NPFL123 Dialogue Systems

5. Neural NLU & State Tracking

https://ufal.cz(npfl123

Ondřej Dušek, Vojtěch Hudeček, Tomáš Nekvinda
& Jan Cuřín, Petr Fousek

14. 3. 2022
Neural networks

• Can be used for both classification & sequence models
• **Non-linear functions**, composed of basic building blocks
  • stacked into **layers**
  • whole network ~ “pipeline”/“flow”
• Layers are built of **activation functions**:
  • linear functions
  • nonlinearities – sigmoid, tanh, ReLU
  • softmax – probability estimates:
    \[
    \text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}
    \]
• Fully differentiable – training by gradient descent
  • gradients **backpropagated** from outputs to all parameters
  • (composite function differentiation)

https://playground.tensorflow.org/ – look at the internals (very simple network)
Neural networks – features

• You can use the same ones as for LR/SVM…
  • but it’s a lot of work to code them in

• **Word embeddings**
  • let the network learn features by itself
    • input is just words (vocabulary is numbered)
      • top ~50k words + <unk>, or subwords
  • distributed word representation
    • each word = **vector of floats** (~50-2000 dims.)
  • part of network parameters – trained
    a) random initialization
    b) pretraining
  • the network learns which words are used similarly
    • they end up having close embedding values
    • different embeddings for different tasks

http://ruder.io/word-embeddings-2017/
Recurrent Neural Networks

- Many identical layers with shared parameters (cells)
  - ~ the same layer is applied multiple times, taking its own outputs as input
    - ~ same number of layers as there are tokens
    - output = hidden state – fed to the next step
  - additional input – next token features

- Cell types
  - **basic RNN**: linear + tanh
    - problem: vanishing gradients
    - can’t hold long recurrences
  - **GRU, LSTM**: more complex, to make backpropagation work better
    - “gates” to keep old values

https://medium.com/@saurabh.rathor092/simple-rnn-vs-gru-vs-lstm-difference-lies-in-more-flexible-control-5f33e07b1e57
Encoder-Decoder Networks

- Default RNN paradigm for sequences/structure prediction
  - **encoder** RNN: encodes the input token-by-token into hidden states $h_t$
    - next step: last hidden state + next token as input
  - **decoder** RNN: constructs the output token-by-token
    - initialized by last encoder hidden state
    - output: hidden state & softmax over output vocabulary + argmax
    - next step: last hidden state + last generated token as input
- LSTM/GRU cells over vectors of ~ embedding size
- MT, dialogue, parsing...
  - more complex structures linearized to sequences

$$h_0 = 0$$
$$h_t = \text{cell}(x_t, h_{t-1})$$

$$s_0 = h_T$$
$$p(y_t | y_1, ... y_{t-1}, x) = \text{softmax}(s_t)$$
$$s_t = \text{cell}(y_{t-1}, s_{t-1})$$

https://medium.com/syncedreview/a-brief-overview-of-attention-mechanism-13c578ba9129
Attention Models

• Encoder-decoder too crude for complex sequences
  • the whole input crammed into a fixed-size vector (last hidden state)

• **Attention** = “memory” of **all** encoder hidden states
  • weighted combination
  • re-weighted every decoder step
    → can focus on currently important part of input
  • fed into decoder inputs + decoder softmax layer

attention value = **context vector**

\[ c_t = \sum_{i=1}^{n} \alpha_{ti} h_i \]

t = decoder step
1 ... n = encoder steps

decoder state

attention weights = **alignment model**

\[ \alpha_{ti} = \text{softmax}(v_{\alpha} \cdot \tanh(W_{\alpha} \cdot s_{t-1} + U_{\alpha} \cdot h_i)) \]

• **Self-attention** – over previous decoder steps

https://skymind.ai/wiki/attention-mechanism-memory-network
Neural NLU

- Various architectures possible
- Classification
  - feed-forward NN
  - RNN + attention weight \( \rightarrow \) softmax
  - convolutional networks
  - Transformer
- Sequence tagging
  - RNN (LSTM/GRU) \( \rightarrow \) softmax over hidden states
    - default version: label bias (like MEMM)
    - CRF over the RNN possible
  - Still treats intent + slots independently

• Same network for both tasks

• **Bidirectional encoder**
  • 2 encoders: left-to-right, right-to-left
  • concatenate hidden states
  • “see the whole sentence before you start tagging”

• Decoder – tag word-by-word, inputs:
  a) attention
  b) input encoder hidden states (“aligned inputs”)
  c) both

• Intent classification:
  softmax over last encoder state
  • + specific intent context vector (attention)
NN NLU – Joint Intent & Slots

• Extended version: use slot tagging in intent classification
  • Bidi encoder
  • Slots decoder with encoder states & attention
  • Intent decoder – attention over slots decoder states

• Works slightly better

(Liu & Lane, 2016) http://arxiv.org/abs/1609.01454
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”
- **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.

