010; Ultes et al., 2015) was used to address negotiation tasks (Sadri et al., 2001; Georgila and Traum, 2011; Barlier et al., 2015; Genevay and Laroche, 2016), we test our methods on different types of users involved in a negotiation game (Laroche and Genevay, 2017). Methods are compared to two baselines: learning without transfer and transfer from a generic policy learnt from all the sources. These methods are tested by interacting with handcrafted users and human-model users learnt from actual human interactions (unlike Genevay and Laroche (2016)). Results show that our clustering methods provide a better dialogue experience than the generic methods in both setups.

After recalling mathematical background in Section 2, we present the full user adaptation process in Section 3. The clustering methods are described

*now with DeepMind, London

in Section 4. Section 5 describes the negotiation game and experiments are summarised.

2 Reinforcement learning

A **Markov Decision process (MDP)** is used for modelling sequential decision making problems. It is defined as a tuple $\{\mathcal{S}, \mathcal{A}, R, P, \gamma\}$; $\mathcal{S}$ is the state set, $\mathcal{A}$ the actions set, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ the reward function, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ the Markovian transition function and γ the discount factor. $\pi : \mathcal{S} \to \mathcal{A}$ is called a policy, which can be either deterministic or stochastic. Solving an MDP consists in finding a policy π^* that maximises the γ-discounted expected return $\mathbb{E}_{\pi^*} \sum_t \gamma^t R(s_t, a_t, s_{t+1})$. The policy π^* satisfies Bellman's optimality equation (Bellman, 1956):

$$\pi^*(s) = \underset{a \in A}{\operatorname{argmax}}\, Q^*(s, a), \tag{1}$$

$$Q^*(s,a) = \sum_{s' \in \mathcal{S}} \left[R\left(s, a, s'\right) + \gamma P\left(s, a, s'\right) Q^*\left(s', \pi^*(s')\right) \right], \tag{2}$$

which is equivalent to $Q^* = T^* Q^*$ where Q^* is the optimal Q function and T^* the Bellman optimality operator. If $\gamma < 1$, this operator is a contraction. Thanks to the Banach theorem, it admits a unique solution. Then, one can find Q^* by iterating on equation 2 : the algorithm is called **Value-Iteration (VI)**. When $\mathcal{S}$ is continuous, the previous algorithm cannot apply.

Fitted Value-iteration (FVI) is used instead. It learns the Q-function at each iteration using a supervised learning algorithm which map some (s, a) couples to their respective value in equation 2 involving R and P . However, in reinforcement learning, R and P are unknown so one must estimate the value.

Fitted-Q resolves the aforementioned problem. Given a batch of samples $(s_j, a_j, r'_j, s'_j)_{j \in [0,N]}$, it learns the Q-function at each iteration of **VI** given the learning batch $\{(s_j, a_j),\ r'_j + \gamma * \max_{a'} Q(s'_j, a')\}_{j \in [0,N]}$ using a supervised learning algorithm, trees for example (Ernst et al., 2005).

Linear least-squares-based Fitted-Q is a special case of Fitted-Q Iteration where Q is represented by a linear parametrisation :

$$Q_{\theta_i}(s, a) = \sum_i \theta_i \phi_i(s, a) = \theta^T \phi(s, a), \tag{3}$$

with $\phi(s_j, a_j) = \phi_j$ (ϕ is called the feature function). Least-square optimisation results in computing θ. Let $M = \left(\sum_{j=1}^N \phi_j \phi_j^T\right)^{-1}$, then :

$$\theta_i = M \sum_{j=1}^{N} \phi_j \left(r'_j + \gamma \max_{a \in A} \left(\theta_{i-1}^T \phi\left(s'_j, a\right)\right)\right), \tag{4}$$

The algorithm terminates when either one of the two following conditions is satisfied: $i \geq max_{it}$ or $||\theta_i - \theta_{i-1}||_2 \leq \delta$. A regularisation parameter λ can be added to the co-variance matrix to avoid divergences of θ_i's values (Tikhonov, 1963; Massoud et al., 2009). Least-squares-based Fitted-Q Iteration is denoted as Fitted-Q in this paper. In the next section, the full adaptation process from (Genevay and Laroche, 2016) is recalled and adapted.

