CS230

A Dialogue Policy with Conversation State Embeddings (NLP)

David Brown *
Department of Computer Science
Stanford University
davidwb@stanford.edu

1 Introduction

The purpose of this project is to develop a deep learning model capable of predicting an appropriate agent action in response to a history of dialogue turns, where the dialogue is modeled as a sequence of human and agent actions. Each dialogue turn consists of an utterance and semantic metadata such as the action type, entities detected in the utterance, and the overall dialogue state which includes the current user intent and a set of "slot" values which are recorded and updated over the course of the conversation and used to fulfil the user's goals. The **input** to our model is an encoded sequence of dialogue state features, i.e. the encoded conversation history. The **output** is a probability distribution over all agent action types.

Selection of appropriate agent actions at each turn of a conversation is a common problem in today's ML-based dialogue systems [9, 4, 1, 5]. Prior methods have often relied on rule-based techniques involving complicated and hard-coded state machines. While those methods work for simple agents they quickly become untenable as developers add more and more capabilities to an agent. Furthermore, this error-prone work is domain-specific must be repeated for every new agent applied in a different setting. Instead, it would be desirable for agents to *learn* the desired behavior from corpora of annotated conversations.

We refer to the process of selecting agent actions as a *dialogue policy*. Because collection of high-quality dialogue data is difficult, we seek to learn robust action policies from minimal data. That goal is addressed in the current work by employing a pretrained contextual embedding model, namely DistilBERT [6], that belongs to the BERT [2] family. Our action prediction model uses DistilBERT at each turn of the dialogue history to embed the sequence utterances into a shared vector space. The embedding vectors serve to augment the encoded semantic metadata with additional context. We find empirically that this significantly enhances the validation accuracy and loss of the action predictor model.

2 Related Work

This project is inspired primarily by Vlasov *et al.*[9] who developed a similar transformer based action prediction policy model. The key difference between [9] and this work is that our model makes use of the raw utterances, embedding them using DistilBERT, while in [9], only binary encodings of the semantic annotations of each dialogue turn were used. Yang *et al.* [10] also train a system action predictor (SAP) model by they take a more sophisticated end-to-end approach that comes NLU/NER

*NDO Student in the graduate certificate program in AI. https://www.linkedin.com/in/davidwb/

CS230: Deep Learning, Fall 2020, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

and SAP into a single end-to-end model. They find that their joint task learning model is able to mitigate the affects of noisy NLU outputs. Although the work in [10] is ambitious and noteworthy, that model does not make use of pretrained contextual embeddings of utterances which makes it less likely generalize well on small training sets, a key requirement of dialogue systems in production settings.

3 Data

3.1 Dataset and Features

The first dataset used in this work is from REDP [8]. REDP consists of 108 fully annotated dialogues, where every dialogue turn is either a user action or an agent action. Agent actions are annotated with an action label. User turns are annotated with an intent label and a list of entity types detected in the user utterance. Unfortunately, REDP does not contain utterances for user actions and the dataset size is highly limited.

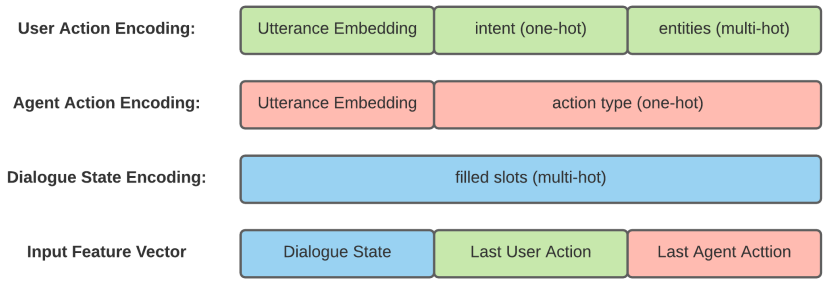
The other primary dataset we used is MultiWOZ2.2 [3], which is a much larger dataset containing over 10k annotated multi-domain dialogues, including raw utterances and span annotations for entities. It is one of the best datasets available for dialogue state tracking. An example conversation from the "hotel" domain is shown in 1, where we can see the sequence of user and agent utterances as well as complete annotations for the turn. The dialogue state of filled slots is omitted.

