5. Dialogue State Tracking

Ondřej Dušek & Vojtěch Hudeček

http://ufal.cz/npfl099

31. 10. 2019
Dialogue State Tracking

• Dialogue management consist of:
  • State update ← here we need DST
  • Action selection (later)

• Dialogue State needed to remember what was said in the past
  • tracking the dialogue progress
  • summary of the whole dialogue history
  • basis for action selection decisions

U: I’m looking for a restaurant in the city centre.
S: OK, what kind of food do you like?
U: Chinese.
× S: What part of town do you have in mind?
× S: Sure, the Golden Dragon is a good Chinese restaurant. It is located in the west part of town.
✓ S: Sure, the Golden Dragon is a good Chinese restaurant. It is located in the city centre.
Dialogue State Contents

• “All that is used when the system decides what to say next” (Henderson, 2015)

• User goal/preferences ~ NLU output
  • slots & values provided (search constraints)
  • information requested

• Past system actions
  • information provided
    • slots and values
    • list of venues offered
  • slots confirmed
  • slots requested

• Other semantic context
  • user/system utterance: bye, thank you, repeat, restart etc.

U: Give me the address of the first one you talked about.
U: Is there any other place in this area?
S: OK, Chinese food. […]
S: What time would you like to leave?
Problems with Dialogue State

• NLU is unreliable
  • takes unreliable ASR output
  • makes mistakes by itself – some utterances are ambiguous
  • output might conflict with ontology

• Possible solutions:
  • detect contradictions, ask for confirmation
  • ignore low-confidence NLU input
    • what’s “low”?
    • what if we ignore 10x the same thing?

• Better solution: make the state probabilistic – belief state
Belief State

• Assume we don’t know the true current dialogue state $s_t$
  • states (what the user wants) influence observations $o_t$ (what the system hears)
  • based on observations $o_t$ & system actions $a_t$, we can estimate a probability distribution $b(s)$ over all possible states – belief state

• More robust than using dialogue state directly
  • accumulates probability mass over multiple turns
    • low confidence – if the user repeats it, we get it the 2nd time
    • accumulates probability over NLU n-best lists

• Plays well with probabilistic dialogue policies (POMDPs)
  • but not only them – rule-based, too
Belief State

1. I'm looking for a Thai restaurant.
   - **observations**
     - `hello(type=restaurant)` 0.6
     - `inform(type=restaurant, food=Thai)` 0.4

2. Thai.
   - **observations**
     - `hello()` 0.5
     - `inform(food=Turkish)` 0.3
     - `inform(food=Thai)` 0.2

   **belief states**
   - `R` 0.6
   - `O` 0.4
   - `type food`

   **actions**
   - You are looking for a restaurant right?
   - `R O` 0.6
   - `O TH` 0.4
   - `type food`

   **belief states**
   - `R` 1
   - `O` 0.6
   - `type food`

   **actions**
   - What kind of food would you like?
   - `R` 0.6
   - `O TH` 0.4
   - `type food`

   **belief states**
   - `R` 0.5
   - `O` 0.5
   - `food`

   **actions**
   - What kind of food would you like?
   - `R` 0.5
   - `O TR TH` 0.3
   - `type food`

   **belief states**
   - `R` 0.8
   - `O 0.2 TH TR O` 0.4
   - `type food`

   **actions**
   - Did you say Thai or Turkish?
   - `R` 0.8
   - `O 0.2 TR TH` 0.4
   - `type food`

   **this is what we need**
   - (=belief state)

(from Milica Gašić's slides)
Basic Discriminative Belief Tracker

- **Partition the state** by assuming conditional independence
  - simplify – assume each slot is independent:
    - state $s = [s^1, \ldots s^N]$, belief $b(s_t) = \prod_i b(s^i_t)$

- **Always trust the NLU**
  - this makes the model parameter-free
  - …and basically rule-based
  - but very fast, with reasonable performance

**Update rule**

$$b(s^i_t) = \sum_{s^i_{t-1}, o^i_t} p(s^i_t|a^i_{t-1}, s^i_{t-1}, o^i_t)b(s^i_{t-1})$$

**Substitution**

$$b(s^i_t) = \begin{cases} 
    p(s^i_t = \text{kdir})p(o^i_t = \text{null}) & \text{if } s^i_t = \text{kdir} \\
    p(o^i_t = s^i_t) + p(o^i_t = \text{null})p(s^i_t = s^i_{t-1}) & \text{otherwise} 
\end{cases}$$

**Belief state update rule**

$$p(o^i_t) = \begin{cases} 
    p(o^i_t) & \text{if } s^i_t = o^i_t \land o^i_t \neq \text{null} \\
    p(o^i_t) & \text{if } s^i_t = s^i_{t-1} \land o^i_t = \text{null} \\
    0 & \text{otherwise} 
\end{cases}$$