3 Adaptation process

Figure 1 shows the full process of user adaptation. We remind the reader that our goal is to improve cold start by transferring data from existing users and learn a policy adapted to a new user by RL. As an input, we assume the existence of a database of dialogues with different users, which allows the training of user specialised policies. At first, the process consists in searching or constructing policy representatives for this database so as to reduce the number of possible transfer sources. This is where the contribution of this paper mainly stands, the rest being mostly inherited from (Genevay and Laroche, 2016):

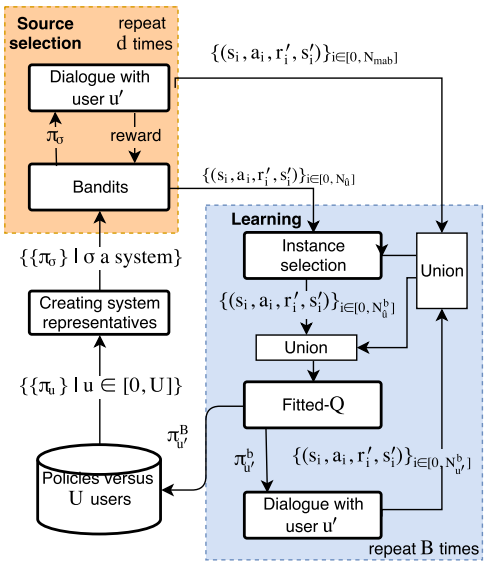


Figure 1: Adaptation process

Source selection The source selection problem is cast into a **multi-armed bandit algorithm (MAB)**, implemented here as UCB1 (Auer et al., 2002), each arm standing for a representative. When the MAB selects an arm, its corresponding policy π interacts with the user for one full dialogue. The MAB performs n_{mab} policy selections. N_{mab} samples from dialogues, with target user u', are collected during this procedure. In the end of this initial MAB step, the representative policy that yielded the highest empirical reward designates the source from which to transfer. The algorithm transfers $N_{\hat{u}}$ samples from its source $\hat{u}$ dialogues, to construct a batch of dialogues. Transitions from the trajectories of the chosen source are added to those already collected from the target as suggested by (Lazaric et al., 2008).

Instance selection Source transitions are subject to an instance selection to alleviate bias when sufficient target data has collected. After instance selection, the $N^b_{\hat{u}}$ remaining samples are added to the target samples for training a first policy with Fitted-Q. The idea is to only transfer transitions that are not present in the target transition dataset. Given a parameter η and given a transition from the source (s, a, r', s'), all the transitions from the target MDP which contain action a are considered. If there is a source transition (s_i, a, r'_i, s'_i) such that $||s - s_i||_2 \leq \eta$ then the transition is not added to the batch. The choice of η is problem-dependent and should be tuned carefully. A large value for this parameter leads to adding too few transitions to the batch, while a small value will have the opposite effect.

The hybrid source-target dataset is used for training the current policy that controls the behaviour during the next epoch, with an ϵ-greedy exploration: at each transition, with probability ϵ, a random action is chosen instead. $N^b_{u'}$ samples are collected this way, and used to refine its training. The algorithm repeats the operation from the transition selection step for every batch $b \in [0, B]$. Eventually, the final learnt policy $\pi^B_{u'}$ on u' is added to the database. Note that this policy does not explore anymore.

4 Source representatives

This section presents the main contributions of the paper. The adaptation process requires a setup of several source representatives in order to do the first dialogues, with a target user, handled by the MAB process. Indeed, setting one arm for every source policy is not sustainable for real-world systems since the stochastic MAB regret is linear in number of arms. The initial phase of MAB dialogue collection lasts $d \sim 100$ dialogues. This is why this paper proposes to create a set of limited size k of source representatives from a large user database. Two methods are proposed: one based on the cost function of k-medoids and the other one based on k-means. All rely on **PD-DISTANCE**, a novel policy-driven distance introduced by this paper:

$$d_{pd}\left(u, u'\right) = \sqrt{\sum_{s \in \Omega} 1 - \mathbb{1}\left(\pi_u(s), \pi_{u'}(s)\right)} \quad (5)$$

where u and u' are source users and π_u and $\pi_{u'}$ the policies trained with them. The state set Ω is obtained by a sampling over the states reached.

In the **KMEDOIDS** method, we propose to choose directly k representatives into the systems database. The cost function optimized by the k-medoids algorithm, denoted as J here, is used. Let $\mathscr{P}_k(\mathcal{U})$ denote the ensemble of k combinations of elements among $\mathcal{U}$, the set of all source users. If $U \in \mathscr{P}_k(\mathcal{U})$, and d is a distance, then the cost function is defined as:

$$J(U) = \sum_{u \in \mathcal{U}} \min_{u' \in U} d(u, u'). \quad (6)$$

Thus, the goal is to find the set P_{min} minimising KMEDOIDS. This paper uses PD-DISTANCE as the distance d. For convenience, instead of optimising over all $U \in \mathscr{P}_k(\mathcal{U})$, we sample uniformly on $\mathscr{P}_k(\mathcal{U})$ and keep the smallest cost value $J(U)$, but one could use better optimization methods to find the best fit according to KMEDOIDS (like a greedy approach).

