Enhancing Designer Knowledge to Dialogue Management: A Comparison between Supervised and Reinforcement Learning Approaches

Bruno Eidi Nishimoto[1,2], **Rogers Silva Cristo**[2], **Alex Fernandes Mansano**[2]
Eduardo Raul Hruschka[1], **Vinicius Fernandes Caridá**[2], **Anna Helena Reali Costa**[1]

[1]Centro de Ciências de Dados, Universidade de São Paulo (USP) - São Paulo - SP - Brazil

[2]Itaú Unibanco - São Paulo - SP - Brazil

{bruno.nishimoto,hruschka,anna.reali}@usp.br

{rogers.cristo,alex.mansano,vinicius.carida}@itau-unibanco.com.br

Abstract. *Task-oriented dialogue systems are complex natural language applications employed in various fields such as health care, sales assistance, and digital customer servicing. Although the literature suggests several approaches to managing this type of dialogue system, only a few of them compares the performance of different techniques. From this perspective, in this paper we present a comparison between supervised learning, using the transformer architecture, and reinforcement learning using two flavors of Deep Q-Learning (DQN) algorithms. Our experiments use the MultiWOZ dataset and a real-world digital customer service dataset, from which we show that integrating expert pre-defined rules with DQN allows outperforming supervised approaches. Additionally, we also propose a method to make better usage of the designer knowledge by improving how interactions collected in warm-up are used in training phase. Our results indicate a reduction in training time by preserving the designer's knowledge, expressed as pre-defined rules in memory during the initial steps of the DQN training procedure.*

1. Introduction

An important class of dialogue systems (DS) refers to goal or task-oriented DS [Zhang et al. 2020]. These systems can interact with users by using natural language and help them reaching a specific goal (or to completing a task), ideally minimizing the number of turns. For instance, most virtual assistants such as Amazon's Alexa[1], Apple's Siri[2] and Google Assistant[3] have a task-oriented chatbot embedded in them. The relevance of DS is clearly supported by the wide range of applications that can help users to complete common daily tasks more practically and easily. These tasks may range from buying a movie ticket to booking a flight and giving tourist information.

There are two types of architecture for modeling a DS: end-to-end and pipeline. In the end-to-end architecture, the dialogue is traditionally formulated as a sequence-to-sequence problem as illustrated in Figure 1. Encoder-decoder with supervised learning

[1]https://alexa.amazon.com
[2]https://www.apple.com/siri/
[3]https://assistant.google.com/

techniques is frequently applied to train such models [Vlasov et al. 2019]. On the other hand, there is the pipeline DS architecture (Figure 2). It is composed of three components, namely: Natural Language Understanding (NLU), Dialogue Management (DM), and Natural Language Generation (NLG) [Chen et al. 2017]. The NLU component extracts the most relevant information from the user's utterance and translates it to the dialogue act. The DM component, in its turn, controls the dialogue flow, i.e., it chooses the best response to give to the user. Finally, the NLG component transforms the system's response into natural language. The DM component is central to the DS as it is directly responsible for the dialog flow control [Dai et al. 2020]. It comprises two modules, the Dialogue State Tracking (DST), which keeps track of the conversation state, and the policy (POL) that chooses the system response. Additionally, the conversation quality is most influenced by the replies determined by the DM for each user interaction. For this reason, we focus on the DM component in this paper when talking about the pipeline architecture.

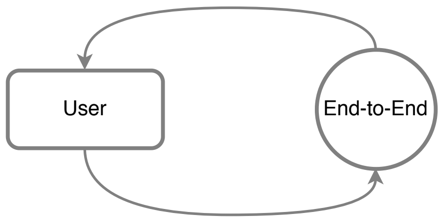

Figure 1. Illustration of end-to-end architecture.

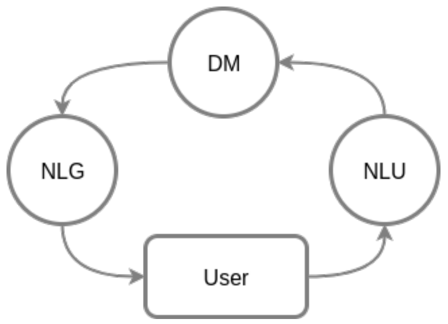

Figure 2. Illustration of pipeline architecture.

