AUDIO EMBEDDINGS HELP TO LEARN BETTER DIALOGUE POLICIES

Asier López Zorrilla, M. Inés Torres

University of the Basque Country UPV/EHU, Spain

Heriberto Cuayáhuitl

University of Lincoln, U.K.

ABSTRACT

Neural transformer architectures have gained a lot of interest for text-based dialogue management in the last few years. They have shown high learning capabilities for open domain dialogue with huge amounts of data and also for domain adaptation in task-oriented setups. But the potential benefits of exploiting the users' audio signal have rarely been explored in such frameworks. In this work, we combine text dialogue history representations generated by a GPT-2 model with audio embeddings obtained by the recently released Wav2Vec2 transformer model. We jointly fine-tune these models to learn dialogue policies via supervised learning and two policy gradient-based reinforcement learning algorithms. Our experimental results, using the DSTC2 dataset and a simulated user model capable of sampling audio turns, reveal that audio embeddings lead to overall higher task success (than without using audio embeddings) with statistically significant results across evaluation metrics and training algorithms.

***Index Terms*—** Spoken dialogue systems, audio embeddings, reinforcement learning

1. INTRODUCTION

Spoken dialogue systems are inherently devoted to process the users' audio signal and to provide the most convenient response given the dialogue context. But due to the difficulties of working directly with audio signals, these are often mapped into words using an Automatic Speech Recogniser (ASR), and then Natural Language Processing (NLP) techniques are applied to understand the user and act accordingly. This is probably the go-to approach, which has been motivated by recent advances in both ASR and NLP. However, a speech audio signal is much richer than the words it can be mapped into—even with perfect recognition—it contains information about prosody, the users' emotional mood or the noise level of the environment, for instance, which should be relevant for a better decision making in human-machine dialogues. This argument is supported by previous studies with young adults that have compared video chat, audio chat and text-based chat, where the latter has shown lower levels of bonding than the other forms of interaction [1].

In this work, we study how audio embeddings can be used to include this kind of information in the dialogue manager (DM) of a dialogue system, and whether including it would yield better dialogues policies. To this end, our first contribution is the creation of a transformer-based DM capable of processing both the text dialogue history and the audio signal of the last user's turn. We employ a fine-tuned GPT-2 [2] network to process the dialogue history and the recent Wav2Vec2 model [3] to handle the users' audio. We compare this DM against a version of itself that does not use audio in different conditions and with different learning algorithms.

Regarding algorithms for training dialogue policies, we train them first via supervised learning (SL) and then via policy gradient-based reinforcement learning (RL) with different reward functions. Spoken dialogue policies are often trained via RL using user models (UMs) that output dialogue acts or text. The effect of the dialogue being spoken is usually simulated by introducing artificial ASR errors. However, using such UMs is not suitable in our framework for two reasons: (1) the audio signal corresponding to the user's turn needs to be fed into the DM; and (2) the amount of audio data of real users is very limited and often scarce. We overcome these limitations with our second contribution: a novel User Audio Sampler. This module is capable of sampling an audio turn that corresponds to the output of the UM from the corpus taking into account the dialogue state (including dialogue act, turn, and repetitions). Our experiments are carried out using the DSTC2 dataset, since the most recent dialogue datasets do not include audio. The results on the test UM are in favour of our hypothesis: audio embeddings help to learn better dialogue policies.

The rest of the paper describes related works in Section 2, our proposed approach for audio-based policy learning in Section 3, our experimental framework (corpus, simulation pipeline and learning algorithms) in Section 4, experimental results in Section 5, and Section 6 presents our conclusions.

2. RELATED WORK

Transformer architectures such as BERT [4] or GPT-2 [2] have shown great potential for text-based dialogue management, both in open domain [5] and goal oriented setups [6]. More recently, transformers such as Wav2Vec2 [3] have been successful in audio-related tasks. Similar to the NLP transformers, Wav2Vec2 has been pretrained on large amounts of data via self-supervised learning, and has demonstrated to be easy to fine-tune for audio-related tasks like ASR [3], emotion recognition [7] or spoken language understanding [8] – even if (very) small domain specific data is available. We are not aware of any work combining these two kinds of transformers to build any similar DM to the one presented in this paper.