In the **KMEANS** method, we cluster systems with the k-means algorithm using PD-DISTANCE as a distance. When implementing, a change must be operated so the k-means can keep using euclidean distance: one must design each vector v to cluster this way : $v(s, a) = 1$ if a has been chosen in s, 0 otherwise. Note that KMEDOIDS directly picks elements from the main set while k-means regroups elements around means of vectors potentially corresponding to non-existent systems. The KMEANS method must construct the k system representatives from the clusters. A representative is a new system learnt using Fitted-Q. The training batch is constructed by gathering N_{ts} transfer

samples (s, a, r', s') of each system of the corresponding cluster.

5 Experiments

In order to test the previous methods, experiences are ran on **the negotiation dialogue game** (**NDG**) (Laroche and Genevay, 2017). In this game, two players must agree on a time-slot for an appointment. For each player p, each time-slot τ is associated to a cost $c_{p,\tau} \in [0, 1]$. At each turn of the game, a player can refuse the other player's time-slot and propose another time-slot: REFPROP(τ), ask the other player to repeat: ASKREPEAT, terminate the game: ENDDIAL or accept the other player's slot: ACCEPT. The noise inherent to spoken dialogues (because of ASR errors) is simulated: when a player proposes a time-slot, there is ser probability that the time-slot proposed is corrupted where ser denotes the sentence error rate of this player. The speech recognition score srs of an utterance is then computed with the following formula:

$$srs = \frac{1}{1 + e^{-X}} \tag{7}$$

where $X \sim \mathcal{N}(x, 0.2)$, $x = x_\top$ if understood, $x = x_\bot$ otherwise. These parameters are relative to each player. The further apart the normal distribution centers are, the easier it will be for the system to know if it understood the right time-slot, given the score. At the end of the game, if there is an agreement (*i.e.* there is no misunderstanding in the slot τ agreed), the system v, receives a dialogue return $r_v = \omega_v - c_{v,\tau} + \alpha_v(\omega_u - c_{u,\tau})$, where u denotes the other player: the user (either real human or a user simulator). For each player p, $\omega_p \in \mathbb{R}$ is the utility of reaching an agreement, $\alpha_p \in \mathbb{R}$ his cooperation tendency and $\gamma_u \in [0, 1]$ his patience. If players, v and u, agreed on different time-slots, the following formula applies to compute v's score $r_v = -c_{v,\tau_v} + \alpha_v(-c_{u,\tau_u})$. In this context, players should better agreed on the same time-slot at the risk of getting a very bad score. The dialogue score is then $score_v = \gamma_u^{t_f} r_v$ where t_f is the size of the current dialogue. Thanks to the γ_u parameter, players are inclined to accept a time-slot in a limited time[1]. In the following, the number of available slots ($gamesize$) is set to 4 and the maximum number of utterances in a dialogue ($maxdialoguesize$) is set to 50 (once this maximum is reached, a zero score is given). α and ω are set to 1.

[1]Please note that γ_u is not the same as the γ in the MDP formulation.

	Merwan	*Nico*	*Will*	*Alex*
ACCEPT	7%	35%	24%	13%
ENDDIAL	0%	0%	0%	0%
ASKREPEAT	1%	14%	10%	6%
REFPROP(0)	88%	45%	60%	64%
REFPROP(1)	3%	5%	6%	15%
REFPROP(2)	0%	0%	1%	2%
REFPROP(3)	1%	0%	0%	0%
learn error	5.2%	5.2%	4.9%	6.8%

Table 1: Rounded actions distributions of humans and learn error of their kNN model.

Both KMEANS and KMEDOIDS methods for searching representatives will be tested. The objective is to show that these methods improve the dialogue quality compared to non adaptive methods. All the tests are done in the following context: a user (human-model user or handcrafted user) and a system play a negotiation dialogue game. A dialogue is defined as one episode of the game. Slot preferences for users and systems are determined randomly at the beginning of each dialogue. The collected target dialogues are used to train a policy for the new user and the baselines and the clustering methods are compared in their ability to enable fast user adaptation.