- user silent about slot $i$
Basic Feed-forward Tracker

• a simple feed-forward network
  • input – features (w.r.t. slot-value $\nu$ & time $t$)
    • SLU score of $\nu$
    • n-best rank of $\nu$
    • user & system act type
    • … – domain-independent, low-level NLU outputs
  • 3 tanh layers
  • output – softmax (= probability distribution over values)
• **static** – does not model dialogue as a sequence
  • uses a sliding window – current time $t$ + few steps back + $\sum$ previous
Basic RNN Tracker

• plain sigmoid RNN with a memory vector
  • not quite LSTM/GRU, but close
  • memory updated separately, used in belief update
• does not need NLU
  • turn features = lexicalized + delexicalized n-grams from ASR n-best list, weighted by confidence
• delexicalization very harsh: <slot> <value>
  • you don’t even know which slot it is
  • this apparently somewhat helps the system generalize across domains
• dynamic – explicitly models dialogue as sequence
  • using the network recurrence
Neural/Rule Hybrid

- explicit update over belief
  - per-slot model (separate for each slot)
  - simple update rule $R$
    - for a value: add $a \cdot$ current NLU confidence, normalize
    - differentiable, can be trained end-to-end
  - trained models $F, G$ provide $a$
    - $F$ is generic LSTM, $G$ is value specific feed-forward

- needs an NLU, but postprocesses it
  - input & output of tracker NLU step
    - = prob. dist. of informs over slot values in current turn
  - generic & specific part again

(Vodolán et al., 2017)
http://arxiv.org/abs/1702.06336

$a =$ “transition coefficients”
(control how much probability mass is moved)

this part is mostly for overriding frequent ASR errors

n-grams from ASR n-best + prev. system DAs

delex. ASR n-grams

base NLU output (prob. dist. of informs over slot values)

belief (prob. dist. over values)

differentiable update rule

$F$ is generic LSTM, $G$ is value specific feed-forward

this can generalize creates scores with LSTM
- good for null value

feed-forward only
Incremental Recurrent Tracker

• simple: LSTM over words + classification on hidden states
  • runs over the whole dialogue history (user utterances + system actions)
  • classification can occur after each word, right as it comes in from ASR

• also dynamic/sequential

• also doesn’t use any NLU
  • infrequent values are delexicalized (otherwise it can’t learn them)

• slightly worse performance – possible causes:
  • only uses ASR 1-best
  • long recurrences (no hierarchy)

(Žilka & Jurčíček, 2015)
https://dl.acm.org/citation.cfm?id=2955040
http://arxiv.org/abs/1507.03471
\[ m_r = (c_s \cdot t_q) r \]
\[ m_c = (c_s \cdot t_s)(c_v \cdot t_v) r \]

\[ c = \sigma(W_c^s(c_s + c_v) + b_c^s) \]
\[ d = r \odot c \]

\[ y = \phi_2(\phi_{100}(d) + \phi_{100}(m_r) + \phi_{100}(m_c)) \]
NBT

- Better use of data: Getting rid of delexicalization
  - pre-trained word vectors – important!
  - shared parameters
  - RNN/CNN feature extractors
- Discriminative learning slot values

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

<table>
<thead>
<tr>
<th>DST Model</th>
<th>DSTC2 Goals</th>
<th>DSTC2 Requests</th>
<th>WOZ 2.0 Goals</th>
<th>WOZ 2.0 Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delexicalisation-Based Model</td>
<td>69.1</td>
<td>95.7</td>
<td>70.8</td>
<td>87.1</td>
</tr>
<tr>
<td>Delexicalisation-Based Model + Semantic Dictionary</td>
<td>72.9*</td>
<td>95.7</td>
<td>83.7*</td>
<td>87.6</td>
</tr>
<tr>
<td>Neural Belief Tracker: NBT-DNN</td>
<td>72.6*</td>
<td>96.4</td>
<td>84.4*</td>
<td>91.2*</td>
</tr>
<tr>
<td>Neural Belief Tracker: NBT-CNN</td>
<td>73.4*</td>
<td>96.5</td>
<td>84.2*</td>
<td>91.6*</td>
</tr>
</tbody>
</table>

(Mrkšić et al., 2017)
https://www.aclweb.org/anthology/P17-1163
Candidate Ranking (Rastogi et al., 2017)
https://arxiv.org/abs/1712.10224

• Previous systems consider all values for each slot
  • this is a problem for open-ended slots (e.g. restaurant name)
  • enumerating over all takes ages, some are previously unseen

• Alternative: always consider just $K$ candidates
  • use last $K$ candidates from system actions and NLU output
    • NB: only way history is incorporated here!
  • select from them using a per-slot softmax

representation of i-th candidate: utterance/slot/candidate features (next slide)
Candidate Ranking – representation