Moreover, the number of previous works describing dialogue systems that process the users' audio directly (without an ASR) is rather scarce. [9] and [10] present sequence-to-sequence models that process audio features in the context of audio visual scene-aware dialogue [11, 12], where the system has to answer a number of questions related to an audio visual scene. However, the audio to be analysed is not the users' audio, but the scenes' one. Closer to our approach is [13], who explore the inclusion of user sentiments in end-to-end dialogue systems. They train a dialogue policy that takes some audio features as input via SL. They found however that using the output of an external sentiment classifier worked better than the raw features. They also fine-tuned their DM using RL, but without including audio features. The work presented in [14] is probably the closest to our proposal. They investigate the inclusion of users' audio in an LSTM-based encoder-decoder network for response generation in open-domain dialogue – without reinforcement learning. To this end, they first train word-level audio embeddings in a response selection task and then concatenate those to traditional word embeddings to form the input to the network. In contrast to their work, our approach is based on reinforcement learning, it is simpler in terms of implementation, it requires no further pretraining, and we apply it to task-oriented dialogue data for the first time. In addition, the audio representations used in [14] were trained without taking into account the word order in the turns, which could miss valuable audio information such as the prosody.

In this paper we show that our DM can be easily fine-tuned using both SL and RL. Even though (deep) RL has established methodologies to train many kinds of dialogue systems [15, 16, 17, 18], the user simulators employed in previous works only generate either words or dialogue acts – not audio. The proposed user audio sampler allows the simulation of audio-based dialogue turns. This is novel in the area of dialogue policy learning. We refer the readers to [19] for a more-in-depth analysis of RL in spoken dialogue systems.

3. AUDIO EMBEDDINGS APPROACH

This section describes the methodological contributions of the proposed approach: the transformer-based dialogue model that creates and processes audio embeddings, and the User Audio Sampler that enables dialogue policy learning via RL.

3.1. Text dialogue history combining audio information

The proposed dialogue model, illustrated in Figure 1, combines two inputs to represent the state of the dialogue: the

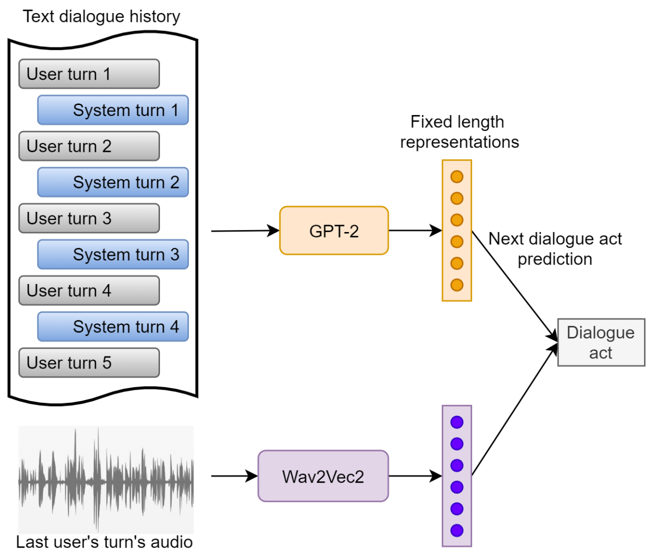


Fig. 1. Proposed dialogue manager diagram.

text dialogue history and the last user turn's audio. Dialogue acts are used as output because they facilitate the integration of a user model (UM) and the policy optimisation with RL.

Dialogue history. In order to keep our approach as simple and general as possible, each turn in the dialogue history is represented as text, i.e. no dialogue acts or name entities are used here. The sequence of turns are processed with a (pretrained) GPT-2 tokenizer and transformer network. The GPT-2 tokenizer is based on based on byte-level Byte-Pair-Encoding, which breaks each turn into sub-words and given as input to the transformer. To indicate whose turn is (user or system), we employ a similar strategy to [20]. First, we add a special token (`<sys_turn>` or `<user_turn>`) at the beginning of each turn. Second, we introduce a parallel sequence of inputs of the same length of the main input consisting of segment/speaker embeddings, which is also represented with the same `<sys_turn>` and `<user_turn>` tokens. Last, we add a `<da_pred>` token at the end of both sequences to indicate the GPT-2 model that the input sequences are complete and the dialogue act prediction should be made. We use the hidden state of the last decoder layer of the GPT-2 model after processing the `<da_pred>` token as the fixed-length vector representation of the dialogue history.

Last user's turn's audio. We use a Wav2Vec2 neural architecture to extract the audio embedding vectors for dialogue act prediction, alongside the aforementioned representation of the dialogue history. Wav2Vec2 [3] consists of two main components: a feature encoder that processes a raw audio signal via temporal convolution layers, and a transformer network on top of it. We follow the same approach to extract the fixed-length vector representation of the last user's audio. We use the hidden state of the transformer's last layer after processing the last chunk of the audio. As recommended by the authors, we keep the feature encoder frozen during training, and update the transformer to adapt it to our task.

