the morning paper

a random walk through Computer Science research, by Adrian Colyer

Made delightfully fast by

Machine learning for dialog state tracking: a review

JULY 6, 2016 ~ ADRIAN COLYER

Machine learning for dialog state tracking: a review Henderson *MLSLP 2015*

Today we turn our attention to the task of figuring out, potentially over multiple interactions with a bot, what it is the user is requesting the bot to do. This task goes by the name of *Dialog State Tracking*, and it's something that Matthew Henderson, today's paper author, knows quite a bit about. Henderson was the lead organiser for the second and third Dialog State Tracking Challenges (DSTC). DSTC is for spoken dialog systems, but if you're building a chatbot you have a very similar problem, albeit without the added ambiguity of trying to figure out what the user actually said from the spoken audio.

> Efficient operation of a spoken dialog system requires a component tha can track what has happened in a dialog, incorporating system outputs, user speech, context from previous turns, and other external information. The output of this Dialog State Tracking (DST) component is then used by the *dialog policy* to decide what action the system should take next, and so DST is essential for the final performance of a complete system.

Given what we've been learning over the last couple of weeks, you won't be at all surprised to hear that following many years of building systems using hand-crafted rules and generative approaches, machine-learned methods came top in all metrics in the second DSTC. Like every area of research, the DST problem has its own terminology. Dialog systems that help a user complete a task are framed in terms of *slots*.

> The slots, and possible values of a slot-based dialog system specify its domain, i.e. the scope of what it can talk about and the tasks that it can help the user

> complete. The slots inform the set of possible actions the system can take, the possible semantics of the user utterances, and the possible dialog states.

- *Informable* slots are slots where the user may provide a value (perhaps to constrain a search for example)
- *Requestable* slots are attributes which the user may ask the value of.
- The *dialog state* comprises everything that is used when the system makes its decision about what to say next.

> Early spoken dialog systems used hand-crafted rules for DST, keeping a single top hypothesis for each component of the dialog state, often with corresponding confidence scores. Such systems require no data to implement, and provide an accessible method for system designers to incorporate knowledge of the dialog domain.

The drawback with hand-crafted rules is that they don't account for uncertainty in a principled way. The need to maintain probability distributions over what the participants of a dialog desire has been argued for since 1996 (Pulman).

> The current dominant DST methods for estimating distributions of multiple competing dialog states can be split into two categories – generative models that jointly model both the inputs (typically the SLU hypotheses) and the dialog state, and discriminative models that directly capture the conditional probability over dialog states, having observed the dialog up to a certain point.

(SLU = Spoken Language Understanding).

Generative approaches use Bayesian dynamic networks to model a dialog, with the true dialog state and the true user action treated as unobserved random variables. Bayesian inference is used to update the probability distribution over dialog states given the system's action and a noisy observation of the true user action.

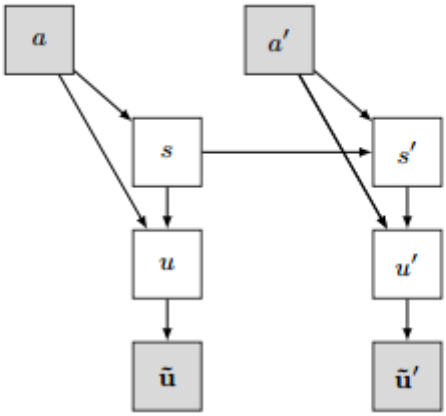

Fig. 2: Dynamic Bayesian network structure for DST, showing two consecutive time steps (dialog turns). A prime ($'$) denotes a variable in the following time step. The true dialog state s is dependent on the previous state and the machine action a. The true user's action u depends on both s and a, and a noisy observation of this is given by $\tilde{\mathbf{u}}$, whose distribution is inferred from the SLU. Observed nodes are shaded grey. The connections in the graph encode the conditional independence relationships in the joint distribution of the random variables.

Since there can be a huge number of possible dialog states (given the slots and all their possible values), approximations to the full distribution are typically used, either maintaining a beam of candidate dialog states, or assuming conditional independence between components. Generative approaches also cannot easily exploit arbitrary but potentially useful features of the dialog history.

> Discriminative models address these issues by directly modeling the distribution over the dialog state given arbitrary and possibly correlated input features [29]. While the parameters of generative models are typically hand-crafted, the model parameters for discriminative approaches are tuned using machine learning and labeled dialog data.

There are two main families of discriminative DSTs: *static classifiers* and *sequence models*.

Static classifiers

Static classifiers learn a probability distribution for the current state based on the sequence of observations so far. They model the sequential nature of the input by using *feature functions* to summarise the sequence. "One key issue with this approach is that it is necessary to extract a fixed dimensional feature vector to summarise a sequence of arbitrary length observations." (Perhaps giving the network a memory might help here?). Static classifiers have been built using maximum entropy linear classifiers, neural networks, and web-style ranking models. To

get good accuracy, it has proven necessary to include value-specific feature representations. Metallinou et al., for example, extract a feature vector f from an observation, and a set of vectors f_v for each v that has appeared in the SLU hypothesis so far.

> The f_v features are intended to convey information about the correctness of the v hypothesis for the goal constraint on slot s.

Sequence models

Sequence models inherently model the sequential nature of the problem (and we've seen a lot of them used in this series so far).

> Recurrent Neural Networks (RNNs) have been proposed as sequential models that are able to deal with high dimensional continuous input features. Notably, such models can be applied to operating directly on the distribution over sentences from the ASR, without requiring SLU and instead using weighted n-gram features from the ASR N-best list. This is termed word-based DST, and has two key benefits: first, it removes the need for feature design, and the risk of omitting an important feature, which can degrade performance unexpectedly. Second, it avoids the engineering task of building a separate SLU model.

LSTMs have also been used, modeling both the sequence of turns, and the sequence of words within a turn.

Tips and tricks

- Delexicalizing (is that really a word???) helps with generalization to unseen dialog states. The idea is to introduce special 'slot-name' and 'slot-value' generic tokens, and turn a sentence such as "I'm looking for a red car" into "I'm looking for a *slot-value slot-name*."
- Delexicalized features for RNN word-based tracking can also be used to share training data across disparate domains, improving performance in each individual domain. (Using if you don't have a lot of training data for your target domain yet – more on this tomorrow).
- RNNs for tracking individual slots can be combined using a softmax combination layer. "The basic idea is to order the slots, and add a recurrent layer that selects an assignment for each slot, given the assignments selected so far."

> Discriminative machine-learned methods are now the state-of-the-art in DST, as demonstrated in the offline corpus-based evaluations of the DSTCs, and in on-line trials with live users.

The paper contains a great set of references for studying some of the techniques used to build DST systems, so I encourage you to check some of them out if you are interested.

POSTED IN UNCATEGORIZED

MACHINE LEARNING

< PREVIOUS
End-to-end learning of semantic role labeling using recurrent neural networks

NEXT >
Multi-domain dialog state tracking using recurrent neural networks