MultiWOZ is a complex dataset containing conversations for multiple domains and also conversation that individually span multiple domains. MultiWOZ is also used for a variety of dialogue state tracking (DST) tasks and not just action prediction. Thus to simplify the data for the needs of developing our model, we extracted dialogues pertaining only to the "hotel" domain and transformed then into a form suitable for our training purposes. We resulting final dataset consists of exactly 657 conversations. The longest conversation contained 12 agent actions for a total of 24 turns.

4 Preprocessing

Preprocessing was a challenging component of this project. The overall encoding scheme we used to generate the final input data for our model is illustrated in figure 2. All user dialogue turns are encoded with a embedding of the utterance, a one-hot vector for the intent and a multi-hot vector indicating the presence or absence of each entity types detected in the user utterance. The entities values are ignored, although that is an interesting opportunity for future work. Agent actions are encoded with an embedding of the agent utterance and a one-hot encoding of the action type.

Figure 2: Input Encoding Scheme



5 Methods

The problem addressed in this work amounts to a multi-class classification problem where we aim to correctly predict one of a set of possible agent actions given a history of dialogue turns. It is important to note that at each turn of a dialogue, there may be multiple equally valid choices for the next agent action. Thus accuracy alone, although a useful metric, is not sufficient and also somewhat

Figure 1: Example MWOZ Dialogue (Hotel Domain)

Annotated Dialogue Example		
USER ACTION	**AGENT ACTION**	**ANNOTATIONS**
I am looking for a particular hotel. Its name is called autumn house		Intent = Hotel-Inform Entities = [Name = autumn house]
	I have the Autumn House. It is at 710 Newmarket Road. Would you like the phone number?	Action = Hotel-Inform Inform-Slots = [Name = Autumn House, Addr = 710 Newmarket Road]
No thanks. Would you book the Autumn House for me starting on Monday, please.		Intent = Hotel-Inform Name=Autumn House
	How many people would be staying and how many days will you be staying?	action = Booking-Request Requested-Slots = [People, Stay]
That's for 8 people and it's for 2 nights.		Intent = Hotel-Inform Entities = [Stay = 2 nights People = 8]
	I'm sorry, your booking was unsuccessful. Would you like to book another day or a shorter stay?	Action = Booking-NoBook
Could you try Wednesday, instead?		Intent = Hotel-Inform Entities = [Day = Wednesday]
	Booking was successful. Reference number is : 3H0WHD4Z.	Action = Booking-Book Inform Slots = [Ref:3H0WHD4Z]
Thank you goodbye		Intent = general-thank
	You're welcome. Thank you for calling and goodbye.	Action = general-goodbye

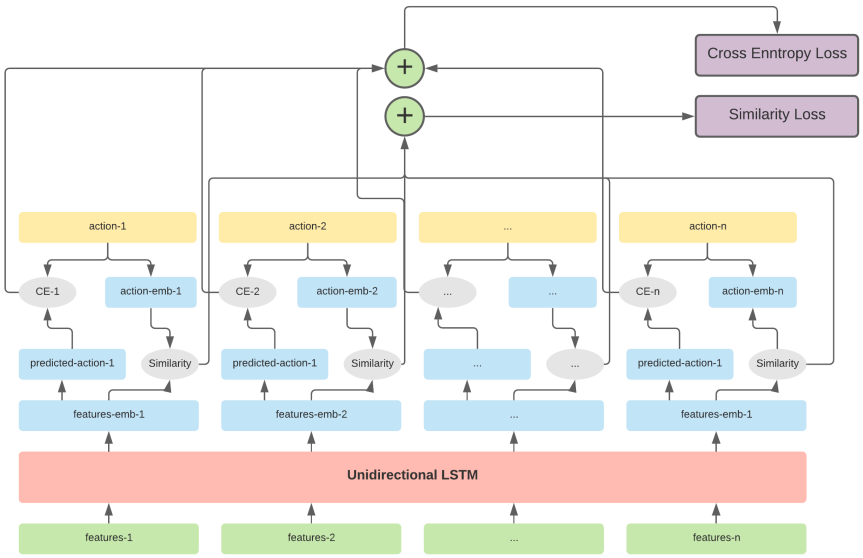
misleading because use of a simple cross-entropy loss function would penalize all wrong actions equally even though clearly some agent actions will be more or less appropriate than others.