• Using BiGRU over lexicalized & delexicalized utterance

• Features:
  • **utterance** – last GRU state + indicators for non-slot DAs (user & prev. system)
  • **slot** – indicators for DAs with this slot (user & prev. system) + last turn scores for *null* & *dontcare*
  • **candidate** – GRU states over matched value words + indicators for DAs with this slot & value (user & prev. system)
Multi-value Candidate Ranking

• What if multiple values are true?
  • previous approach picks one (softmax)
  • use set of binary classifiers (log loss) instead
• + more flexible regarding candidates
  • can be past $k$ from NLU, but also just current ASR n-grams
  • this model keeps track of context by itself

(Goel et al., 2018)
http://arxiv.org/abs/1811.12891
Hybrid Classify/Rank

- Ranking is faster & more flexible
- Classification over all values is more accurate
  - at least for most slots, where # of values is limited
- Solution: combine classification & ranking
  - choose best model for each slot based on dev data performance
- Ranking approach – multi-value from previous slide
- Classification approach – straightforward: hierarchical LSTM + per-slot feed-forward + softmax

**metric: joint goal accuracy**
- exact match on dialogue state (most probable value only)

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Baseline</td>
<td>1.5%</td>
</tr>
<tr>
<td>MultiWOZ-2.0 Benchmark</td>
<td>25.83%</td>
</tr>
<tr>
<td>Ranking only</td>
<td>31.11% (29.73%)</td>
</tr>
<tr>
<td>Classification only</td>
<td>40.74% (38.42%)</td>
</tr>
<tr>
<td>Hybrid</td>
<td>44.24% (42.33%)</td>
</tr>
</tbody>
</table>

- ensemble (majority vote of 3 models)
- single model

- separate for each slot

- shared across slots

NPFL099 L5 2019
Using BERT

- Very basic:
  - run BERT over previous system & current user utterance
  - from 1st token’s representation, get a decision: none/dontcare/span
    - per-slot (BERT is shared, but the final decision is slot-specific)
  - span = need to find a concrete value as a span somewhere in the text
    - predict start & end token of the span using 2 softmaxes
  - carry-over across multiple turns is rule-based:
    - if none is predicted, keep previous value, otherwise change it

(Chao & Lane, 2019)
http://arxiv.org/abs/1907.03040
Slot-Utterance Matching Belief Tracker

- different take on BERT trackers
  - inspired by reading comprehension
  - considers “domain – slot” a question & tries to find the value in the input utterance

- tracker over BERT
  - attention + turn-based RNN
    - attention in current utterance
    - RNN (LSTM/GRU) for carry-over of past values
  - layer normalization to match BERT outputs
    - BERT includes layer normalization by default
  - trained to match the correct values in the utterance
    - loss: distance of true value BERT encoding from the tracker output (Euclidean/Cosine)

(Lee et al., 2019)
http://arxiv.org/abs/1907.07421
Even More Reading Comprehension

- Also uses BERT, but not necessarily
  - works slightly worse with random-initialized word embeddings

- sequence of 3 decisions
  - do we carry over last turn’s prediction? (Yes/No)
  - if no: what kind of answer are we looking for? (yes/no/dontcare/span of text)
  - if span: predict span’s start and end

- 2 prediction softmaxes: 1 for span start, 1 for end
- input: whole dialogue, concatenated
- this can be BERT
- BiLSTM
- block diagram

(Gao et al., 2019)
https://www.aclweb.org/anthology/W19-5932/
Dialogue State as SQL

- User goal is a query → why not SQL query?
- Text-to-SQL models used for tracking
  - with contextual enhancements, input:
    - all user inputs so far
    - previous system response
    - database schema
- Seq2seq-based model example:
  - hierarchical LSTM for encoding user & system
  - database column embeddings
    - averaged embeddings over table + column name
  - decoder:
    - decide between SQL keyword vs. column
    - then select which keyword / column via softmax
- So far, experimental – performance is low

Summary

• State tracking is needed to maintain user goal over multiple turns
• Best to make the state probabilistic – belief state
• Architectures – many options
  • good NLU + rules – works well!
  • neural, hybrid
  • static (sliding-window) vs. dynamic (recurrent, modelling dialogue as sequence)
  • with/without NLU
• classifiers vs. candidate rankers vs. reading comprehension
  • classifiers are more accurate than rankers but slower, limited to seen values
  • reading comprehension is a very new approach, works nicely but probably slow
• BERT & co. as usual – good but slow
• incremental – not used too much so far
• Alternative/experimental: SQL instead of slots/values
Thanks

Contact us:
odusek@ufal.mff.cuni.cz
hudecek@ufal.mff.cuni.cz

Get these slides here:
http://ufal.cz/npfl099

References/Inspiration/Further:
• Milica Gašić’s slides (Cambridge University): http://mi.eng.cam.ac.uk/~mg436/teaching.html
  https://ai.google/research/pubs/pub44018