Before jumping to the results, next section presents the user ensemble design.

5.1 Users design

Experiments are split in two parts with different sets of (source and target) users: the first set is artificially handcrafted (handcrafted users), while the second one is trained on human-human trajectories (human-model users).

Handcrafted users: to expose the need of user adaptation, different types of handcrafted users are defined:

- The deterministic user (**DU**) proposes its slots in decreasing order (in term of its own costs). If a slot proposed by the other user fits in its $x\%$ better slots, it accepts, otherwise it refuses and proposes its next best slot. If the other user proposes twice the same slot (in other words, he insists), **DU** terminates the

dialogue. Once that **DU** proposed all its slots, it restarts with its best slots all over again.

- The random user (**RU**) accepts any slot with a probability of x, otherwise it refuses and proposes a random slot.
- The always-refprop-best user (**ARPBU**) always refuses other user's slot and proposes its best slot.
- The always-accept user (**AAU**) always accept the other user slot. If AAU begins the dialogue, it proposes its best slot.
- The stop-after-one-turn user (**SAOTU**) proposes a random slot then ends the dialogue regardless of the other user response.

Human-model users: in order to gather dialogues from human users, a multi-human version of the negotiation game has been created. Making the humans play together avoids too fast adaptation from the humans ((unlike human versus computer setup) and thus keep the experiments in a stationary environment.

The number of slots available has been set to 4 and all human users share the same parameters from the negotiation game which are $\gamma_u = 0.9$, $\omega = 1$, $ser = 0.3$, $c_\top = 1$, $c_\bot = -1$ and $\alpha = 1$. The game is then fully cooperative. Four humans : *Alex*, *Nico*, *Merwan* and *Will* played an average of 100 dialogues each. Using human trajectories, we design human-model users. State/action couples are extracted from these trajectories.

Human-model users can do the following actions: ACCEPT, ASKREPEAT and ENDDIAL. They can also REFPROP(i) to refuse the other user slot and propose their i^{th} best slot. The action REFPROP(0) then means that the human-model user refuses and proposes its best slot. We find the corresponding human-model users actions with the humans actions. Table 1 shows the empirical distribution on the human-model users actions space for each (real) human. Even if a human has not been subjected to the same dialogue trajectories, some behavioural differences clearly appear. *Merwan* tends to insist on his best slot while *Nico* seems more compliant. *Alex* is more versatile in the actions chosen.

Human-model users require an approximate representation, or projection, of the human state. Let $nbslots \in \mathbb{N}^+$, the number of available slots of the game, then the dialogue state representation is defined as a vector of the $2 + 3 * nbslots$ following attributes: the speech recognition score of the last received utterance, the costs of all slots sorted, the frequencies of all REFPROP(i) actions done by this user during the dialogue, the frequencies of all slot propositions done by the other user (ordered by cost for this user) during the dialogue and finally the cost of the last slot proposed by the other user.

Each human is modelled with a k-nearest-neighbour algorithm (kNN), with $k = 5$, fed with their corresponding data couples state/action. Table 1 also shows the training errors.

Finally, handcrafted and human-model users share the following parameter values: $ser = 0.3$, $c_\top = 1$, $c_\bot = 0$, $\omega = 1$, $\alpha = 1$ and $\gamma_u = 0.9$.

5.2 Systems design

Each system is trained with the least-square Fitted-Q algorithm. Their actions set is restricted to: ACCEPT, ASKREPEAT, ENDDIAL, and two REFPROP actions: REFPROPNEXTBEST to refuse the other user's slot and propose the next best slot after the last slot the system proposed (once all slots have been proposed, the system loops) and INSISTCURRENTBEST to propose his last proposed slot. The dialogue state tracker collects three attributes, the current iteration number of the dialogue ($nbUpdate$), the speech recognition score (srs) and the difference between the cost of the next slot the system can propose and the cost of the slot currently proposed by the user ($cost$). Fitted-Q's feature representation is then defined with 7 attributes for each action: $\phi(s, a) = (1, cost, srs, up, cost * srs, cost * up, srs * up)$ where $up = 1 - \frac{1}{nbUpdate}$. The learning is done in B batches of Fitted-Q (with $\gamma = 0.9$, $\delta = 0.001$ and $max_{it} = 200$). For each batch, a set of D dialogues is generated between the system and a user and then a new policy is computed with Fitted-Q fed with all the dialogues done so far. Policies are ϵ-greedy, ϵ annealing from $\epsilon = 0.25$ at the 1^{st} batch to $\epsilon = 0.01$ at the last batch. In between, $\epsilon(b) = \frac{1}{a_e * b + b_e}$ where $a_e = 19.2$, $b_e = -15.2$ and b the current batch index. ϵ is set to 0 during the test phase (in order to greedily exploit the current policy). In the human setup, $B = 6$ and $D = 500$. In the handcrafted setup, $B = 6$ and $D = 200$.