Last, the dialogue history and audio representations are concatenated, and the unnormalised probability distribution of the next dialogue act is computed via a single linear layer.

3.2. User Audio Sampler

User simulations often output a dialogue act corresponding to the next user turn, which is then converted to text via a user Natural Language Generator (NLG). This approach, however, is not appropriate in the proposed framework because of the requirement of an audio signal corresponding to that text.

In order to optimise the chance of finding an audio corresponding to the output of the user model (UM), we do not use any NLG. Instead, we directly search for turns labeled with the same dialogue act and associated slots in the corpus of real dialogues. Multiple user turns are found in most cases, unless the UM generates dialogue act-slot combinations not appearing in the corpus. From the set of candidates, any turn should already be valid and its audio and transcription (if needed) could be provided to the next module in the dialogue system. However, we consider a couple of factors that make some candidates potentially more suitable than others.

First, we take into account the turn number of the current simulated dialogue compared to the turn number of a given candidate. The justification is as follows. Assume that a dialogue is taking too long and the user starts to feel tired of the interaction. The user may speak in a different way than in the first few turns (when the user first met the system). We assume that selecting a turn that occurred in a similar situation (turn number-wise) of the dialogue to the one the simulation is in should lead to more realistic simulated dialogues, and with potentially more relevant audio information.

The second factor is the number of repetitions of the dialogue act output by the UM in the dialogue, and its reasoning is the following. Assuming that a given dialogue act/slot combination has been used more than once in a dialogue, it is probably due to the system not understanding it correctly and requesting the same information again. When such a situation happens in a dialogue, users tend to get upset on the one hand, and to speak louder and slower on the other hand. This information should also be reflected in the audio signal and could be exploited to improve dialogue policies.

Thus, the sampling probability of the candidate user turns is computed as follows. First, we compute a score for each candidate in terms of the aforementioned two criteria according to $s_i = \frac{1}{|t_i - t_d|} + \frac{1}{|r_i - r_d|}$, where s_i is the score obtained by the i-th candidate, t_i is the turn number of the candidate in the original dialogue in the corpus, t_d is the turn number of the simulated dialogue, r_i is the number of repetitions of the dialogue act/slot combination in the original dialogue until the appearance of the candidate, and r_d is the number of repetitions of the dialogue act/slot combination in the simulated dialogue so far. The scores are then converted into probabilities by dividing them by the sum of all the scores: $p_i = \frac{s_i}{\sum_j^N s_j}$, where p_i is the probability of sampling the i-th candidate and N is the number of candidates.

4. EXPERIMENTAL FRAMEWORK

4.1. Corpus

Recent dialogue corpora released in the last few years (e.g. MultiWOZ [21], STAR [22] or SGD [23]) have focused on text based dialogue modelling and none include audio. The DSTC2 dataset [24] is by far the most used corpus for research in spoken dialogue technology and we use this corpus in this work. It contains 3235 human-machine dialogues on the topic of restaurant search acquired with three different dialogue systems. There are 8 slot types in the corpus: *area*, *food type*, *restaurant name*, *price range*, *address*, *phone*, *postcode* and *signature*. All the slots are requestable, i.e. the user can ask information about any of those. On the contrary, only the first 4 slot types are informable, i.e. they can be used as a constraint in the restaurant search. The corpus is split into three partitions: *train*, *dev* and *test*, which contain 1612, 506 and 1117 dialogues respectively. We merge the *dev* and *test* partitions to build the test UM. In that way the training and test UMs use similar amounts of data, and therefore this will not bias the behaviour of the User Audio Sampler, which is sensitive to the number of available audio turns to sample from.

4.2. Dialogue pipeline for simulations

We first initialise our dialogue policies with some iterations of SL, and then we will further improve them using RL on a UM built using the same corpus. To this end, we will use the simulation pipeline described next and illustrated in Figure 2. Let us describe its components.

