NPFL123 Dialogue Systems

7. Neural NLU & State Tracking

https://ufal.cz/npfl123

Ondřej Dušek, Vojtěch Hudeček & Jan Cuřín

13. 4. 2021
Neural networks

• Can be used for both classification & sequence models
• **Non-linear functions**, composed of basic building blocks
  • stacked into **layers**
• Layers are built of **activation functions**:
  • linear functions
  • nonlinearities – sigmoid, tanh, ReLU
  • softmax – probability estimates:
    \[
    \text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}
    \]

• Fully differentiable – training by gradient descent
  • gradients **backpropagated** from outputs to all parameters
  • (composite function differentiation)

Sigmoid
\[
\sigma(x) = \frac{1}{1 + e^{-x}}
\]

**tanh**
\[
tanh(x)
\]

**ReLU**
\[
\max(0, x)
\]

https://medium.com/@shrutijson10104776/survey-on-activation-functions-for-deep-learning-9689331ba092
Neural networks – features

• You can use 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

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

$h_0 = 0$

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

$s_0 = h_T$

$p(y_t | y_1, \ldots, y_{t-1}, x) = \text{softmax}(s_t)$

$s_t = \text{cell}(y_{t-1}, s_{t-1})$


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

- encoder hidden state
- decoder state
- trained parameters

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 → softmax
  • convolutional networks
• Sequence tagging
  • RNN (LSTM/GRU) → 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)
• 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

• 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.

---

**X**

**S:** What part of town do you have in mind?

**X**

**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
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 2\textsuperscript{nd} time
    • accumulates probability over **NLU n-best lists**

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

1. "I'm looking for a Thai restaurant.

- **belief states**: hello(type=restaurant) 0.6, inform(type=restaurant, food=Thai) 0.4
- **actions**: You are looking for a restaurant right?

2. Thai.

- **belief states**: hello() 0.5, inform(food=Turkish) 0.3, inform(food=Thai) 0.2
- **actions**: You are looking for a restaurant right?

---

**no probability accumulation (1-best, no state)**

**accumulating over NLU n-best list (still no state)**

**accumulating over NLU n-best + turns**

(this is what we need (=belief state)

(from Milica Gašić's slides)
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

Partially Observable (PO)MDP
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
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 (= Belief Monitoring)

• Using the HMM model
  • estimate the transition & observation probabilities from data

\[ b(s) = \frac{1}{Z} p(o_t | s_t) \sum_{s_{t-1} \in S} p(s_t | a_{t-1}, s_{t-1}) b(s_{t-1}) \]

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

• Solutions: pruning/beams, additional assumptions…
  • or different models altogether
Generative BT: Pruning/Beams

• Tricks to make the naïve model tractable:
  • only track/enumerate states supported by NLU
    • “other” = all equal, don’t even keep the rest in memory explicitly
  • just keep $n$ most probable states (beam)
    • prune others & redistribute probability to similar states
  • merge similar states (e.g. same/similar slots, possibly different history)
    • along with probability mass

• Model parameters estimated from data
  • transition probabilities $p(s_{t+1}|s_t, a_t)$
  • observation probabilities $p(o_t|s_t)$
  • this is hard to do reliably, so they’re often set by hand
Hypotheticals not supported by NLU are ignored.

- Pruning an unlikely state & redistributing probability to similar ones.
- Merging similar states (note they're not the same).

*(from Filip Jurčiček’s slides)*
Generative BT: Independence Assumptions

- **Partition the state** by assuming conditional independence
  - track parts of the state independently → reduce # of combinations
  - e.g. “each slot is independent”:
    - state \( s = [s^1, \ldots, s^N] \), belief \( b(s_t) = \prod_i b(s^i_t) \)
  - other partitions possible – speed/accuracy trade-off

- Per-slot updates:
  \[
  b(s^i_t) = \sum_{s_{t-1}, o_t} p(s^i_t | a^i_{t-1}, s^i_{t-1}, o^i_t) b(s^i_{t-1})
  = \sum_{s_{t-1}, o_t} p(s^i_t | a^i_{t-1}, s^i_{t-1}) p(o^i_t | s^i_t) b(s^i_{t-1})
  \]
  - per-slot dependencies only
  - transition probability
  - observation probability
  - last belief

(Žilka et al., 2013)
https://www.aclweb.org/anthology/W13-4070/
Generative BT: Parameter Tying

- Further simplification: keep the partition + tie some parameters
  - you basically end up with 2 parameters only 😊

transition probabilities:

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

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

observation probabilities:

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

$\theta_O \sim$ confidence in NLU

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

substitution

$\frac{\text{the rule is now deterministic}}{\begin{cases} s^i_t = \bigcirc: & p(s^i_{t-1} = \bigcirc) p(o^i_t = \bigcirc) \\ s^i_t \neq \bigcirc: & p(o^i_t = s^i_t) + p(o^i_t = \bigcirc) p(s^i_t = s^i_{t-1}) \end{cases}}$

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

• 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  (Williams, 2012) https://ieeexplore.ieee.org/document/6424197

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

• 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: (Metallinou et al., 2013) [https://www.aclweb.org/anthology/P13-1046]
  • NLU n-best size, entropy, lengths (current turn, cumulative)
  • ASR scores

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

- Dialogue as a sequence \( p(s_1, \ldots, s_t | o_1, \ldots, o_t) \)
- **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>Hello, How may I help you?</td>
<td>Persian, 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>Persian, South</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, South</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, South</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, South</td>
</tr>
<tr>
<td>( U_6 )</td>
<td>The phone number of nandos is 01233 323908.</td>
<td></td>
</tr>
<tr>
<td>( S_6 )</td>
<td>And the address?</td>
<td>Portuguese, South</td>
</tr>
<tr>
<td>( S_7 )</td>
<td>Start, nandos is on Cambridge Leisure Park Clifton Way.</td>
<td></td>
</tr>
<tr>
<td>( U_7 )</td>
<td>Thank you good bye.</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 + \( \sum \text{previous} \)

(Imagine this part for all \( v \)'s)

(\( T \) previous timesteps)

sum of everything before then

\( f_1 \) \( f_1(t, v) \) \( f_1(t - T + 1, v) \) \( \sum_{t=0}^{T-1} f_1(t, v) \)
\( f_2 \) \( f_2(t, v) \) \( f_2(t - T + 1, v) \) \( \sum_{t=0}^{T-1} f_2(t, v) \)
\( \vdots \)
\( f_M \) \( f_M(t, v) \) \( f_M(t - T + 1, v) \) \( \sum_{t=0}^{T-1} f_M(t, v) \)

\( h_1 [= \tanh(W_0 f^T + b_0)] \)
\( h_2 [= \tanh(W_1 h_1^T + b_1)] \)
\( h_3 [= \tanh(W_2 h_3^T + b_2)] \)

\( E(t, v) [= W_3 h_3^T] \)

softmax over all possible \( v \)'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
Summary

• Neural networks primer
  • embeddings
  • layers (sigmoid, tanh, ReLU)
  • recurrent networks (LSTM, GRU), attention

• NN NLU 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}@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

Next week:
Lab questions 9am
Lab assignment 9:50
Lecture 10:40