(Henderson, 2015)
https://ai.google/research/pubs/pub44018
Ontology

- To describe possible states
- Defines all concepts in the system
  - List of slots
  - Possible range of values per slot
  - Possible actions per slot
    - requestable, informable etc.
  - Dependencies
    - some concepts only applicable for some values of parent concepts

food_type – only for type=restaurant
has_parking – only for type=hotel

“if entity=venue, then…”

entity = {venue, landmark}
venue.type = {restaurant, bar,…}

some slot names may need disambiguation
(venue type vs. landmark type)

(Young, 2009)
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

ASR: 0.5 I’m looking for an expensive hotel
0.5 I’m looking for inexpensive hotels

NLU: 0.3 inform(type=restaurant, stars=5)

only hotels have stars!
Belief State

• Assume we don’t know the true dialogue state
  • but we can estimate a probability distribution over all possible states
  • In practice: per-slot distributions

• More robust
  • 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
  • but not only them – rule-based, too
Belief State

(balance on Milica Gašić’s slides)

<table>
<thead>
<tr>
<th>turn</th>
<th>observations</th>
<th>NLU (no state over turns)</th>
<th>dialogue state (1-best)</th>
<th>belief state (probability distributions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>I want a Danish place in the center</td>
<td>inform(area=center) 0.6 area=center</td>
<td>What food would you like?</td>
<td>area=</td>
</tr>
<tr>
<td></td>
<td>inform(food=Danish) 0.4</td>
<td></td>
<td></td>
<td>center</td>
</tr>
<tr>
<td>2.</td>
<td>Danish</td>
<td>inform(food=Spanish) 0.5 food=Spanish</td>
<td>Which area do you prefer?</td>
<td>area=</td>
</tr>
<tr>
<td></td>
<td>inform(food=Danish) 0.4</td>
<td></td>
<td></td>
<td>center</td>
</tr>
</tbody>
</table>

15
Dialogue as a Markov Decision Process

• MDP = probabilistic control process
  • model – Dynamic Bayesian Network
    • random variables & dependencies in a graph/network
    • “dynamic” = structure repeats over each time step $t$
  • $s_t$ – dialogue states = what the user wants
  • $a_t$ – actions = what the system says
  • $r_t$ – rewards = measure of quality
    • typically slightly negative for each turn, high positive for successful finish
  • $p(s_{t+1}|s_t, a_t)$ – transition probabilities
• Markov property – state defines everything
• Problem: we’re not sure about the dialogue state

(from Milica Gašić’s slides)
• Dialogue states are **not observable**
  • modelled probabilistically – belief state $b(s)$ is a prob. distribution over states
  • states (what the user wants) influence observations $o_t$ (what the system hears)

• Still Markovian
  • $b'(s') = \frac{1}{Z} p(o|s') \sum_{s \in S} p(s'|s,a)b(s)$
  • $b(s)$ can be modelled by an HMM

![Diagram of a partially observable Markov decision process (POMDP)](from Filip Jurčiček’s slides)
Digression: Generative vs. Discriminative Models

What they learn:

• **Generative** – whole distribution $p(x, y)$
• **Discriminative** – just decision boundaries between classes $\sim p(y|x)$

To predict $p(y|x)$…

• **Generative models**
  • Assume some functional form for $p(y), p(x|y)$
  • Estimate parameters of $p(y), p(x|y)$ directly from training data
  • Use Bayes rule to calculate $p(y|x)$

• **Discriminative models**
  • Assume some functional form for $p(y|x)$
  • Estimate parameters of $p(y|x)$ directly from training data

they get the same thing, but in different ways
Generative vs. Discriminative Models