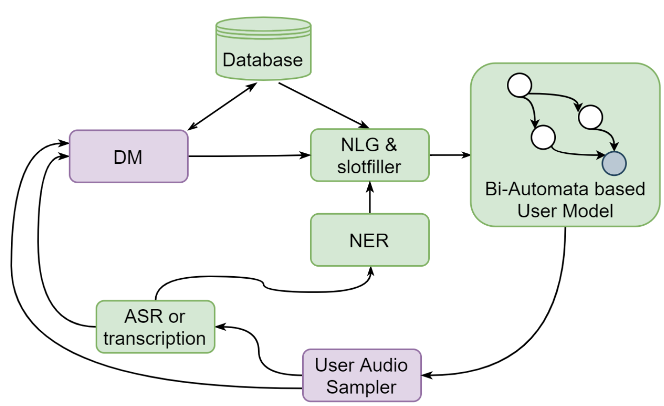


Fig. 2. Complete simulation pipeline, see DM in Figure 1. Boxes in purple indicate our contributions.

Dialogue Manager. For comparison purposes we use two DMs: our proposal (described in Section 3.1) and the same dialogue model but without using audio embeddings. For GPT-2 we use the publicly available *small* pretrained architecture, and for the Wav2Vec2 network the *base* one.

Regarding the dialogue act format, in the original DSTC2 challenge each turn was labeled with multiple dialogue acts, for example *confirm* | *area* and *request* | *food*. In order to simplify the learning task we follow the same procedure as in the DeepPavlov DSTC2 example [25], and whenever more than one dialogue act is found in the dataset, we concatenate them to form a new dialogue act. The example would correspond to *confirm* | *area* + *request* | *food*. Our dialogue history was truncated to include the last 9 turns.

Database. Although no database was released as part of DSTC2, database calls can be inferred from the data to form a large enough dataset to perform dialogue simulations with it. Our DMs are able to make database queries. In order to log this activity in the text dialogue history, every time a database query is made, a `<db_search>` token is added to the dialogue history. If the query is succesful and one or more restaurants are found, a `<db_result>` token is added next. A `<no_db_result>` token is concatenated otherwise.

Name Entity Recognizer. The outputs of our DMs are dialogue acts, and some of them contain one or a few slots. We use a simple Name Entity Recognizer (NER) based on fuzzy matching to extract name entities that can then be used to fill these slot of the dialogue acts. Our NER is a slightly improved version of the NER of DeepPavlov for this task.

Slot Filler and NLG. We use a rule-based slot filler to select the slot values associated to a dialogue act. As the dialogue progresses, we keep track of the name entities recognised by the NER and the output of the DB searches. Depending on the dialogue act, we fill the slots with the last values recognised by the NER or provided by the DB. The NLG module generates text corresponding to the system turns given a pair of dialogue act and selected slots using predefined templates. Since the UM works at the dialogue act level, this text is only used to fill the dialogue history processed by the DM.

User Model. The employed UM is based on Attributed Probabilistic Finite State Bi-Automata [26] and is an extended version [1] of the work presented in [27, 28]. This data-driven UM works at the dialogue act level, both for its input and output, and its goal is selected at the beginning of the simulations according to the same goal probability distribution found in the training data. In the DSTC2 corpus the user turns are labeled with one or more dialogue act and slot value combinations. In order to sample a user turn (audio included) according to a given UM output, the User Audio Sampler (Section 3.2) selects as candidates those user turns that exactly match all the dialogue act and slot combinations in the selected dataset partition (*train* for the training UM, *dev* and *test* for the test UM). If no candidates are found (20% of the dialogues) the simulation is prematurely finished. Fortunately and in 96% of cases when this happens, the sampling errors occur in the very first user turn, due to the constraint combination of the user goal not appearing in the dataset. This means that only very rarely—in 0.8% of the simulated dialogues—computation time is wasted without adverse effects.

ASR/Transcription. Our experiments are carried out with two types of text input: perfect speech recognition corresponding to manual transcriptions (TRS), and automatic speech recognition (ASR) corresponding to noisy outputs. For the latter we use a separate and publicly available Wav2Vec2 network, the *wav2vec2-base-960h* checkpoint that was fine-tuned for ASR on 960 hours of Librispeech [3].

4.3. Evaluation metrics

We measure the quality of the dialogue policies according to the following metrics based on [29].

User Request Score (URS): this score (between 0 and 1) indicates whether the system answers to the user in focus. For example, this score is high if the system provides a phone number after the user has requested it. This metric does not take into account, however, whether that phone number is correct or not, i.e., if it corresponds to the restaurant the user was talking about or not. Note that whenever the user did not explicitly request any information, this score is not computed. A typical case where this happens is when the system provides information to the user without the user requesting it. This happens in 15% of the simulated dialogues.

System Offered Valid Venue (SOVV): this score (also between 0 and 1) indicates how correct the system informs are. It is the ratio of the system informs that satisfy the constraints of the user over the total informs.