Many works focus in the pipeline architecture, specially in the policy component of the DM module. In brief, they employ rule-based DM, supervised learning [Vlasov et al. 2019, Hosseini-Asl et al. 2020]; and reinforcement learning [Saha et al. 2020, Wang et al. 2020, Gordon-Hall et al. 2020].

Rule-based systems, such as the well-known ELIZA[4] [Weizenbaum 1966], were initially introduced as a straightforward solution in which DM determines its response based on pre-defined and handcrafted rules. Although this approach is easy to implement and performs appropriately on simple domains [Yan et al. 2017, Montenegro et al. 2019, Dai et al. 2020] it lacks versatility and, accordingly, is hard to scale for other domains without rewriting most of the rules. Furthermore, the number of rules can increase exponentially as the domain becomes more complex, making it difficult to specify all rules manually.

In order to avoid these limitations, many researchers have been concentrating on applying machine learning techniques to generalize DM tasks. With this approach, the DM's input and output are data structures known as dialogue acts instead of natural language utterances. One possible approach is to apply supervised learning (SL) directly on the DM module [Vlasov et al. 2018, Vlasov et al. 2019]. As pointed out in other fields of application, the main drawback with this approach is the massive amount of labeled data

[4]ELIZA was created in 1966 by Joseph Weizenbaum. Its functionality is to copy the behavior of a psychologist, and the response are taken from a set of template rules.

needed to yield a trained model. Even with some pre-trained models, there is a need of task-specific labeled data to train the model. And dialogue systems suffers to collect good labeled data for a specific domain. Another approach is reinforcement learning (RL) in which the agent learns to drive a good conversation by interacting and receiving rewards for each action it takes [Sutton and Barto 2018]. Applying RL, thus, can promote a less labeled data-dependent system. Furthermore, the conversation can also be treated as a sequential decision-making problem, making the use of RL even more reliable.

A relevant task for dialogue management is how to improve the pre-training or warm-up phase so that we can leverage expert knowledge during training. Also, most works aim to develop new techniques of either SL or RL approaches, but there are not many comparisons between the two approaches. [Vlasov et al. 2019, Takanobu et al. 2020] makes a comparison between end-to-end architecture and the pipeline architecture either with supervised or reinforcement learning. But it lack a direct comparison between supervised and reinforcement learning both in the pipeline architecture This fact hinders the decision of which method is better than the other when developing dialogue systems. In addition, the literature focuses on toy dataset that are less complex than real-world data. In this context, results from the literature tend to be somewhat optimistic.

This paper presents three contributions addressing the problems mentioned above:

- We enhance the warm-up phase for reinforcement learning algorithm to better manage the knowledge given by the designer by keeping it longer and improve the initial learning in the training;
- We perform a comparison between the supervised and reinforcement learning approaches applied to DM in the pipeline architecture, demonstrating that the latter presents better results in DS as it is able to perform user's tasks more often;
- We also compare these techniques in the digital customer service domain by constructing a user simulator from real-world interactions, showing that reinforcement learning also presents better results, yielding a similar behavior in both toy datasets and real-world applications.

2. Related Works

As mentioned earlier, researchers have used supervised learning and reinforcement learning to model a policy in the in the pipeline architecture (Figure 2). In the SL approach [Griol et al. 2008] for pipeline architecture (SL$_{pip}$) used a Multi-Layer Perceptron (MLP) network to model the dialogue management policy. [Bocklisch et al. 2017] used similar approaches to [Williams et al. 2017], where the authors proposed the Hybrid Code Networks (HCN). This model contains a Recurrent Neural Network (RNN) to compute the hidden state and a dense layer with a softmax activation function to choose the next action. [Vlasov et al. 2018] used a embedding layer as the RNN input to learn vector representation for dialogue states and system actions in a supervised setting. More recent publications use the transformer architecture for this task. [Vlasov et al. 2019] proposed the use of a transformer layer to replace the RNN and build a retrieval model. Likewise, [Ham et al. 2020, Hosseini-Asl et al. 2020] used GPT-2, a pre-trained transformers model to build generative model in the pipeline architecture.