5.3 Cross comparisons

To show the importance of user adaptation, source systems are respectively trained versus users. Then,

u \ s	type	$c_\top$	$c_\perp$	x	vspu1	vspu2	vspu3	vspu4	vspu5	vspu6	vspu7
pu1	DU	1	-1	0.1	**0,62**	0,44	0,46	0,40	0,40	0,40	0,59
pu2	DU	5	-5	0.1	0,53	**0,82**	0,81	0,51	0,70	0,41	0,71
pu3	DU	5	-5	0.2	0,53	**0,81**	**0,81**	0,52	0,72	0,42	0,71
pu4	RU	5	-5	0.1	0,42	0,94	0,94	**1,00**	0,92	0,85	0,94
pu5	ARPBU	1	-1		0,84	0,98	1,00	1,11	**1,16**	1,13	1,05
pu6	AAU	1	-1		0,95	1,06	1,07	1,29	1,27	**1,30**	1,06
pu7	SAOTU	1	-1		0,43	0,26	0,27	0,10	0,18	0,03	**0,58**

Table 2: Handcrafted users characteristics and cross comparison between handcrafted users and systems. For $i \in [0, 7]$, vspu$_i$ is the system trained versus the user pu$_i$.

u \ s	*vsAlex*	*vsNico*	*vsWill*	*vsMerwan*
Alex	**1.077**	1.041	1.071	1.066
Nico	1.246	**1.251**	1.246	1.231
Will	1.123	1.109	**1.126**	1.117
Merwan	0.989	0.903	0.985	**0.998**

Table 3: Cross comparison between human-model users and systems

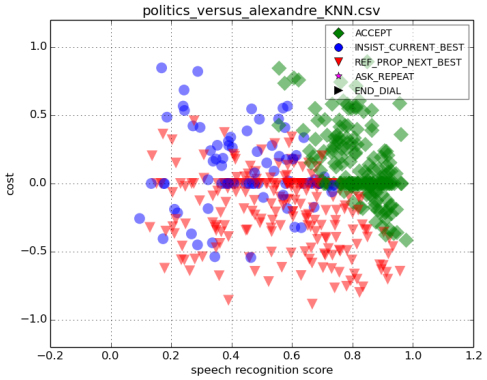
each system interacts versus all the users and we compare the results. The experiences are repeated for 10 runs. Dialogue testing size is set to 10^3 for each run. In the **handcrafted setup**, as in (Genevay and Laroche, 2016), handcrafted users are created. Parameters of these users are listed in Table 2. Also, cross comparisons between source users and systems are displayed. Results in bold show that each system trained versus a specific user is the best fit to dialogue with this user. One can see clear similarities between some of the results. This is where the representatives design method will operate by grouping all these similar policies.

In the **human setup**, test systems are trained against the human-model user. Results are shown in Table 3. Note that label *Will* means model of *Will* and not *Will* himself as well as *vsWill* means the system trained against *Will*'s model. Again, trained systems perform better than others against the user they learnt on. However, differences are not as clear as in the handcrafted setup. The reason is shown in Figure 2a, 2b and 2c where learnt policies are quite similar. Computed policies are tested on the states s_i from the set of $(s_i, a_i, r'_i, s'_i)_{i \in N}$ they learnt from. The $(srs, costs)$ projection explains better policy differences. One can see that *vsAlex* and *vsWill* are pretty similar as they insist often when the cost is negative, in contrary of other policies. On the other hand, *vsNico* tends to REFPROP instead of ACCEPT when the speech recognition score is high. It's pretty straightforward to remark that is because *Nico* has tendencies to ACCEPT more than others as we saw in Table 1. One can remark that even if statistics gather from human actions distribution shows significant differences (in Table 1), policies computed are not necessarily different (like *vsAlex* and *vsWill*).

