Abstract

In writing programs that operate on natural language input, a common task is to recognize a specific utterance and extract specific portions of that utterance. We describe a novel method which learns how to match a natural language template to the utterance, while extracting information from the utterance in the process. We also describe our progress in implementing this method, as well as details about how we are training and evaluating the method.

1 Introduction

The purpose of this paper is to describe in more detail a novel method for processing user utterances in interactive natural-language systems. As covered in the previous survey report, our method attempts to combine the ease of use and generalization of information retrieval techniques with the ability to extract information from the utterance.

2 Design Constraints

Our method consists of training a deep neural network model, which takes in two inputs, a template and an utterance. The model attempts to match the utterance to the template, and returns a confidence in the match as well as the extracted values of each of the slots in the template.

For example, consider the example of an automated customer service kiosk at a grocery store, which can directly provide information to customers and also serve as a hub for other services. At a high level, the main logic of this system needs to be able to classify user utterances into different categories (e.g. an inquiry about product locations vs. an inquiry about store hours), and then dispatch to a relevant subroutine, most likely using some aspect of the user’s utterance. For example, the user might provide the utterance “Can you please tell me where the paper towels are?”. The system needs to recognize this as a request for an item location. Using our method, it does this by attempting to match the utterance against all of its templates. For example, the system might contain the template Where are the <item>?, which should match the user’s utterance with high confidence. This high confidence will indicate the rest of the system that this is likely a query about a product location. In the process, our method will match the slot <item> to the substring “paper towels.” The product location subroutine can then use this substring to look up the location of the desired product. That location can then be provided to the user, completing the interaction loop.

This example demonstrates two key design constraints for our method. The first design constraint is that it must handle non-trivial mappings between the utterance and the template. In this case, the input substring “Can you please tell me” needs to be ignored, and the word “are” needs to be mapped to the end of the sentence. This type of generalization has traditionally been achieved using the bag-of-words model (Harris). However, our model must also maintain enough structure in its mapping to locate the slot’s value in the utterance, which the bag-of-words model fundamentally cannot do. The second design constraint is that each slot’s value must be a portion of the input utterance. Modelling the mapping between the utterance and the template using attention, in a way similar to the Transformer model (Vaswani et al., 2017) can fulfills both of these design constraints well. Unlike recurrent neural networks, which have a very strong bias towards matching the precise ordering of sequences, Transformers use positional encodings, which provide a weak bias towards exact order matching, allow the model to conditionally ignore ordering
3 Implementation and Current Progress

We have begun implementing our method by modifying the Transformer implementation provided in (Klein et al., 2017). We chose this implementation because it is very easy to understand, and is a faithful implementation of the architecture described in (Vaswani et al., 2017).

We have made a few major changes to the model. First, we have replaced the learned embedding encoder with ELMo (Peters et al., 2018). This allows us to generalize more effectively with less training data. We chose this language model over other recent ones since it includes a character level embedding model, allowing our implementation to directly handle vocabulary not seen when training the language model. Another major change is that we have removed the decoder. Since the result our method uses is extracted from the model’s attention vector, we do not need to ever decode any latent embedding into text. Instead, we encode both the template and the utterance, and perform our mapping in the embedding space.

The current status of our implementation is that we have begun this modification. We have not yet begun training any initial versions. However, we do not foresee any major show-stopping issues.

4 Training Method

We demonstrate how our method can be trained using the “AskUbuntu” dataset (Lei et al., 2016). The “AskUbuntu” dataset is derived from questions asked on askubuntu.com. Of particular interest to our design is that these questions have already been labelled as similar by the users of the website. We can leverage these similar questions to train our method, by using one of the question titles to generate a template, and training the model to extract the same slots when that template is matched against the titles of the similar questions.

For example, consider the question “how to boot wubi installed ubuntu within windows 8?”. We can transform this question into a template by replacing the uncommon words found in similar questions with slots. The resulting template would be: how to boot wubi installed <slot1> within <slot2> 8?. We then match this template against the similar question titles “installing ubuntu besides windows 8” and “installing ubuntu 13.10 within windows”. We then introduce a loss between the extracted slot values from these titles and the expected slot values (“ubuntu” and “windows” in this case). We then use Stochastic Gradient Descent on this loss to train our model.

Note that although we are demonstrating this training method on the “AskUbuntu” dataset, any paraphrasing source can be used in this general training procedure. Optimally, this method would be trained on a corpus of utterances which match the domain that the templates will be used in. We intend to experiment with paraphrasing extracted from News Tweets, as in (Shwartz et al., 2017).
5 Evaluation

Since our problem setting is novel, there do not exist any prior benchmarks for measuring the performance of our method. We are intend to demonstrate the effectiveness of our methods in two ways.

First, we intend to show examples of held-out question pairs where the method is able to generalize, as well as failure cases where dissimilar questions are erroneously matched together. We also intend to show quantitative analysis of how frequently these cases occur. We also intend to use this method to analyze the effect of the training set on the method.

Secondly, we intend to build a natural-language interface to the Unix shell using our method. This will serve as a demonstration of how our method can be used to build a practical system. We intend to present example transcripts showing how the system can interpret commands using a wide range of syntactic forms. We have chosen this demonstration because it matches the domain of our training set, is sufficiently structured to be practical to implement a system for, and is a problem which clearly demonstrates the advantages of our method. In particular, there do not exist large datasets for controlling the Unix shell using natural language, and the ability to extract slots from the user’s utterances is essential to building a system of this nature.

References