To account for this, our model uses an embedding layer for agent actions. The idea is that similar actions will be closer together in the embedding space, while highly different actions will be farther apart. We therefore use a loss function consisting of two equally weighted parts:

1. A categorical cross-entropy loss on the predicted action type.
2. A cosine similarity loss on the action embeddings.

The utterances at each turn of the conversation are embedded in a shared vector space via a DistilBERT model layer the utterance embeddings are concatenated with the other encoded dialogue features before the full sequence of input features is fed a recurrent unidirectional LSTM model. An opportunity for future work is to replace this LSTM with a unidirectional transformer model which may perform better due to its ability to selectively attend over the entire dialogue history and capture long-range dependencies [7]. The overall architecture we use model is similar to the one used in [9], although that model does not including utterance embeddings. The full structure of our system action prediction (SAP) model is illustrated in figure 3.

Figure 3: Baseline LSTM Model



The features input at the bottom at each time step are the encoding of the most recent user action, most recent agent action, and the global dialogue state which consists of the slots filled in throughout the conversation.

The actions input at the top are the encoding of the agent action for each agent action that occurs during the conversation. Note that the number of timesteps input to the model is equal to the total number of agent actions in the dialogue.

6 Results

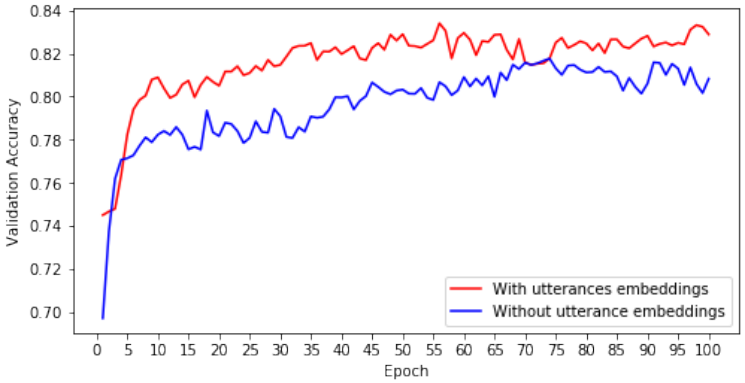
Our model achieves approximately 82% action prediction accuracy on the MWOZ2.2 hotel conversations. The difference in validation accuracy and loss between using and not using utterance embeddings is depicted in figures 4 and 5. We can see that including utterance embeddings in the training data yields a slight but consistent improvement in prediction performance. However, this slight improvement comes at a significant computational cost, since a large BERT model must be evaluated separately at every dialogue turn.

The fact that training with semantic annotations alone yields comparative performance to training with utterance embeddings is somewhat surprising. This suggests that for a similar model used in a production setting, it may be sufficient and even advisable to train and predict without utterance embeddings. Evaluating the model without utterance embeddings can be done in real-time without a GPU and could thus save significant cost and compute time in a production environment. However, obtaining semantic annotations in the first place necessitates that a large model must be used earlier in the pipeline as a prepossessing step before action prediction, and utterance embeddings could be cached and carried over from the preceding stages.

