NPFL123 Dialogue Systems
5. NLU vol. 2 & State Tracking

https://ufal.cz/npfl123

Ondřej Dušek, Simone Balloccu, Mateusz Lango, Kristýna Klesnilová & Jan Cuřín
13. 3. 2024
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:
    \[
    \text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{\text{|x|}} \exp(x_j)}
    \]
- 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
  • network learns words with similar usage
    • 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
    - softmax: focuses mainly on 1 thing
  - re-weighted every decoder step
    → can focus on currently important part of input
- **Self-attention** – over previous decoder steps
- In RNNs: added to dec. inputs & dec. softmax layer

[Image: Attention Mechanism]

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

• getting rid of (encoder) recurrences
  • making it faster to train, allowing bigger nets
  • replace everything with blocks of attention & feed-forward
    • ⇒ needs more layers
    • ⇒ needs to encode positions

• positional encoding
  • adding position-dependent patterns to the input

• attention: more heads
  • ~more attentions in parallel
  • focus on multiple inputs

—Waswani et al., 2017—
https://arxiv.org/abs/1706.03762

we can put classification on top (see →→)

all attention no recurrences
(depends on lower layer only)


http://jalammar.github.io/illustrated-transformer/
Neural NLU

- Various architectures possible
- **Classification**
  - feed-forward NN
  - RNN + attention weight → softmax
  - convolutional networks
  - Transformer
- **Sequence tagging**
  - RNN (LSTM/GRU) → softmax over hidden states
    - default version: label bias (like MEMM)
    - CRF over the RNN possible
  - Transformer works the same
  - Intent can be tagged at start of sentence

---

https://medium.com/swlh/nlu-for-everyone-with-bert-7bedaa609a61
(Chen et al., 2019) http://arxiv.org/abs/1902.10909
RNN-based NLU

- Same RNN-based network for both tasks
- **Bidirectional encoder**
  - 2 encoders: LTR, RTL & concatenate hidden states
  - “see the whole sentence before you start tagging”
- **Decoder** – tags 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)
  - use attention over slot hidden states
Transformer-based NLU

• slot tagging on top of pretrained BERT Transformer model
  • BERT = Transformer trained to guess masked words (on very large data)
  • further trained for NLU
  • standard **IOB approach**
  • just feed final hidden layers to **softmax over tags**
    • classify only at 1st subword in case of split words
      (don’t want tag changes mid-word)

• special start token tagged with intent
• optional CRF on top of the tagger
  • for global sequence optimization

(Chen et al., 2019)

http://arxiv.org/abs/1902.10909
Handling ASR noise

- ASR produces multiple hypotheses
- Combine & get resulting NLU hypotheses
  - NLU: $p(\text{DA}|\text{text})$
  - ASR: $p(\text{text}|\text{audio})$
  - we want $p(\text{DA}|\text{audio})$
- Easiest: **sum it up**

$$p(\text{DA}|\text{audio}) = \sum_{\text{texts}} P(\text{DA}|\text{text})P(\text{text}|\text{audio})$$

0.33 — I am looking for a bar
0.26 — I am looking for the bar
0.11 — I am looking for a car
0.09 — I am looking for the car
0.59 — inform(task=find, venue=bar)
0.20 — null()

(from Filip Jurčíček’s slides)
Handling ASR noise

- Alternative: **use confusion networks**
  - per-word ASR confidence
- Word features weighed by word confidence

0.33 - I am looking for a bar
0.26 - I am looking for the bar
0.11 - I am looking for a car
0.09 - I am looking for the car

~equivalent confusion network

(features:)
- I 0.9
- hi 0.02
- am 0.9
- looking 1
- for 1
- ...
- I am 0.81
- my am 0.063
- looking 0.9
- a bar 0.3
- a car 0.24
- ...

should be normalized by n-gram length

(from Filip Jurčiček’s slides)
• user response can depend on last system action
  • fragments/short replies are ambiguous without context
  • → add last system DA/text into input features
    • helps disambiguate
• careful – user may not play nice!
  • system needs to be trained with both alternatives in mind

U: I’m looking for flights from JFK.  
S: Where would you like to go?  
U: Atlanta.

inform(??=Atlanta)  
inform(to_city=Atlanta)

x  U: Actually I’d rather fly from Newark.
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
• 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 2\textsuperscript{nd} time
    - accumulates probability over **NLU n-best lists**

- Plays well with probabilistic dialogue policies
  - but not only them – rule-based, too
(based on 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

- **State**: $s_t$
- **Action**: $a_t$
- **Observation**: $o_t$
- **Reward**: $r_t$

- Grey = observed
- White = unobserved

(from Milica Gašić’s slides)
Naïve Generative Belief Tracking

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

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

• Solutions:
  • 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: \( b(s_t^i) = \sum_{s_{t-1}, o_t^i} p(o_t^i | s_t^i) p(s_t^i | a_t^i, s_{t-1}^i) b(s_{t-1}^i) \)

  *observation probability*  *transition probability*  *previous belief*

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

transition probabilities:

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

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

observation probabilities:

\[
p(o_t^i | s_t^i) = \begin{cases} \theta_O p(o_t^i) & \text{if } o_t^i = s_t^i \\ \frac{1-\theta_O}{\# \text{values}^{t-1}} p(o_t^i) & \text{otherwise} \end{cases}
\]

\( \theta_O \sim \text{confidence in NLU} \)

\( p(o_t^i) = \text{NLU output} \)

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

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

NLU output

- “user mentioned this value”
- “no change”

user silent about slot \( i \)

substitution

the rule is now deterministic

(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

- Another 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, NN, …)
  - need fixed feature vector from $o_1, a_1, ..., a_{t-1}, o_t$ (where $t$ is arbitrary)
    - current turn, cumulative (sum), sliding window ($k$ previous turns)

- 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

(Henderson et al., 2013) https://aclweb.org/anthology/W13-4073
(Metallinou et al., 2013) https://www.aclweb.org/anthology/P13-1046
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

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

• Based on RNNs (turn-level or word-level)
• Typically **not** using a 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 (in 2 weeks!): dialogue policies
Contact us:  
https://ufaldsg.slack.com/  
odusek@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

No lecture next week!