Another alternative is to use the RL approach in the pipeline architecture, in which the dialogue system is considered a sequential decision problem and formulated as a Markov Decision Process (MDP) framework. [Li et al. 2017] and [Zhao and Eskenazi 2016] employed the Deep Q-Leaning (DQN) algorithm in order to model a robust DM. [Nishimoto and Reali Costa 2019] extended the first work by showing that a good balance in exploration and exploitation during training can significantly improve the performance. Some other recent works also used the classical DQN algorithm to train the policy [Gordon-Hall et al. 2020, Wang et al. 2020], showing that despite simple, this algorithm can provide good results [Mo et al. 2018] and [Weisz et al. 2018] tried out other RL algorithms to model the DM, such as SARSA and actor-critic, respectively. Finally, [Saha et al. 2020] proposed a hierarchical deep reinforcement learning approach to deal with more complex dialogue systems and [Takanobu et al. 2019] proposed a method to learn the reward and optimize the policy jointly. As the learning of a policy from scratch in reinforcement learning is data and time consuming, many works focus on techniques using expert knowledge to enhance and accelerate the training of the agent. [Li et al. 2017] and [Nishimoto and Reali Costa 2019] take advantage of a set of pre-defined rules employed in a warm-up phase before training. Besides being a straightforward way to enhance expert knowledge, i.e, using the pre-defined rules to generate good interactions so the agent can use them to learn, this is still effective. Another alternative is to execute a pre-training, such as imitation learning or even supervised learning [Su et al. 2016]. However, it is more complicated and needs a lot of labeled data collected from experts. [Gordon-Hall et al. 2020] proposed the Deep Q-learning from Demonstrations (DQfD), which uses expert demonstrators in a weakly supervised fashion.

Despite the numerous proposals for modeling a good DM, just a few work in the literature compares the two paradigms in terms of their performances. [Vlasov et al. 2019] compared end-to-end (SL_{e2e}) and modular approaches for SL (SL_{pip}) and stated that the second one achieves better results. [Takanobu et al. 2020] compared different configurations of dialogue systems using RL, including GDPL (for RL), and DAMD (for SL_{e2e}) and concluded that modular approaches with RL also outperforms end-to-end ones. In summary, no work made a comparison between SL_{pip} and RL, both in the pipeline architecture.

3. Proposal

In this section we show the algorithms implemented for both supervised and reinforcement learning techniques used in this work.

3.1. Supervised Learning in Dialogue Systems

The transformer architecture has become quite relevant in recent years for handling sequential data, reaching state-of-the-art in many tasks. For this reason and due to the sequential nature of dialogue systems, we implemented a DM using the transformer model for our comparisons. The implementation is based on the work from [Vlasov et al. 2019] and Figure 3 shows a schematic representation of the DM architecture using the transformer architecture.

The input sequence of the transformer architecture is the sequence of the dialogue turns of a conversation. At first, DM encodes the user input, the system action and the

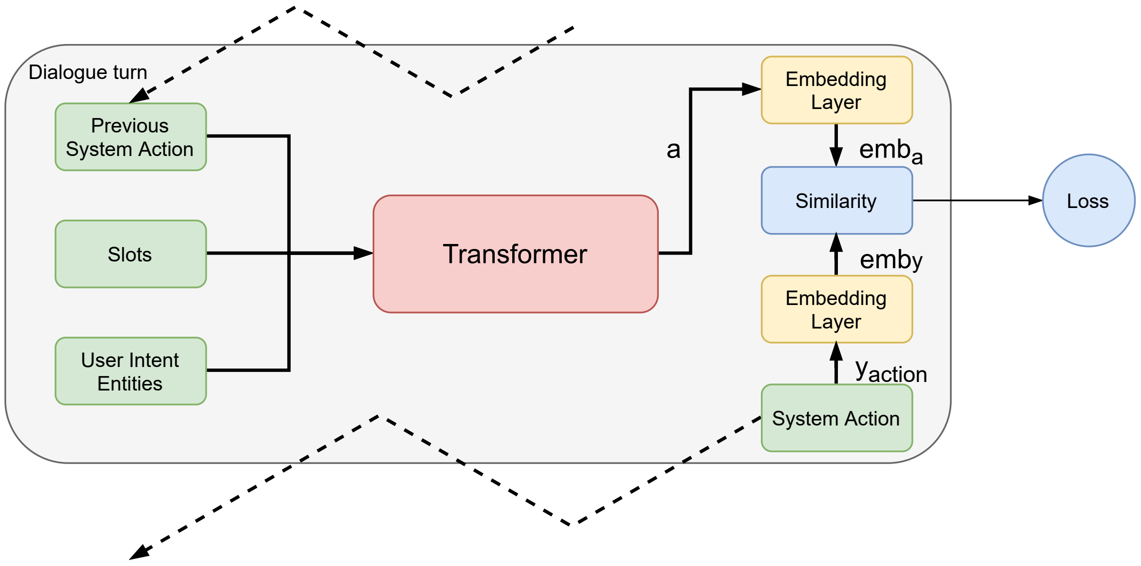

Figure 3. Schematic representation of the transformer dialogue management. Inspired in figure from [Vlasov et al. 2019]