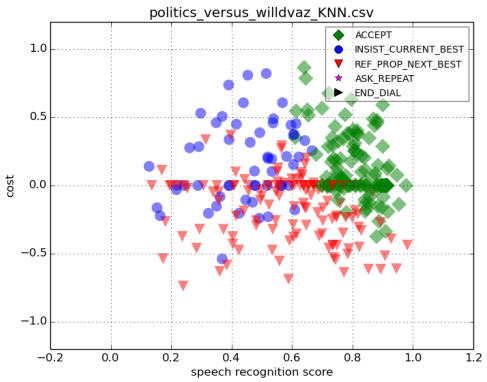
5.4 Adaptation results

Now specialised systems have been shown leading to better results, we test the full adaptation process with KMEDOIDS and KMEANS methods. As previously, tests are performed on both handcrafted and human-model users. But first, the database of source systems is constructed. 100 handcrafted source users and 100 human source user models are created. Those are designed by changing some parameters of the vanilla users. For example, a model from Alex is changed switching its speech error rate from 0.3 to 0.5. Parameters take random value between the following intervals: $c_\top \in [0, 5]$ with $c_\perp = -c_\top$, $ser \in [0, 0.5]$, $\alpha \in [0, 1]$, $x \in [0.1, 0.9]$ and $p \in [0.3, 0.9]$. It is useful for human setup because we do not have enough dialogue corpora to design 100 systems specialized versus 100 unique human-model users. The same method is applied to generate a large number of handcrafted users as well. For each user, a source policy is trained after 6 batches of 200 dialogues (for a total of 1200 dialogues). Each system is added to its respective database (human-model or handcrafted). We end up with 100 source trained policies with 100 handcrafted source users and 100 source trained policies with 100 human-model source users.

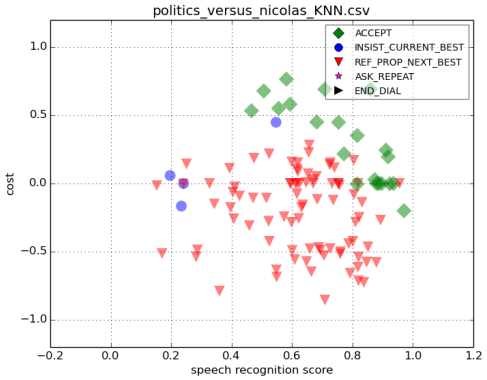
KMEANS and KMEDOIDS are tested for the com-



(a) *vsAlex policy's 2D projection.*



(b) *vsWill policy's 2D projection.*



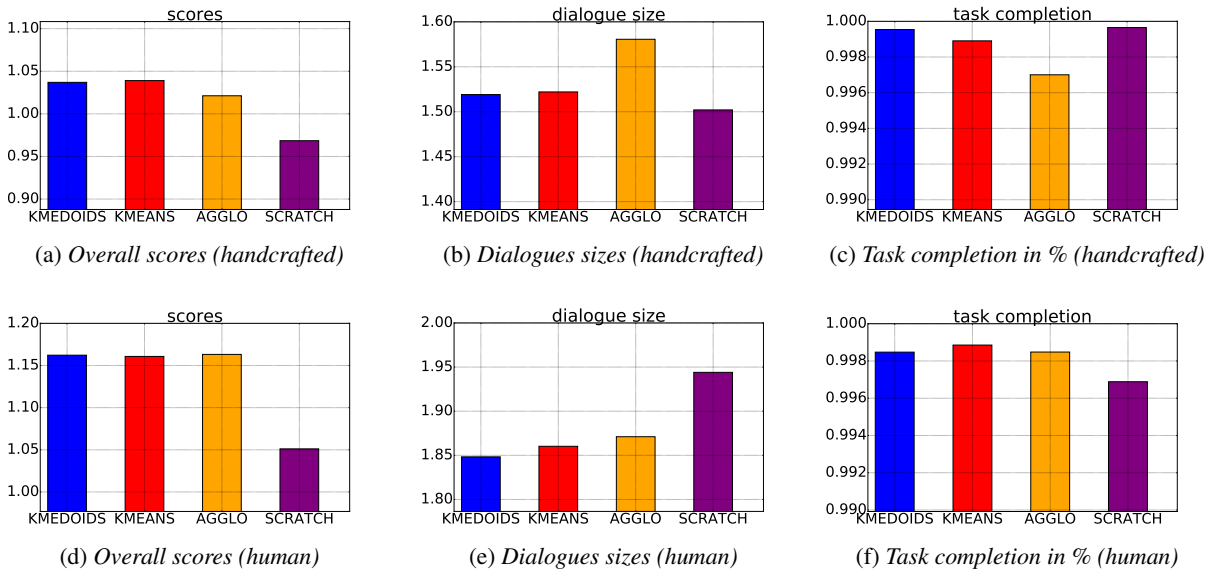
(c) *vsNico policy's 2D projection.*

Figure 2: Some projections of policies optimised versus human-model users.