Example: elephants vs. dogs

• **Discriminative:**
  • establish decision boundary (~find distinctive features)
  • classification: just check on which side we are

• **Generative**
  • ~2 models – what elephants & dogs look like
  • classification: match against the two models

• Discriminative – typically better results
• Generative – might be more robust, more versatile
  • e.g. predicting the other way, actually generating likely \((x, y)’s\)
Naïve Generative Belief Tracking

• Using the HMM model
  • estimate the transition & observation probabilities from data
    \[ b(s) = \frac{1}{Z} p(o_t | s_t) \sum_{s_{t-1} \in S} p(s_t | a_{t-1}, s_{t-1}) b(s_{t-1}) \]
    - observation probability
    - transition probability
    - previous belief

• Problem: too many states
  • e.g. 10 slots, 10 values each → \(10^{10}\) distinct states – intractable

• Solutions:
  • only track stuff that appeared in NLU
  • only track \(n\) most probable (beam)
  • merge similar states
  • **partition the state** – assume slots are independent, use per-slot beliefs
    • state \(s = [s^1, ... s^N]\), belief \(b(s_t) = \prod_i b(s^i_t)\)
Generative BT: Parameter Tying

• Per-slot: $b(s_t^i) = \sum_{s_{t-1}, o_t^i} p(o_t^i | s_t^i) p(s_t^i | a_{t-1}^i, s_{t-1}^i) b(s_{t-1}^i)$

  observation probability  transition probability  previous belief

• Further simplification: **tie most parameters**
  • estimates from data are unreliable anyway → basically uses 2 parameters only 😊

  transition probabilities:
  
  $$p(s_t^i | a_{t-1}^i, s_{t-1}^i) = \begin{cases} 
  \theta_T & \text{if } s_t^i = s_{t-1}^i \\
  1 - \theta_T & \text{otherwise}
  \end{cases}$$

  $\theta_T = \text{“rigidity” (bias for keeping previous values), otherwise all value changes have the same probability}$

  observation probabilities:
  
  $$p(o_t^i | s_t^i) = \begin{cases} 
  \theta_O p(o_t^i) & \text{if } o_t^i = s_t^i \\
  1 - \theta_O & \text{otherwise}
  \end{cases}$$

  $\theta_O \sim \text{confidence in NLU}$
  
  $p(o_t^i) = \text{NLU output}$
  
  i.e. believe in value given by NLU with $\theta_O$, distribute rest of probability equally

(Žilka et al., 2013)

https://www.aclweb.org/anthology/W13-4070/
Basic Discriminative Belief Tracker

• Based on the previous model
  • same slot independence assumption
• Even simpler – “always trust the NLU”
  • this makes it parameter-free
  • …and kinda rule-based
  • but very fast, with reasonable performance

update rule: \[b(s_t^i) = \sum_{s_{t-1}^i, o_t^i} p(s_t^i | a_{t-1}^i, s_{t-1}^i, o_t^i)b(s_{t-1}^i)\]

\[b(s_t^i) = \begin{cases} 
    p(o_t^i) & \text{if } s_t^i = o_t^i \land o_t^i \neq \text{null} \\
    0 & \text{otherwise}
\end{cases}\]

1. “user mentioned this value”
2. “no change”
3. user silent about slot \(i\)

(Žilka et al., 2013)
https://www.aclweb.org/anthology/W13-4070/
Tracker types

• **Generative** trackers – need many assumptions to be tractable
  • cannot exploit arbitrary features
  • … or they can, but not if we want to keep them tractable
  • often use handcrafted parameters
  • … may produce unreliable estimates  

• **Discriminative** trackers – can use any features from dialogue history
  • parameters estimated from data more easily
  • generally used nowadays

• General distinction
  • **static models** – encode whole history into features
  • **dynamic/sequence models** – explicitly model dialogue as sequential
Static Discriminative Trackers

- Generally predict $p(s_t | o_1, a_1, ..., a_{t-1}, o_t)$
  - any kind of classifier (SVM, LR…)
  - need fixed feature vector from $o_1, a_1, ..., a_{t-1}, o_t$ (where $t$ is arbitrary)
    - current turn, cumulative, sliding window
    - per-value features & tying weights – some values are too rare

- Global feature examples:
  - NLU n-best size, entropy, lengths (current turn, cumulative)
  - ASR scores