slots just as showed before on Section 3 resulting in the dialogue state which is the input of the transformer policy. Therefore, the transformer model learns to map the dialogue state into the system action. The attention mechanism embedded in the transformer allows it to look at previous dialogue turns dynamically and use these information to select the best response, i.e., at each turn the transformer attention mechanism computes the relevance of the previous turns that will help the system make its decision. This relevance is what the attention mechanism will learn from data during the training. The transformer output a and the system action y_{action} (which is the label) are transformed into an embedding through the embedding layer $emb_a = E(a)$, and $emb_y = E(y_{action})$, where $emb \in \mathbb{R}^d$, with $d = 20$ being the embedding dimensions and E represents the embedding layer.

The loss function is based on the similarity between the transformer output embedding and all the system actions $S = emb_a^T emb_y$. Precisely, the loss function for one dialogue is:

$$L_{\text{dialogue}} = -\left\langle S^+ - \log\left(e^{S^+} + \sum_{\Omega^-} e^{S^-}\right)\right\rangle, \qquad (1)$$

where $S^+ = emb_a^T emb_y^+$, and emb_y^+ is the embedding for the target action y_{action}^+ (true label). $S^- = emb_a^T emb_y^-$, where emb_y^- is the embedding for all other actions y_{action}^- (negative samples) which are elements of the set Ω^-. The idea of this loss is to increase the similarity between the transformer output and the target, and decrease the similarity between the output and all other actions. Finally the global loss is computed as an average of the loss of all dialogues in the train dataset.

The feed forward layers inside the transformer contain one layer with 128 neurons. The multi-head attention comprises $h = 4$ heads and the dimension size of embedding vectors is $d = 20$. The batch size increases linearly from 8 to 32 for each epoch. So during the training, we sample the batch size of dialogues from the training dataset and for each dialogue we compute the loss, and then the global loss to optimize the transformer with the Adam Optimizer.

During the inference time, the system chooses the action which is the most similar to the transformer output as its response.

3.2. Reinforcement Learning in Dialogue Systems

The DM implementation with reinforcement learning follows [Nishimoto and Reali Costa 2019], where the authors used the DQN algorithm [Mnih et al. 2015] with a softmax distribution to balance exploration and exploitation during training. In order to compare the results to a baseline, we adopt the same parameters' values from [Nishimoto and Reali Costa 2019] work. Although its a simple algorithm, there are still many works that use DQN (or some variant) to train the DM [Li et al. 2017, Nishimoto and Reali Costa 2019, Gordon-Hall et al. 2020, Wang et al. 2020]. For this reason we used the DQN algorithm and focused in improving the warm-up phase. Moreover, DQN is an algorithm that is compatible with our problem because we have discrete actions and a deterministic policy which ensure that it does not take any unexpected action during a dialogue.

Figure 4 summarizes the architecture used in our implementation. It contains the training and target networks. The environment is represented by the user, which samples the user dialog act u_a. The user action u_a passes through the DST module that returns the dialogue state s. The agent acts either choosing the action a based on a set of rules predefined by the system designer – if it is in the warm-up phase – or following the policy given by the training action-value function q – if it is in the learning phase. Then the agent receives the next user action, a reward r, and the next dialogue state s'. The reward function is defined as giving as small penalty of -1 at each turn, a great reward $R_{success}$ if the user task is completed and a great penalty $-R_{fail}$ otherwise.

All the experiences (s, a, r, s') are stored in the replay memory buffer $\mathcal{M}$. These experiences are used to update the parameters θ of the training network by minimizing the loss function $\mathcal{L}(\theta) = \mathbb{E}\left[(r + \gamma \max_{a'} \hat{q}(s', a'; \theta^-) - q(s, a; \theta))^2\right]$, where $\hat{q}(.; \theta^-)$ represents the target network and $q(.; \theta)$ the training network.