plete adaptation process versus a base of 500 target users generated randomly (in the same way as users have been generated to create source systems). As discussed in Section 3, the adaptation process implies a bandit phase: 25 dialogues are done versus the target user then the mean score is saved to be plotted. Then all the samples (s, a, r', s') are retrieved from the source system winner of the bandit. The process actually transfers a maximum of 1200 dialogues. Transfer samples are submitted to a filtering using density-based selection with η parameter picked in the set $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ [2]. Then, a new policy is learnt with Fitted-Q fed with samples from the source system and samples from the bandit dialogues. To avoid divergence, a λ-regularization is applied to Fitted-Q with $\lambda = 1$. Once the policy learnt, 25 additional dialogues are sampled versus the target user. After this sampling, the mean score is saved to be plotted later. The process is repeated 6 times for a total of 25+6*25+1200 dialogues maximum for the learning and 25+6*25+25 dialogues for the evaluation. All systems, sources and targets, share the following parameters; $\omega = 1$, $\gamma_u = 0.9$, $ser = 0.0$, $c_\top = 5$, $c_\bot = -5$, $\alpha = 1$ but differ in their policy. All systems are learnt with Fitted-Q using the following parameters: $\delta = 0.001$, $maxit = 200$ and $\gamma = 0.9$. They all follow an ϵ-greedy policy with ϵ defined as in Section 5.2.

In order to compare the previous methods, we introduce two naive ways for user adaptation, AGGLO and SCRATCH. The first one learns a unique system to represent the whole systems database and the second one adopts a random policy during the bandit phase then follows an ϵ-greedy policy like other methods but without any transfer. Before running experiment, pre-processing is done for some the methods: for AGGLO, 1200 dialogues are gathered among all source systems in the database. That means 12 dialogues are collected randomly from the dialogue set of each of the 100 source systems. A policy is learnt with one batch of Fitted-Q with $\delta = 10^{-6}$, γ=0.9 and max_{it}=200. This policy is used to create one unique system representative for all the database. For KMEANS, PD-DISTANCE vector representations of each system in the database are created by sampling over 20000 states (picked from source systems). Then these systems are clustered with k=5 using k-means with

[2] We kept only $\eta = 0.3$ as results are pretty similar with any η



(a) *Overall scores (handcrafted)* (b) *Dialogues sizes (handcrafted)* (c) *Task completion in % (handcrafted)*

(d) *Overall scores (human)* (e) *Dialogues sizes (human)* (f) *Task completion in % (human)*

Figure 3: Dialogue quality in the handcrafted and human setup.

euclidean distance. For each cluster the previous AGGLO method is applied in order to create a cluster representative. Finally, for KMEDOIDS, a random sampling of five-element sets is ran. The J value of each set is computed and the one who minimizes this value is kept. Finally, 10 different sets of AGGLO, KMEANS and KMEDOIDS are created and tested. Results are shown in Figure 3. In the handcrafted setup, the overall dialogue quality of the proposed methods is significantly better than AGGLO and SCRATCH baselines. Indeed, dialogues are shorter [3], final score is higher and the task is more often completed. On the other side, scores and task completions are similar in the human setup. Still, the size of the dialogues is improved by KMEANS and KMEDOIDS offering a better dialogue experience and thus users keep using the dialogue system.

6 Related work

To our knowledge, just one paper treats the subject of searching system representatives among a systems database: (Mahmud et al., 2013) has a similar adaptation process as the one presented in this paper. It isn't applied to dialogue systems specifically. In order to choose good representatives from the policies/MDP/systems database, clustering is done using the following distance

$$d_V(M_i, M_j) = max\{V_i^{\pi_i^*} - V_i^{\pi_j^*}, V_j^{\pi_j^*} - V_j^{\pi_i^*}\}$$

given two MDP M_i and M_j, where V_k^π is the score of the policy π when executed on MDP M_k and π_k^* refers to the optimal policy for MDP M_k. Thus, to compute all the systems distances two by two, one needs to sample dialogues between the source users and all the source systems of the database. In a real life dialogue applications with humans, it is not possible to do such thing unless one creates a model of each source user.

7 Conclusion

In this paper, user adaptation has been proved to improve dialogue systems performances when users adopt different behaviours. The paper shows that indeed, each human adopts a different way to play the NDG although the shade is subtle. So, a system learnt versus a particular user is more efficient than other systems for this user, in the handcrafted user setup as in the human-model user setup.