Can't Help Score (CHS): this score (also between 0 and 1) is only computed in a fraction of the dialogues, about 20% approximately. Sometimes the UM has unreachable goals, for example a user may want to find a Basque restaurant in the north but there is none. In that case, the system should inform that there is no way to find such a restaurant. This score is 1 if the system provides this information, and 0 otherwise.

4.4. Supervised Learning (SL)

Prior to RL, we fine-tune four versions of the DM to adapt the GPT-2 and Wav2Vec2 models to the DSTC2 task via SL: (1) a DM without audio embeddings using the users' transcriptions; (2) a DM with audio embeddings using the users' transcriptions; and (3,4) another pair of DMs where the textual input is generated by the aforementioned ASR. Each model was fine-tuned using the training data and optimised according to the cross entropy loss at the dialogue act level for 3 epochs. A batch size of four was used throughout all the experiments, and the loss was minimised using the Adam optimiser with a learning rate of 5e-5.

[1] We would like to acknowledge Manex Serras for providing us with the Bi-Automata-based UM.

4.5. Reinforcement Learning (RL)

We further optimised the models above using policy gradient RL algorithms. To do that, we used the REINFORCE algorithm [30] with three different reward functions. In addition, we used an Actor-Critic algorithm [31] with the best of the three designed reward functions described as follows.

$$R_1 = \begin{cases} 100 \times \textit{evaluation score}, & \text{if end of dialogue} \\ -1, & \text{otherwise,} \end{cases}$$

$$R_2 = \begin{cases} 100 \times \textit{evaluation score}, & \text{if end of dialogue} \\ -0.1, & \text{otherwise,} \end{cases}$$

$$R_3 = \begin{cases} 100 \times \textit{evaluation score} - 50\lambda, & \text{if end of dialogue} \\ 50 \times \textit{evaluation score} - 25\lambda, & \text{if } \lambda \text{ has changed} \\ -0.1, & \text{otherwise,} \end{cases}$$

where *evaluation score* is a weighted average of the evaluation metrics, and $\lambda = 1 - \textit{evaluation score}$.

While the justification of R_2 is due to dialogue optimisation with less weight on dialogue length, R_3 is motivated by using denser rewards as opposed to sparse ones. The weights for the URS, SOVV and CHS scores in *evaluation score* are 0.2, 0.4 and 0.4 respectively in the case of R_2; and 0.1, 0.5 and 0.4 in the case of R_3. If any metric value is *NaN* after the dialogue evaluation, the weights of the remaining scores are increased proportionally according to their value.

The hyperparameter and implementation details of the RL algorithms is as follows. In all the cases a discount factor of 0.95 and ADAM optimiser were used with a learning rate of 5e-6. In the case of the Actor-Critic algorithm, the Actor and the Critic use separate networks initialised with the resulting weights after the SL stage (except the last linear layer of the critic). We experienced some convergence problems with the actor-critic algorithm. We solved them by implementing two separate losses, one for the actor and the other for the critic. We also used gradient clipping to prevent gradient exploding. Each training run, out of 7 runs to provide statistically significant results, used 10K simulated dialogues.

5. RESULTS

For each learning algorithm-model pair, the results correspond to 7K simulated dialogues (1000 after each of the 7 experiments in the case of the RL algorithms) with the test UM. Table 1 shows the cumulative reward obtained in the test UM both after SL and RL. The evaluated reward in the RL algorithms is the same that was used during the training stage. Values in bold indicate that a given model performs better than its counterpart. We have additionally performed unpaired t-tests for statistical significance with the performance values obtained after the 7000 test dialogues: one star (*) means that the obtained p-value < 0.05 and two stars (**) that the p-value < 0.01. We used the Welch's t-test to this end, which tests whether two populations have equal means without assuming equal variances.

Table 1. Cumulative reward on the test UM after SL and RL.