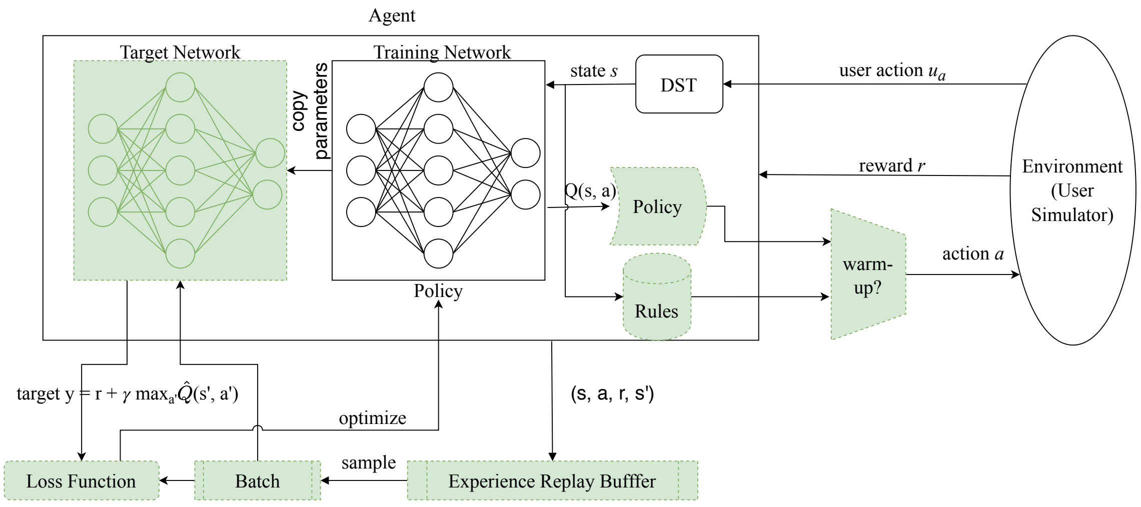

Figure 4. Architecture of the DQN algorithm.

Similar to other fields such as robotics, learning from *tabula rasa* by cooperating with human experts would be unfeasible due to the large number of training interactions required until the agent reaches an acceptable policy. In order to avoid this issue, we adopted a user simulator to replace humans during training. The user simulator follows an agenda-based approach [Schatzmann and Young 2009]. It first samples a rule from a rule list and then starts a conversation with the dialogue system using a rule-based policy. These rules are domain-specific, so for each domain, there is one user simulator with different rules. However, in general, at each turn, the user simulator checks the missing slots to complete the task and informs one of them or requests some information it needs.

The warm-up process is conducted in order to improve agent learning with the help of pre-defined rules. It occurs in a stage before the training. During this phase, the agent acts by following a collection of expert-defined rules. It fills the experience replay memory with these interactions that will prevent the agent to start training without any "experience". Since it has a limited size, older experiences are replaced when it is full. The central strategy in [Li et al. 2017] and [Nishimoto and Reali Costa 2019] resides in the flush of the experience replay when the agent achieves a specific success rate (30%) and then, during the training process, flush again every time the training network outperforms its previous policy. [Li et al. 2017] argue that the agent can learn better if it uses data only from the current best policy than from experiences from older policies.

In this work, we propose to keep experience replay instead of flushing it. The intuition behind it is that the experience replay is filled with some rule-based actions which may guide the agent's learning during the early steps of the training phase, mainly due to the pre-defined rules inherited from the warm-up process. If we do not flush this memory, the agent can better use these expert-defined rules and learn a good policy faster than otherwise. Flushing the memory when the agent reaches a certain success rate threshold can lead the algorithm not to use appropriately the knowledge summed up in these rules.

4. Experiments and Results

We performed three types of experiments. First, we compared two RL architectures, one applying the memory flush and another not flushing the memory replay during the training phase. The second experiment compared the agent's performance using reinforcement learning and supervised learning. Finally, the last experiment was aimed to test all the approaches mentioned above in a more complex domain extract from real-world conversations, also comparing the reinforcement learning and supervised learning approaches. In the following subsections, we specify the domains used for the experiment and show the results.

4.1. Domains

The experiments were executed in two domains: MultiWOZ [Budzianowski et al. 2018, Eric et al. 2019] and Itaú bank virtual assistant domain. The major difference between these two domains is that the data of the first one is publicly available [5] so that experiments are reproducible. In the second domain, the real-world data is not publicly available due to bank's privacy issues, but they are beneficial to illustrate the relative performances of the assessed methods. Furthermore, all data for the bank's domain follow the GDPL

[5]`https://github.com/budzianowski/multiwoz`

(General Data Protection Law), in which sensible data were not employed and client's identification was anonymised.