- Per-value $ν$ examples:
  - rank & score of hypo with $ν$ on current NLU n-best + diff vs. top-scoring hypo
  - # times $ν$ appeared so far, sum/average confidence of that
  - # negations/confirmations of $ν$ so far
  - reliability of NLU predicting $ν$ on held-out data

(Metallinou et al., 2013) https://www.aclweb.org/anthology/P13-1046
Dynamic Discriminative Trackers

- Dialogue as a sequence $p(s_1, ..., s_t | o_1, ..., o_t)$
- **CRF** models
  - similar features as static
  - feature value: NLU score for the given thing (e.g. DA type + slot + value)
  - target: per-slot BIO coding

<table>
<thead>
<tr>
<th>Utterance</th>
<th>Goals</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>Persian</td>
<td>South</td>
</tr>
<tr>
<td>$U_1$</td>
<td>I need a Persian restaurant in the south part of town.</td>
<td></td>
</tr>
<tr>
<td>$S_2$</td>
<td>What kind of food would you like?</td>
<td></td>
</tr>
<tr>
<td>$U_2$</td>
<td>Persian.</td>
<td></td>
</tr>
<tr>
<td>$S_3$</td>
<td>I'm sorry but there is no restaurant serving persian food</td>
<td></td>
</tr>
<tr>
<td>$U_3$</td>
<td>How about Portuguese food?</td>
<td>Portuguese</td>
</tr>
<tr>
<td>$S_4$</td>
<td>Peking restaurant is a nice place in the south of town.</td>
<td></td>
</tr>
<tr>
<td>$U_4$</td>
<td>Is that Portuguese?</td>
<td>Portuguese</td>
</tr>
<tr>
<td>$S_5$</td>
<td>Nandos is a nice place in the south of town serving tasty Portuguese food.</td>
<td></td>
</tr>
<tr>
<td>$U_5$</td>
<td>Alright. What's the phone number?</td>
<td>Portuguese</td>
</tr>
<tr>
<td>$S_6$</td>
<td>The phone number of nandos is 01223 327908</td>
<td>Portuguese</td>
</tr>
<tr>
<td>$U_6$</td>
<td>And the address?</td>
<td>Portuguese</td>
</tr>
<tr>
<td>$S_7$</td>
<td>Start, nandos is on Cambridge Leisure Park Clifton Way.</td>
<td></td>
</tr>
</tbody>
</table>

(Kim & Banchs, 2014) [https://www.aclweb.org/anthology/W14-4345](https://www.aclweb.org/anthology/W14-4345)
Neural State Trackers

• discriminative, many architectures

• basic **static** example:
  use a **feed-forward** as your classifier
  • input – features (w.r.t. slot-value \( v \) & time \( t \))
    • NLU score of \( v \)
    • n-best rank of \( v \)
    • user & system intent (**inform/request**)
  • … – other domain-independent, low-level NLU features
• 3 tanh layers
• output – softmax
  (= probability distribution over values)
• static – uses a **sliding window**:
  current time \( t \) + few steps back + \( \sum \) previous

\( \text{softmax over all possible } v \text{'s + “other”} \)

(Henderson et al., 2013)
https://aclweb.org/anthology/W13-4073
Dynamic Neural State Trackers

• Based on RNNs (turn-level or word-level)
• Typically **not** using NLU – directly ASR/words → belief
• Simple example: RNN over words + classification on hidden states
  - runs over the whole dialogue history (user utterances + system actions)

(Žilka & Jurčiček, 2015)
http://arxiv.org/abs/1507.03471
• Neural networks primer
  • embeddings
  • layers (sigmoid, tanh, ReLU)
    • recurrent networks (LSTM, GRU), attention
• NN SLU examples: classifier/sequence
• Dialogue state vs. belief state
• Dialogue as (Partially observable) Markov Decision Process
• Tracker examples:
  • **Generative** (partitioning, parameter tying)
  • **Discriminative** (basic “rule-based”, classifier, neural)
  • **static** vs. **dynamic**
• Next time: dialogue policies
Contact us:
https://ufaldsg.slack.com/  
{odusek,hudecek,nekvinda}@ufal.mff.cuni.cz  
Skype/Meet/Zoom (by agreement)

Get these slides here:
http://ufal.cz/npfl123

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