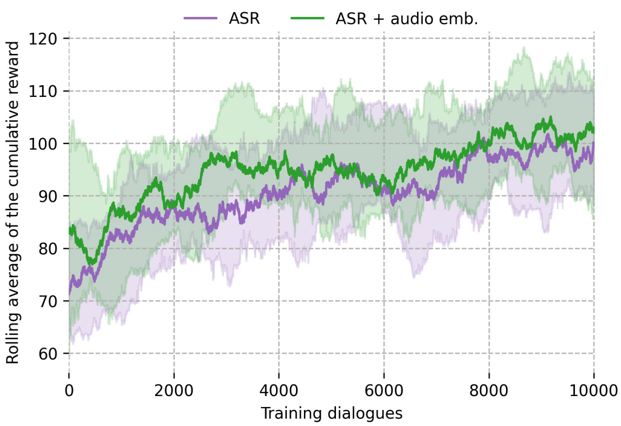
	ASR	ASR + AE	TRS	TRS + AE
SL R_1	50.4	**50.8**	67.3	**74.4****
SL R_2	61.2	**62.1**	76.2	**81.5****
SL R_3	74.9	**83.9****	127.2	**142.4****
REINFORCE R_1	62.3	**66.2****	78.9	**79.7**
REINFORCE R_2	68.8	**71.6****	84.1	**87.7****
REINFORCE R_3	92.1	**95.4****	134.4	**145.2***
Actor-Critic R_3	114.7	**119.8****	166.5	**179.3****

Our results above provide evidence to support the argument that learnt dialogue policies using audio embeddings are superior than their counterpart (without audio embeddings), both after SL and RL. The cumulative reward—overall performance of the evaluation metrics—is always higher, and statistically significant in most of the cases. Breaking down the results into the three evaluation metrics, as shown in Table 2, we can see that the biggest difference in performance lies in the SOVV metric. This makes sense because its weight is the highest in the reward functions. The other metric with a similar weight is CHS, but since it is only computed around a 20% of the times, its total contribution to the cumulative reward is much lower, and therefore CHS is not optimised as much as the SOVV metric by the RL algorithms. Our results also show that the USR metric is harder to optimise because the policies trained via SL already achieve near optimal performance. This justifies a low weight for this metric in the reward functions. As expected, better results in the test UM are derived from a more successful training. Figure 3 shows the evolution of dialogue policy learning with reward R_3 and the Actor-Critic RL algorithm.

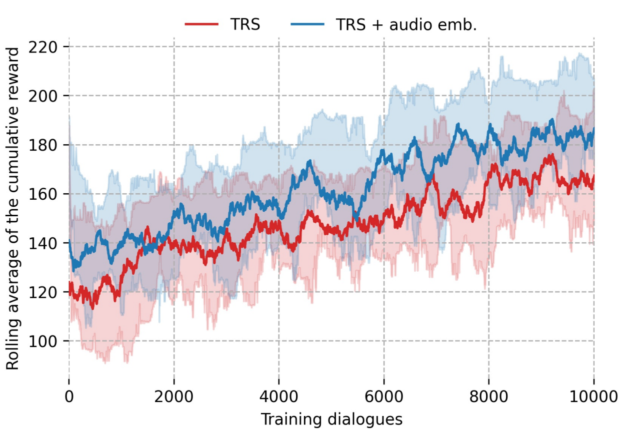
To analyse our results further, we plotted the dialogue act frequency distribution of different DMs. Figure 4 shows this using the best REINFORCE-based DM (with R_3 rewards) in two ways: in general terms (using all user turns), and filtering noisy user turns where the audio could be potentially relevant. We have grouped the dialogue acts in general categories and have plotted the most frequent ones to keep the histogram as clear as possible. We should note that the noisy user turns occur in $\sim 7\%$ in our dataset. In this Figure (4) we can observe that the audio-driven DMs select more informs and restaurant offers instead of requests, especially after noisy user turns. This behaviour indicates that the system is probably aware that it has difficulties in understanding the user, and prefers to provide information according to the data it has gathered until that point in the dialogue, which could possibly be what the user is looking for – as supported by our SOVV metric.

Table 2. Evaluation metrics comparing policies after SL and RL. Notation: R.=REINFORCE algorithm, AC=Actor-Critic.

	SOVV				URS				CHS			
	ASR	ASR + AE	TRS	TRS + AE	ASR	ASR + AE	TRS	TRS + AE	ASR	ASR + AE	TRS	TRS + AE
SL	0.746	**0.748**	0.873	**0.938****	0.969	**0.976***	0.985	**0.991****	0.658	**0.681**	**0.976**	0.971
R. R_1	0.744	**0.766****	0.868	**0.872**	**0.980****	0.958	**0.994***	0.990	**0.550***	0.505	**0.920****	0.842
R. R_2	0.747	**0.769****	0.874	**0.913****	**0.961**	0.957	**0.990**	0.988	**0.579**	0.569	**0.863**	0.852
R. R_3	0.765	**0.795****	0.906	**0.921****	**0.958****	0.933	0.979	**0.981**	**0.627**	0.623	0.903	**0.921**
AC R_3	0.765	**0.777***	0.938	**0.951***	0.985	**0.986**	**0.993**	0.992	**0.728**	0.710	0.916	**0.922**