In the MultiWOZ domain, the agent's task span across seven domains: restaurant, hotel, attraction, taxi, train, hospital and police. So the agent needs to accomplish the user goal in any of these domain. For example, the user may want to go to a expensive Japanese restaurant at 08:00pm and also book a five star hotel, in the north area at 10:00pm. In the Itapu bank virtual assistant domain, the agent needs to solve some clients' interactions related to the bank. For example, in this domain, a rule is composed of the intent `consult` and the slots `credit card` and `debt` then the corresponding action would be related to consulting the credit card's debt. The interactions are specifically related to non-account holders in this work, such as problems with credit cards and debt and doubts about loans.

4.2. Experiments

For our experiments, we adopt accuracy as our evaluation metric, i.e., the success rate or the rate of dialogues for which models choose the correct action. For the comparison between supervised learning and reinforcement learning we also look at the precision, recall and f1 scores. These metrics are related to the ability of the agent to fulfill the slots of the user goal, i.e., to inform correct slots that are present in the user goal.

Following widely adopted experimental standards, we report results on training and also on testing datasets, which allow us to estimate the generalization capabilities of the assessed approaches. More precisely, after every epoch during training, we test the current policy on a testing dataset to observe the learning behavior on unseen data. All experiments are executed five times for 300 epochs, and the results are presented with a 95% confidence interval.

4.2.1. Flush vs No Flush

Figures 5 and 6 show the learning curve of the success rate during training and testing for the MultiWOZ dataset. We compare our proposed approach, which involves not flushing the replay memory, with the previous works [Li et al. 2017, Nishimoto and Reali Costa 2019] that flush the replay memory in the MultiWOZ dataset. Table 1 shows the hyperparameter used in the DQN algorithm. Although both procedures achieve comparable accuracy during training, our approach achieves better results in while testing. It shows that not flushing the memory, that is, using more the expert knowledge, specially in early stage of training, makes the agent to generalize better. Moreover, it also causes the agent to learn faster and achieves the asymptotic performance before the flush method. We only evaluated the results on the MultiWOZ domain for this specific experiment as there already was an established baseline to compare. Beyond that, the idea with the digital assistant domain is particularly to verify that the behavior (RL better than SL) also occurs in a more complex real-world application.

While our proposal reaches an average success rate greater than 0.8 at epoch 115 on the training dataset, the flush method only reaches such a result at epoch 130. We can view this improvement in the learning speed as better usage of the designer knowledge. Indeed, if we flush the memory replay after some threshold, all the rule actions took in

Table 1. Hyperparameters for RL approach

Parameters	Value
α	10^{-3}
γ	0.9
batch size	16
τ	$2.0 - 0.5$
hidden layer size	164

the warm-up phase are lost. Consequently, not flushing the memory keeps the designer knowledge in memory replay as long as possible and makes better use of it.

Therefore, we can state that not flushing the replay memory improves the agent's knowledge generalization regarding the warm-up rules, thus leading the agent to learn an appropriate policy faster — see Figure 6.

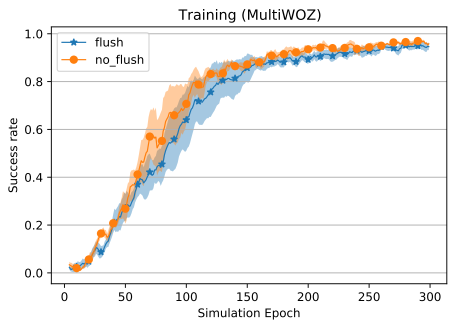

Figure 5. Comparison between the flush and no flush strategies in the training dataset. Metrics with 95% confidence interval.

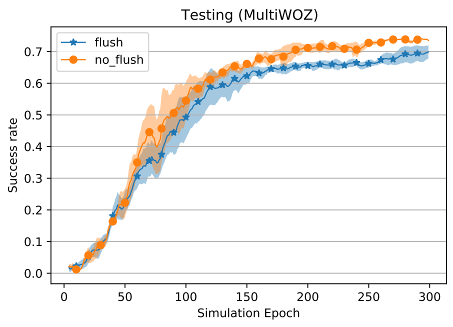

Figure 6. omparison between the flush and no flush strategies in the testing dataset. Metrics with 95% confidence interval.

