Bootstrapping Dialog Models from Human to Human Conversation Logs

Pankaj Dhoolia, Vineet Kumar, Danish Contractor, Sachindra Joshi
IBM Research AI
Vasant Kunj
New Delhi, India
{pdhoolia,vineeku6,dcontrac,jsachind}@in.ibm.com

Abstract

State-of-the-art commercial dialog platforms provide powerful tools to build a conversational agent. These platforms provide complete control to the *dialog designer* to model user-agent interactions. However, a dialog designer needs to rely on domain experts to manually build the dialog model – by creating dialog flow nodes and modeling user intents. This process is laborious, time consuming and expensive and does not allow the designer to exploit human to human conversation logs effectively. In this work, we present a research prototype that can ingest human-to-human conversation logs between an end-user and an agent, and suggest user-intents and agent-responses, given a conversation context. We utilize human to human conversation logs to build two emulators: user and agent. An agent emulator models an agent response given the conversation context so far, and a user emulator outputs possible user responses. Our system is able to recommend *conversational intents* as well as *conversation flow* using emulators based on real-world data, thus making the process of designing a bot more efficient. To the best our knowledge this is the first system that enables data-driven dialog model creation by emulating users and agents.

Introduction

Real-world deployments of dialog systems, use dialog-authoring frameworks such as IBM Watson® Assistant[1], DialogFlow[2] and Microsoft® Bot framework[3]. These frameworks enable the *Dialog Designer* to specify *user intents* and a *dialog flow*. Conversation models used by these frameworks may be thought of as a tree of dialog nodes, where each node expects a user intent as an input, generates an agent *response* as an output, and updates the dialog *state*.

In this paper, we present a research prototype that bootstraps dialog models using human-to-human conversation logs. Manual specification of user intents and dialog flow is time consuming and expensive. Our prototype offers a method to ease this effort by using human-to-human conversation logs between two participants: an end-user and support agent effectively. We utilize the advances in deep learning to cluster semantically similar sentences using sentence vectors (Cer et al. 2018) to help a dialog designer build a conversation model with user-intents and dialog flow. Our system ingests logs of human-to-human conversations between an end-user and an agent and starts by suggesting user-intents to the Dialog Designer by triggering a *user emulator*, which returns possible user responses given what has been said so far. The designer may then browse through the user-intent suggestions, include a few in the model, and explore them further to build the dialog-flow. Further exploration of intents triggers an *agent emulator*, which models an agent response. We further add a dialog node to the dialog flow with suggestions for possible agent-responses towards the user-intents. To the best our knowledge this is the first system that enables data-driven dialog model creation by emulating users and agents.

We demonstrate our prototype by integrating it with *IBM Watson® Assistant*. Our prototype system offers the following advantages over existing dialog frameworks: (i) coverage and prioritization of conversational intents can be grounded in real data, (ii) examples of intents represent real user expression and are easy to surface, (iii) the created dialog flow better represents real conversational variations.

System Design

The main features of our system are:

- **Suggest user intents**: Our system can ingest human-to-human conversation logs to suggest user-intents, conditioned on the conversation context.
- **Suggest agent responses**: Exploit conversation logs to suggest agent-responses given the previous conversation context and a selected user-intent.
- **Data-driven Dialog Model Creation**: Our system thus enables a dialog designer to interactively explore human-to-human conversation logs to create a data-driven dialog model quickly.

Implementation

We plugged our dialog-model bootstrapping system into a customized instance of IBM Watson® Assistant to facilitate the process of building a chatbot using user-agent conversation logs. A chatbot is intended to emulate a human

