Dialogue Systems
NPFL123 Dialogové systémy

7. NLU with Neural Networks & Dialogue State Tracking

Ondřej Dušek & Ondřej Plátek & Jan Cuřín

ufal.cz/npfl123

2. 4. 2019
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:
    \[
    \text{softmax}(\mathbf{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{\|\mathbf{x}\|} \exp(x_j)}
    \]
- 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**
\[
\text{max}(0, x)
\]

https://medium.com/@shrutijadon10104776/survey-on-activation-functions-for-deep-learning-9689331ba092
Neural networks – features

• You can use the same 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)
  • distributed word representation
    • each word = **vectors of floats**
  • part of network parameters – trained
    a) random initialization
    b) pretraining
  • 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

$h_0 = 0$
$h_t = \text{cell}(x_t, h_{t-1})$
$s_0 = h_T$

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

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

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

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

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

attention weights = **alignment model**
attention weights = alignment model

• **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

• 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
NN NLU – Joint Intent & Slots


• 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

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

• 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

same as (c) on previous slide

this is new
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?
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)

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 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

no probability accumulation (1-best, no state)

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

accumulating over NLU n-best + turns

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

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

You are looking for a restaurant right?

Did you say Thai or Turkish?

What kind of food would you like?

What kind of food would you like?

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)
Partially Observable (PO)MDP

• 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

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

• **Discriminative models**
  1) Assume some functional form for $p(y|x)$
  2) 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

http://cs229.stanford.edu/notes/cs229-notes2.pdf
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 \( \rightarrow 10^{10} \) 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
Generative BT: Pruning/Beams

hypotheses not supported by NLU are ignored

merging similar states (note they’re not the same)

pruning an unlikely state & redistributing probability to similar ones

(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 $s = [s_1, ... s_N]$, belief $b(s_t) = \prod_i b(s^i_t)$
    - other partitions possible – speed/accuracy trade-off

- **Slot partition:**
  - $b(s^i_t) = \sum_{s^i_{t-1}, o^i_t} p(s^i_t | a^i_{t-1}, s^i_{t-1}, o^i_t) b(s^i_{t-1})$
  - $= \sum_{s^i_{t-1}, o^i_t} p(s^i_t | a^i_{t-1}, s^i_{t-1}) p(o^i_t | s^i_t) b(s^i_{t-1})$
    - transition probability
    - observation probability
    - last belief

- **Further simplification:** parameter tying
  - $\theta_T \sim$ rigidity (bias for keeping old values)
  - $p(s^i_t | a^i_{t-1}, s^i_{t-1}) = \begin{cases} \theta_T & \text{if } s^i_t = s^i_{t-1} \\ \frac{1 - \theta_T}{\# \text{values}^i_{t-1}} & \text{otherwise} \end{cases}$
  - $p(o^i_t | s^i_t) = \begin{cases} \theta_o p(o^i_t) & \text{if } o^i_t = s^i_t \\ \frac{1 - \theta_o}{\# \text{values}^i_{t-1}} p(o^i_t) & \text{otherwise} \end{cases}$
  - $\theta_o \sim$ confidence in NLU
  - $p(o^i_t) = \text{NLU output}$
Basic Discriminative Belief Tracker

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

**Update Rule**

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

**Discriminative Model**

\[
p(s_t^i | a_{t-1}^i, s_{t-1}^i, o_t^i) = \begin{cases} 
  p(o_t^i) & \text{if } s_t^i = o_t^i \land o_t^i \neq \topi \\
  p(o_t^i) & \text{if } s_t^i = s_{t-1}^i \land o_t^i = \topi \\
  0 & \text{otherwise}
\end{cases}
\]

**Substitution**

\[
b(s_t^i) = \begin{cases} 
  p(s_t^i = \topi) p(o_t^i = \topi) & \text{if } s_t^i = \topi \\
  p(o_t^i = s_t^i) + p(o_t^i = \topi) p(s_t^i = s_{t-1}^i) & \text{otherwise}
\end{cases}
\]
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


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

• General distinction
  • **static models** – encode whole history into features
  • **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 \( \nu \) examples:
  • rank & score of hypo with \( \nu \) on current NLU n-best + diff vs. top-scoring hypo
  • # times \( \nu \) appeared so far, sum/average confidence of that
  • # negations/confirmations of \( \nu \) so far
  • reliability of NLU predicting \( \nu \) on held-out data

https://www.aclweb.org/anthology/P13-1046
**Sequence-Based Discriminative Trackers**

- Dialogue as a sequence $p(s_1, \ldots, s_t | o_1, \ldots, o_t)$
- **CRF** models
  - similar features as previously – can be current-slot only (CRF will handle it)
  - feature value: NLU score for the given thing (e.g. DA type + slot + value)
  - target: per-slot BIO coding

---

**Utterance** | **Goals** | **Food** | **Area**
---|---|---|---
S_1 | Hello, How may I help you? | Persian | South
S_2 | I need a Persian restaurant in the south part of town. | Persian | South
S_3 | What kind of food would you like? | Persian | South
S_4 | I'm sorry but there is no restaurant serving persian food. | Persian | South
S_5 | How about Portuguese food? | Portuguese | South
S_6 | Peking restaurant is a nice place in the south of town. | Persian | South
S_7 | Is that Portuguese? | Portuguese | South
S_8 | Nanos is a nice place in the south of town serving tasty Portuguese food. | Portuguese | South
S_9 | Alright, What's the phone number? | Portuguese | South
S_10 | The phone number of nanos is 01223 327908. | Portuguese | South
S_11 | And the address? | Portuguese | South
S_12 | Sure, nanos is on Cambridge Leisure Park Clifton Way. | Portuguese | South
S_13 | Thank you good bye. | Portuguese | South

---

NPFL123 L7 2019

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

27
Neural State Trackers

• Many different architectures
• Typically sequential, discriminative
• 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)
https://dl.acm.org/citation.cfm?id=2955040
http://arxiv.org/abs/1507.03471
Neural State Trackers

• More complex – better generalization across slots

( Zhong et al., 2018)
http://arxiv.org/abs/1805.09655

encoders shape:

local = per-slot, global = shared among slots

attention over prev. system actions
w. r. t. current user utterance

weighted sum + sigmoid

if utterance refers to previous system actions

β · global + (1 − β) · local

does the utterance specify this slot-value pair?
attention over utterance w. r. t. slot-value pair

http://arxiv.org/abs/1805.09655
Summary

• Neural networks primer
  • embeddings
  • layers (sigmoid, tanh, ReLU)
  • recurrent networks (LSTM, GRU)
  • attention
• NN SLU examples
• Dialogue state, belief state
• Dialogue as (Partially observable) Markov Decision Process
• Generative belief trackers
• Discriminative belief trackers
• NN tracker examples
Thanks

Contact me:
odusek@ufal.mff.cuni.cz
room 424 (but email me first)

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

Labs tomorrow
9:00 SU1