4.2.2. Supervised Learning vs Reinforcement Learning

The results of the comparison between the supervised and reinforcement learning approaches for both the MultiWOZ dataset and the virtual assistant domain are shown in Tables 2 and 3, respectively. They show the results with 95% confidence interval in the testing phase. We can see that although they present similar performance in the accuracy, the reinforcement learning approach shows better results on the other metrics. It means that the agent trained with reinforcement learning can learn better on how to fulfill the user goal slots during the dialogue. The intuition behind this is that the sequential nature of reinforcement learning algorithms fits well in the dialogue management problems. Even though the transformer networks works well with sequential data, it does not leverage the dynamic interactions that we have in reinforcement learning.

We can also see that this behaviour (RL with better performance than SL) keeps in the virtual assistant domain. It was expected a decreasing in the performance compared to the MultiWOZ domain because of the complexity of domain (greater number of slots, actions and intents) and the difficulty of collecting good data for training.

Table 2. Metrics with 95% confidence interval of MultiWOZ in the test dataset.

MultiWOZ	F1	Precision	Recall	Accuracy
Transformer	0.61 ± 0.02	0.63 ± 0.01	0.61 ± 0.04	0.70 ± 0.01
DQN	0.83 ± 0.04	0.78 ± 0.06	0.89 ± 0.08	0.71 ± 0.04

Table 3. Metrics with 95% confidence interval of virtual assistant in the test dataset.

Virtual Assistant	F1	Precision	Recall	Accuracy
Transformer	0.57 ± 0.01	0.57 ± 0.01	0.59 ± 0.04	0.73 ± 0.01
DQN	0.79 ± 0.04	0.78 ± 0.06	0.78 ± 0.03	0.78 ± 0.04

5. Conclusions

Our contribution is two-fold. First, we showed that not flushing the memory during the training phase improves the agent's performance, as it starts the learning procedure with pre-loaded expert knowledge. It happens because the agent can generalize faster by using the knowledge contained in the pre-defined rules, as they stay available in the experience replay memory for a longer time. Another reason for this is that the experiences collected with the pre-defined rules are 'good'. If we have 'bad' experiences it would not be very effective. Second, we also showed that reinforcement learning techniques could outperform supervised learning ones. This result is somehow expected as goal-oriented dialogue systems, being domain-specific, typically lack labeled data for training. Furthermore, the nature of the problem fits well in the reinforcement learning framework, i.e., dialogue systems easily fit in a sequential decision problem. However there is the classic trade-off, in the reinforcement learning approach we need a good simulator while in the supervised learning approach we need a lot of labeled data.

Finally, we shall emphasize that our results were obtained not only by using simulated data, which are abundant in the literature, such as those from the MultiWOZ domain, but also domains extracted from real-world interactions.

Acknowledgements

The authors acknowledge the support of CNPq under grant 310085/2020-9. This research was partially being carried out with the support of Itaú Unibanco, through the scholarship program of *Programa de Bolsas Itaú* (PBI), and it is also financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Finance Code 001, Brazil.

Conflicts of Interest

Any opinions, findings, and conclusions expressed in this manuscript are those of the authors and do not necessarily reflect the views, official policy or position of Itaú Unibanco.

References

Bocklisch, T., Faulkner, J., Pawlowski, N., and Nichol, A. (2017). Rasa: Open source language understanding and dialogue management. *ArXiv*, abs/1712.05181.

Budzianowski, P., Wen, T.-H., Tseng, B.-H., Casanueva, I., Ultes, S., Ramadan, O., and Gašić, M. (2018). MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Chen, H., Liu, X., Yin, D., and Tang, J. (2017). A Survey on Dialogue Systems: Recent Advances and New Frontiers. *SIGKDD Explor. Newsl.*, 19(2):25–35.

Dai, Y., Yu, H., Jiang, Y., Tang, C., Li, Y., and Sun, J. (2020). A Survey on Dialog Management: Recent Advances and Challenges. *CoRR*, abs/2005.02233.

Eric, M., Goel, R., Paul, S., Sethi, A., Agarwal, S., Gao, S., and Hakkani-Tür, D. (2019). MultiWOZ 2.1: Multi-Domain Dialogue State Corrections and State Tracking Baselines. *CoRR*, abs/1907.01669.