[1]https://www.ibm.com/cloud/watson-assistant

[2]https://dialogflow.com/

[3]https://dev.botframework.com that allow manual specification of dialog models

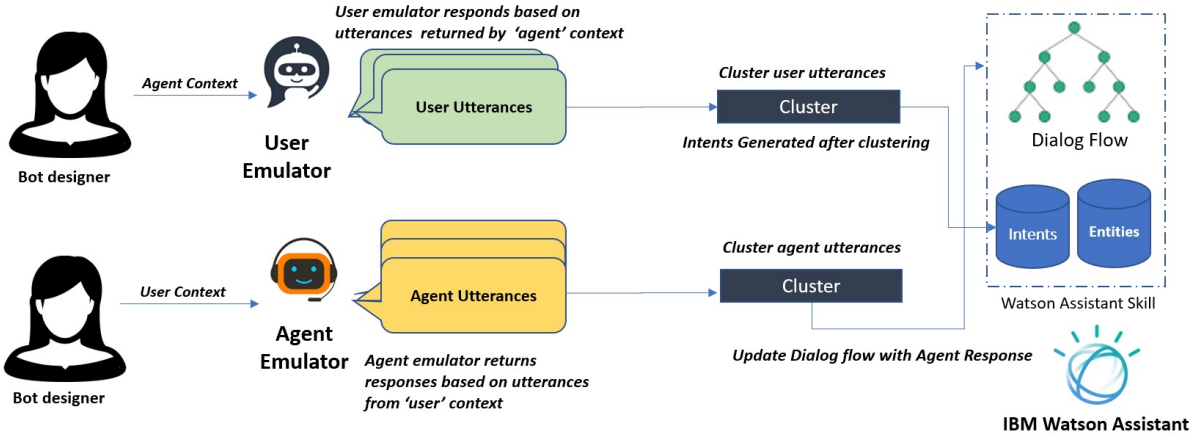

Figure 1: Dialog Design Runtime: User Intents and agent responses surfaced from conversation logs for review by the bot designer.

agent in responding to user utterances in a conversation context. we represent conversation contexts as neural vectors that are generated by pre-trained sentence encoders such as (Cer et al. 2018).

For each training context corresponding to the response universe U, we use a sentence encoder to generate a context vector. We index each context vector using the open source indexing library Annoy (Bernhardsson 2013) which is optimized for vector based nearest neighbour search. To emulate the dialog behavior, for an unseen conversation context C, sentence encoder is used to generate a query vector q, which is then used to search for the closest matching context vectors indexed earlier. Corresponding responses are returned as probable next utterances.

The Intent Generator could be seen as a function that consumes a list of user utterances and returns a set of intents. An intent is a group of similar utterances. In order to group utterances, we need to convert each sentence into a feature vector such that semantically similar sentences have similar feature vectors. We use the Universal Sentence Encoder (USE) (Cer et al. 2018) to obtain sentence vectors. We then cluster these sentence vectors using k-means clustering to generate intents. Since USE generates a single vector irrespective of the length of the sentence, sentence vectors degrade in quality as the sentence length increases (Cer et al. 2018). We thus, observe that processing sentences in order of sentence length (shortest sentence first) generates better clusters. Finally, given that the screen space is limited and we would like to reduce the cognitive load for dialog designers, we use clustering along with determinantal point process to get dominant and diverse suggestions for user and agent responses.

User Experience

Conversation Logs: In order to bootstrap a dialog workspace from human-to-human conversation logs, the system exposes a service endpoint to upload conversation/chat logs. Specifically, the service expects conversation logs in a CSV or JSON format with speaker-role annotations, as well as dialog boundary markers. The system then generates context-response pairs for both the 'user' and the 'agent' in the conversation logs.

Intent Suggestions: Once the conversation logs are uploaded, the system automatically processes them to extract initial *user-intents*. These intents are derived using examples from the user utterances at the first *turn* of the conversation. The easy-to-use interface allows the designer to browse intents along with example utterances from the conversation logs, corresponding to each of those intents. A designer may choose to import a selection of intents (along with related selections of examples) into the conversation model.

Dialog Flow: On identifying an intent of interest, the bot designer may choose to *explore* how an 'agent' would respond. The exploration step creates a dialog node for the intent and presents candidate agent-responses for the user-intent from the conversation logs. Once the designer selects a suitable agent-response (along with some linguistic variants), the modelling of one conversational exchange is complete. The designer may now choose to explore subsequent conversational exchanges by repeating the process of intents and agent-response selection from the context of this dialog node, leading to the construction of a dialog tree representing the conversational possibilities as demonstrated in the chat logs. The Dialog Designer may remove, reorder, and edit the suggested responses.

A video demonstrating our bootstrapping is system is available at: http://ibm.biz/BootstrappingDemo.

References

Bernhardsson, E. 2013. ANNOY: Approximate nearest neighbors in C++/Python optimized for memory usage and loading/saving to disk, 2013. *URL https://github.com/spotify/annoy* .

Cer, D.; Yang, Y.; yi Kong, S.; Hua, N.; Limtiaco, N.; John, R. S.; Constant, N.; Guajardo-Cespedes, M.; Yuan, S.; Tar, C.; Sung, Y.-H.; Strope, B.; and Kurzweil, R. 2018. Universal Sentence Encoder. *CoRR* abs/1803.11175.