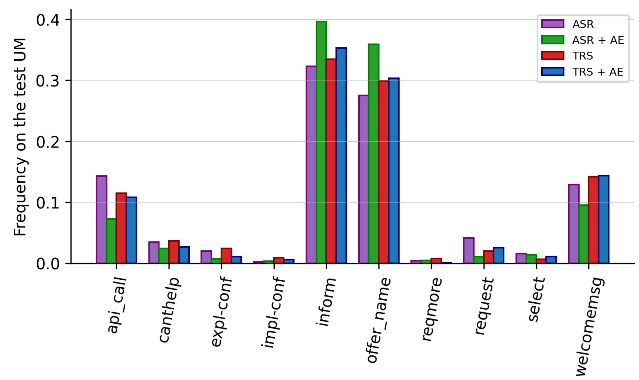


(a) With ASR output.

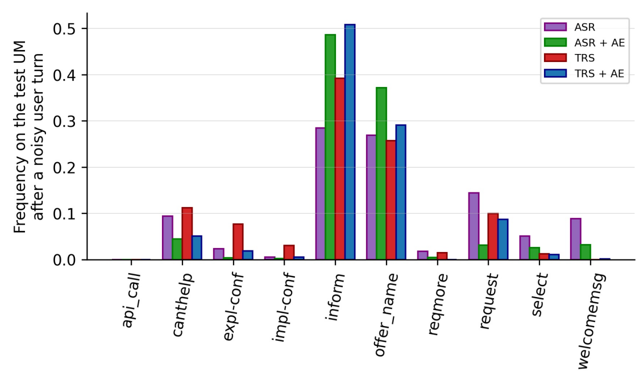


(b) With manual transcription.

Fig. 3. Cumulative rewards on the training UM with the R_3 reward function using the Actor-Critic RL algorithm.



(a) General.



(b) After noisy user turns.

Fig. 4. Histograms of frequent dialogue acts produced by four versions of the DM on the test UM after REINFORCE R_3.

6. CONCLUSION AND FUTURE WORK

We have presented a novel methodology to fine-tune strong pretrained transformers capable of processing audio and text for dialogue policy learning. Compared to those that do not use the users' audio, audio-driven policies obtain significantly higher rewards on a simulated test user model with the DSTC2 dataset, both after SL and RL. The higher rewards also lead to higher task-completion rates. Among the evaluation metrics, SOVV improves the most, while the URS and CHS scores do not improve as much because the URS performance was already very high even with the SL-baseline, and the CHS is only computed in 20% of the dialogues. In future work we plan to optimise the contribution of each metric in our reward function using learnt rewards, to test our approach using a larger set of RL algorithms and DM networks, and further validate our contributions in other tasks and corpora.

7. REFERENCES

[1] Lauren E. Sherman, Minas Michikyan, and Patricia Greenfield, “The effects of text, audio, video, and in-person communication on bonding between friends,” *Cyberpsychology: Journal of Psychosocial Research on Cyberspace*, vol. 7(2), 2013.

[2] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever, “Language models are unsupervised multitask learners,” 2019.

[3] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli, “wav2vec 2.0: A framework for self-supervised learning of speech representations,” in *NeurIPS*, 2020.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[5] Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric M Smith, et al., “Recipes for building an open-domain chatbot,” *arXiv preprint arXiv:2004.13637*, 2020.

[6] Donghoon Ham, Jeong-Gwan Lee, Youngsoo Jang, and Kee-Eung Kim, “End-to-end neural pipeline for goal-oriented dialogue systems using gpt-2,” in *ACL*, 2020.

[7] Leonardo Pepino, Pablo Riera, and Luciana Ferrer, “Emotion recognition from speech using wav2vec 2.0 embeddings,” *arXiv preprint arXiv:2104.03502*, 2021.

[8] Seunghyun Seo, Donghyun Kwak, and Bowon Lee, “Integration of pre-trained networks with continuous token interface for end-to-end spoken language understanding,” *arXiv preprint arXiv:2104.07253*, 2021.

[9] Dat Tien Nguyen, Shikhar Sharma, Hannes Schulz, and Layla El Asri, “From film to video: Multi-turn question answering with multi-modal context,” *CoRR*, vol. abs/1812.07023, 2018.