Gordon-Hall, G., Gorinski, P. J., and Cohen, S. B. (2020). Learning Dialog Policies from Weak Demonstrations. In *ACL*, pages 1394–1405.

Griol, D., Hurtado, L. F., Segarra, E., and Sanchis, E. (2008). A statistical approach to spoken dialog systems design and evaluation. *Speech Communication*, 50(8):666–682.

Ham, D., Lee, J.-G., Jang, Y., and Kim, K.-E. (2020). End-to-End Neural Pipeline for Goal-Oriented Dialogue Systems using GPT-2. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 583–592, Online. Association for Computational Linguistics.

Hosseini-Asl, E., McCann, B., Wu, C.-S., Yavuz, S., and Socher, R. (2020). A Simple Language Model for Task-Oriented Dialogue. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 20179–20191. Curran Associates, Inc.

Li, X., Chen, Y.-N., and Li, L. (2017). End-to-End Task-Completion Neural Dialogue System. In *Proceedings of the The 8th International Joint Conference on Natural Language Processing*, page 733–743.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Mo, K., Zhang, Y., Li, S., Li, J., and Yang, Q. (2018). Personalizing a Dialogue System with Transfer Reinforcement Learning. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*.

Montenegro, J. L. Z., da Costa, C. A., and da Rosa Righi, R. (2019). Survey of conversational agents in health. *Expert Systems with Applications*, 129:56–67.

Nishimoto, B. E. and Reali Costa, A. H. (2019). Dialogue Management with Deep Reinforcement Learning: Balancing Exploration and Exploitation. In *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 449–454.

Saha, T., Gupta, D., Saha, S., and Bhattacharyya, P. (2020). Towards integrated dialogue policy learning for multiple domains and intents using Hierarchical Deep Reinforcement Learning. *Expert Systems with Applications*, 162:113650.

Schatzmann, J. and Young, S. (2009). The Hidden Agenda User Simulation Model. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(4):733–747.

Su, P., Gasic, M., Mrksic, N., Rojas-Barahona, L. M., Ultes, S., Vandyke, D., Wen, T., and Young, S. J. (2016). Continuously Learning Neural Dialogue Management. *CoRR*, abs/1606.02689.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press, Cambridge, MA, USA, 2nd edition.

Takanobu, R., Zhu, H., and Huang, M. (2019). Guided Dialog Policy Learning: Reward Estimation for Multi-Domain Task-Oriented Dialog. In *EMNLP-IJCNLP*, pages 100–110.

Takanobu, R., Zhu, Q., Li, J., Peng, B., Gao, J., and Huang, M. (2020). Is your goal-oriented dialog model performing really well? empirical analysis of system-wise evaluation. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 297–310, 1st virtual meeting. Association for Computational Linguistics.

Vlasov, V., Drissner-Schmid, A., and Nichol, A. (2018). Few-Shot Generalization Across Dialogue Tasks. *CoRR*, abs/1811.11707.

Vlasov, V., Mosig, J. E. M., and Nichol, A. (2019). Dialogue Transformers. *ArXiv*, abs/1910.00486.

Wang, S., Zhou, K., Lai, K., and Shen, J. (2020). Task-completion dialogue policy learning via Monte Carlo tree search with dueling network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3461–3471, Online. Association for Computational Linguistics.

Weisz, G., Budzianowski, P., Su, P., and Gašić, M. (2018). Sample Efficient Deep Reinforcement Learning for Dialogue Systems With Large Action Spaces. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(11):2083–2097.

Weizenbaum, J. (1966). ELIZA - a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Williams, J. D., Asadi, K., and Zweig, G. (2017). Hybrid Code Networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 665–677, Vancouver, Canada. Association for Computational Linguistics.

Yan, Z., Duan, N., Chen, P., Zhou, M., Zhou, J., and Li, Z. (2017). Building Task-Oriented Dialogue Systems for Online Shopping. In *AAAI Conference on Artificial Intelligence*.

Zhang, Z., Takanobu, R., Zhu, Q., Huang, M., and Zhu, X. (2020). Recent Advances and Challenges in Task-oriented Dialog System. *ArXiv*, abs/2003.07490.

Zhao, T. and Eskenazi, M. (2016). Towards End-to-End Learning for Dialog State Tracking and Management using Deep Reinforcement Learning. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 1–10, Los Angeles. Association for Computational Linguistics.