User adaptation requires selecting source systems to transfer knowledge. This paper proposed 2 methods: KMEANS and KMEDOIDS combined to a novel distance PD-DISTANCE to select representative source systems, from a large database, which are used for transferring dialogue samples. These methods outperformed generic policies in the handcrafted setup and improved dialogue quality when facing models learnt on human-human data.

[3] SCRATCH's dialogue size can be shorter because it use random policy on cold start and then ends the dialogue more often.

References

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine Learning* .

Merwan Barlier, Julien Perolat, Romain Laroche, and Olivier Pietquin. 2015. Human-machine dialogue as a stochastic game. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue (Sigdial)*.

Richard Bellman. 1956. Dynamic programming and lagrange multipliers. *Proceedings of the National Academy of Sciences of the United States of America* 42(10):767.

Nigo Casanueva, Thomas Hain, Heidi Christensen, Ricard Marxer, and Phil Green. 2015. Knowledge transfer between speakers for personalised dialogue management pages 12–21.

Senthilkumar Chandramohan, Matthieu Geist, Fabrice Lefèvre, and Olivier Pietquin. 2012. Clustering behaviors of spoken dialogue systems users. *Proceedings of the 37th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* .

Senthilkumar Chandramohan, Matthieu Geist, and Olivier Pietquin. 2010. Optimizing Spoken Dialogue Management with Fitted Value Iteration. In *Proceedings of the 10th Annual Conference of the International Speech Communication Association (Interspeech)*.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. 2005. Tree-Based Batch Mode Reinforcement Learning. *Journal of Machine Learning Research* 6:503–556.

Milica Gašić, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. 2013. Pomdp-based dialogue manager adaptation to extended domains. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (Sigdial)*.

Aude Genevay and Romain Laroche. 2016. Transfer learning for user adaptation in spoken dialogue systems. In *Proceedings of the 15th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*. International Foundation for Autonomous Agents and Multiagent Systems.

Kallirroi Georgila and David R Traum. 2011. Reinforcement learning of argumentation dialogue policies in negotiation. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (Interspeech)*. pages 2073–2076.

Srinivasan Janarthanam and Oliver Lemon. 2010. Adaptive Referring Expression Generation in Spoken Dialogue Systems: Evaluation with Real Users pages 124–131.

Romain Laroche and Aude Genevay. 2017. The negotiation dialogue game. In *Dialogues with Social Robots*, Springer, pages 403–410.

Alessandro Lazaric. 2012. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*, Springer, pages 143–173.

Alessandro Lazaric, Marcello Restelli, and Andrea Bonarini. 2008. Transfer of samples in batch reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*. ACM, pages 544–551.

Esther Levin and Roberto Pieraccini. 1997. A stochastic model of computer-human interaction for learning dialogue strategies. In *Proceedings of the 5th European Conference on Speech Communication and Technology (Eurospeech)*.

Lihong Li, Jason D. Williams, and Suhrid Balakrishnan. 2009. Reinforcement learning for dialog management using least-squares policy iteration and fast feature selection. In *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*. pages 2475–2478.

M. M. Hassan Mahmud, Majd Hawasly, Benjamin Rosman, and Subramanian Ramamoorthy. 2013. Clustering markov decision processes for continual transfer. *CoRR* abs/1311.3959.

Amir Massoud, Farahmand Mohammad Ghavamzadeh, Csaba Szepesvári, and Shie Mannor. 2009. Regularized Fitted Q-iteration for Planning in Continuous-Space Markovian Decision Problems .

Olivier Pietquin, Matthieu Geist, Senthilkumar Chandramohan, and Hervé Frezza-Buet. 2011. Sample-efficient batch reinforcement learning for dialogue management optimization. *ACM Transactions on Speech and Language Processing (TSLP)* 7(3):7.

Fariba Sadri, Francesca Toni, and Paolo Torroni. 2001. Dialogues for negotiation: agent varieties and dialogue sequences. In *ATAL*. Springer, pages 405–421.

Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Matthew E Taylor and Peter Stone. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10:1633–1685.

Andrei Nikolaevich Tikhonov. 1963. Regularization of incorrectly posed problems. Soviet Mathematics Doklady.

Stefan Ultes, Matthias Kraus, Alexander Schmitt, and Wolfgang Minker. 2015. Quality-adaptive Spoken Dialogue Initiative Selection And Implications On Reward Modelling pages 374–383.