[10] Hung Le, Doyen Sahoo, Nancy F. Chen, and Steven C. H. Hoi, “Multimodal transformer networks for end-to-end video-grounded dialogue systems,” in *ACL*, 2019.

[11] Huda AlAmri, Vincent Cartillier, Abhishek Das, Jue Wang, Anoop Cherian, Irfan Essa, Dhruv Batra, Tim K. Marks, Chiori Hori, Peter Anderson, Stefan Lee, and Devi Parikh, “Audio visual scene-aware dialog,” in *CVPR*, 2019.

[12] Huda AlAmri, Vincent Cartillier, Raphael Gontijo Lopes, Abhishek Das, Jue Wang, Irfan Essa, Dhruv Batra, Devi Parikh, Anoop Cherian, Tim K. Marks, and Chiori Hori, “Audio visual scene-aware dialog (AVSD) challenge at DSTC7,” *CoRR*, vol. abs/1806.00525, 2018.

[13] Weiyan Shi and Zhou Yu, “Sentiment adaptive end-to-end dialog systems,” in *ACL*, 2018.

[14] Tom Young, Vlad Pandelea, Soujanya Poria, and Erik Cambria, “Dialogue systems with audio context,” *Neurocomputing*, vol. 388, 2020.

[15] Iñigo Casanueva, Pawel Budzianowski, Pei-Hao Su, Stefan Ultes, Lina Maria Rojas-Barahona, Bo-Hsiang Tseng, and Milica Gasic, “Feudal reinforcement learning for dialogue management in large domains,” in *NAACL-HLT*, 2018.

[16] Heriberto Cuayáhuitl, Seunghak Yu, Ashley Williamson, and Jacob Carse, “Scaling up deep reinforcement learning for multi-domain dialogue systems,” in *IJCNN*, 2017.

[17] Ryuichi Takanobu, Hanlin Zhu, and Minlie Huang, “Guided dialog policy learning: Reward estimation for multi-domain task-oriented dialog,” in *EMNLP-IJCNLP*, 2019.

[18] Jason D. Williams and Geoffrey Zweig, “End-to-end lstm-based dialog control optimized with supervised and reinforcement learning,” *CoRR*, vol. abs/1606.01269, 2016.

[19] Siddique Latif, Heriberto Cuayáhuitl, Farrukh Pervez, Fahad Shamshad, Hafiz Shehbaz Ali, and Erik Cambria, “A survey on deep reinforcement learning for audio-based applications,” *CoRR*, vol. abs/2101.00240, 2021.

[20] Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue, “Transfertransfo: A transfer learning approach for neural network based conversational agents,” *arXiv preprint arXiv:1901.08149*, 2019.

[21] Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić, “Multiwoz–a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling,” *arXiv preprint arXiv:1810.00278*, 2018.

[22] Johannes EM Mosig, Shikib Mehri, and Thomas Kober, “Star: A schema-guided dialog dataset for transfer learning,” *arXiv preprint arXiv:2010.11853*, 2020.

[23] Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan, “Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset,” in *AAAI*, 2020, vol. 34.

[24] Matthew Henderson, Blaise Thomson, and Jason D Williams, “The second dialog state tracking challenge,” in *SIGDIAL*, 2014.

[25] Mikhail Burtsev, Alexander Seliverstov, Rafael Airapetyan, Mikhail Arkhipov, Dilyara Baymurzina, Nickolay Bushkov, Olga Gureenkova, Taras Khakhulin, Yurii Kuratov, Denis Kuznetsov, et al., “Deeppavlov: Open-source library for dialogue systems,” in *ACL System Demonstrations*, 2018.

[26] M Inés Torres, “Stochastic bi-languages to model dialogs,” in *FSMNLP*, 2013.

[27] Manex Serras, María Inés Torres, and Arantza del Pozo, “Goal-conditioned user modeling for dialogue systems using stochastic bi-automata.,” in *ICPRAM*, 2019.

[28] Manex Serras, María Inés Torres, and Arantza del Pozo, “Improving dialogue smoothing with a-priori state pruning.,” in *ICPRAM*, 2020.

[29] Florian Kreyssig, Iñigo Casanueva, Paweł Budzianowski, and Milica Gasic, “Neural user simulation for corpus-based policy optimisation of spoken dialogue systems,” in *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, 2018, pp. 60–69.

[30] Ronald J Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine learning*, vol. 8, no. 3-4, 1992.

[31] Vijay R Konda and John N Tsitsiklis, “Actor-citic agorithms,” in *NIPS*, 1999.