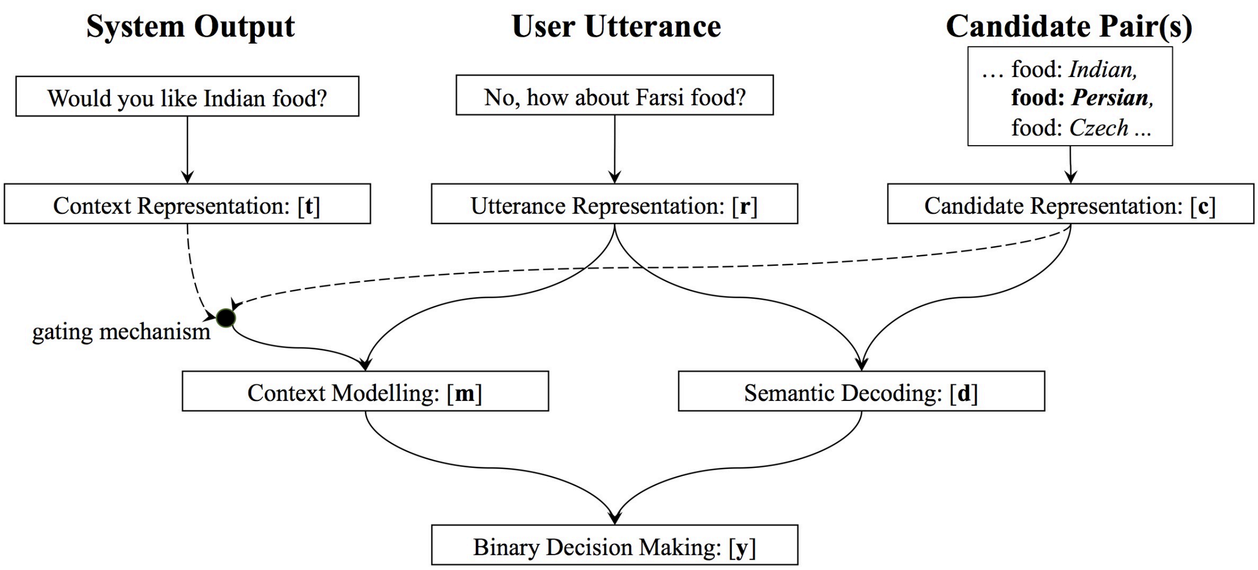
This article is a note from the paper [Neural Belief Tracker- Data-Driven Dialogue State Tracking](). In this article, the authors combine NLU and DST as a whole, which takes merely the results of Automated Speech Recognition and context as input to determine which slot-value pair (s, v) should be added to the dialog state.

The structure of their model could be seperated as three parts: representation learning, semantic decoding and context modeling. As the picture shows, it first encodes system output, user utterance and candidate pairs into vectors t, r, c, and then compute the match score between c and r, at the same time computing the match score between t and r to pay attention to **confirmation replies**.



Representation Learning

Two modules are utilized to learn the representation for three parts. For **NBT-DNN** part, **the sum of dense n-gram representations** are calculated and transformed non-linearly into the final hidden vector r. Representation of slots and values are merely parameter vectors, denoted as c_v and c_s. For the context, it comprises of three parts: utterance t_q, slot t_s and value t_v. Note that the latter two items might be zero, except the last system response is a **confirmation** like "Do you want Thai food?".

Semantic Decoding

This module measures the match score between r and c. If the utterance of a user simply gives slot-value information like "I want *cheap restaurants*.", the score could be directly used to determine the most probable c_s and c_v he mentions. Another thing to notice is that this core is NOT a **scalar** but a vector, because this score does not represent all probabilities (for t_u to be a confirmation, the response might be merely "Yes."). Here is the formula:

$$\mathbf{c} = \sigma(W_c^s(\mathbf{c}_s + \mathbf{c}_v) + b_c^s)$$
$$\mathbf{d} = \mathbf{r} \circ \mathbf{c}$$

Context Modeling

Remember the existance of confirmation. For such situations the new information should be found from the previous system response, and the match score should come mainly from the measurement between t_s ,t_v and c_s, c_v. Also, sometimes system will ask the user his preference about a certain **slot**. So a match score between t_q and c_s could also be computed. The resulting formula is:

$$\mathbf{m}_r = (c_s \cdot t_q)\mathbf{r}$$
$$\mathbf{m}_c = (c_s \cdot t_s)(c_v \cdot t_v)\mathbf{r}$$

Decision Making

The decision is straightforward: simply putting three score vectors into a linear layer and get a length-2 vector representing the probability distribution over positive and negative.

$$\mathbf{y} = \phi_2(\phi_{100}(\mathbf{m}_c) + \phi_{100}(\mathbf{m}_r) + \phi_{100}(\mathbf{d}))$$

Belief State Update Mechanism

Our final purpose is to generate an overall state distribution over all possible states. The dicision vector $\mathbf{y}$ only represents the single-step decision. We need to get a global state distribution. Formally $p(s, v|h^t, sys^{t-1})$, where t denotes the t-th step, h^t the N-best ASR hypotheses and sys^{t-1} the preceding system output. This could be factorized by weight of h_i^t:

$$p(s, v|h^t, sys^{t-1}) = \sum_{i=1}^{N} p(s, v|h_i^t, sys^{t-1})$$

We want $p(s, v|h^{1:t}, sys^{1:t-1})$, which could be simply viewed as "keep and update", controled by a parameter λ:

$$p(s, v|h^{1:t}, sys^{1:t-1}) = \lambda p(s, v|h^{1:t-1}, sys^{1:t-2}) + (1-\lambda)p(s, v|h^t, sys^{t-1})$$

And the activated goals are selected as follows:

$$V_s^t = \{v \in V_s | p(v, s|h^{1:t}, sys^{1:t-1}) \geq 0.5\}$$

with requested selected as:

$$W = \{s | \exists v \in V(p(v, s|h^{1:t}, sys^{1:t-1}) \geq 0.5)\}$$