7 Conclusion

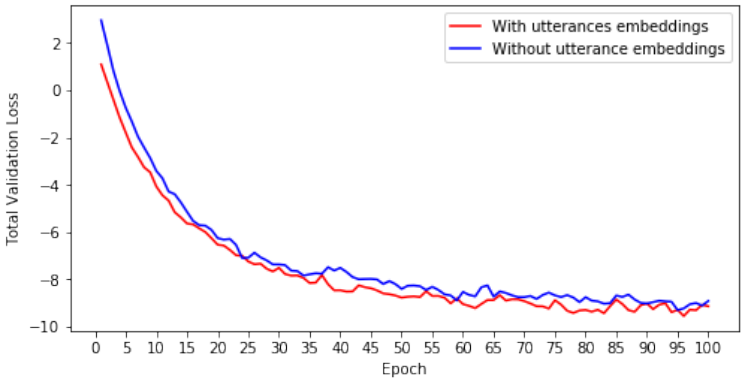
In this project we developed an LSTM based dialogue action prediction model that takes as input an encoded conversation history and predicts as output the next agent action from a finite set of possible actions. Our model achieves are 80% accuracy on the chose dataset and also learns similarity scores so that highly similar action types cluster together in the embeddings space. We observe that including utterances in training data yields a modest improvement in performance, but at a large computational cost.

Figure 4: Validation Accuracy



In this figure, the epoch is shown on the horizontal axis and accuracy on the vertical axis. The red line depicts accuracy for the model using utterances embeddings. The blue line is for the model without utterance embeddings.

Figure 5: Total Validation Loss



The red line depicts loss for the model using utterances embeddings. The blue line is for the model without utterance embeddings.

This project presents several noteworthy opportunities for future work. These are (in order of importance)

1. Turning the current model into a multi-tasks model capable of predicting not just the next agent action, but also the current intent, entities, and slot values altogether. Such an end-to-end approach would simplify production deployments by requiring only a single model for the entire NLU dialogue subsystem. It could also potentially yield performance improvements since multi-task models have been observed to outperform single-task models.
2. Including the slot and entity values in the dialogue turn encodings. In principle, information about the slot values and not merely the presence of slots could help inform agent actions.
3. Using a unidirectional transformer model in place of the LSTM. This idea could yield significant benefits, especially for long dialogues or multi-domain dialogues which both often have long-range decencies would could be better modeled by a transformer architecture.

8 Contributions

This project was carried out independently by David Brown (davidwb@stanford.edu) who is the sole author and contributor.

References

[1] Antoine Bordes, Y-Lan Boureau, and Jason Weston. Learning end-to-end goal-oriented dialog, 2017.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[3] Mihail Eric, Rahul Goel, Shachi Paul, Adarsh Kumar, Abhishek Sethi, Peter Ku, Anuj Kumar Goyal, Sanchit Agarwal, Shuyang Gao, and Dilek Hakkani-Tur. Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines, 2019.

[4] Matthew Henderson, Ivan Vulić, Daniela Gerz, Iñigo Casanueva, Paweł Budzianowski, Sam Coope, Georgios Spithourakis, Tsung-Hsien Wen, Nikola Mrkšić, and Pei-Hao Su. Training neural response selection for task-oriented dialogue systems, 2019.

[5] Shikib Mehri, Evgeniia Razumovskaia, Tiancheng Zhao, and Maxine Eskenazi. Pretraining methods for dialog context representation learning, 2019.

[6] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[8] Vladimir Vlasov, Akela Drissner-Schmid, and Alan Nichol. Few-shot generalization across dialogue tasks, 2018.

[9] Vladimir Vlasov, Johannes E. M. Mosig, and Alan Nichol. Dialogue transformers, 2020.

[10] Xuesong Yang, Yun-Nung Chen, Dilek Hakkani-Tur, Paul Crook, Xiujun Li, Jianfeng Gao, and Li Deng. End-to-end joint learning of natural language understanding and dialogue manager, 2017.