

Dialogue Systems

NPFL123 Dialogové systémy

7. Neural NLU & Dialogue State Tracking

Ondřej Dušek & Vojtěch Hudeček & Jan Cuřín

<http://ufal.cz/npfl123>

31. 3. 2020

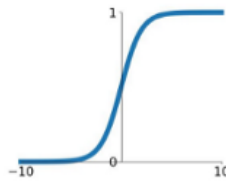
Neural networks

- 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)

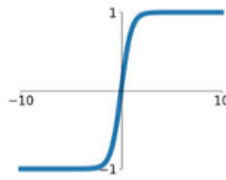
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



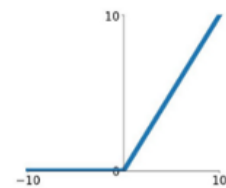
tanh

$$\tanh(x)$$



ReLU

$$\max(0, x)$$

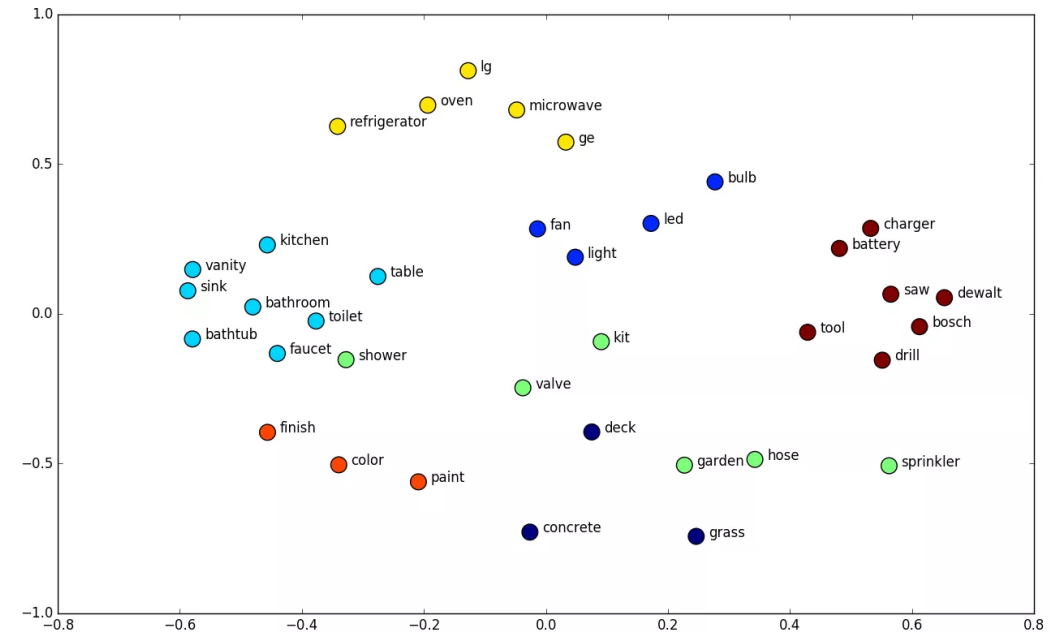


https://medium.com/@shrutija_don10104776/survey-on-activation-functions-for-deep-learning-9689331ba092

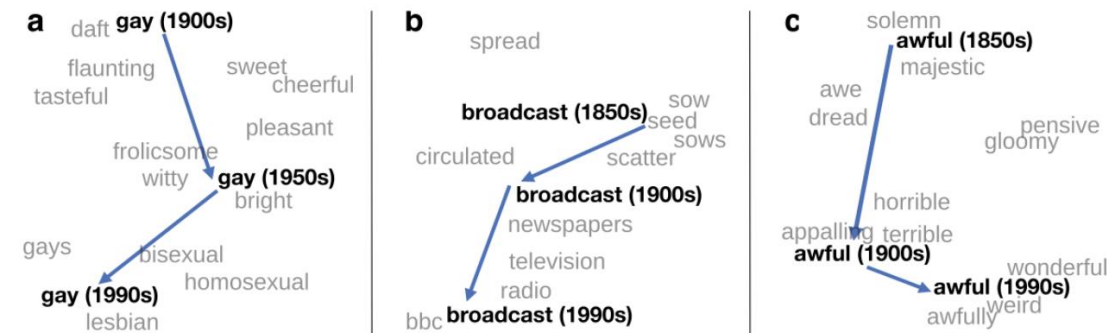
Neural networks – features



- 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://blog.kaggle.com/2016/05/18/home-depot-product-search-relevance-winners-interview-1st-place-alex-andreas-nurlan/>



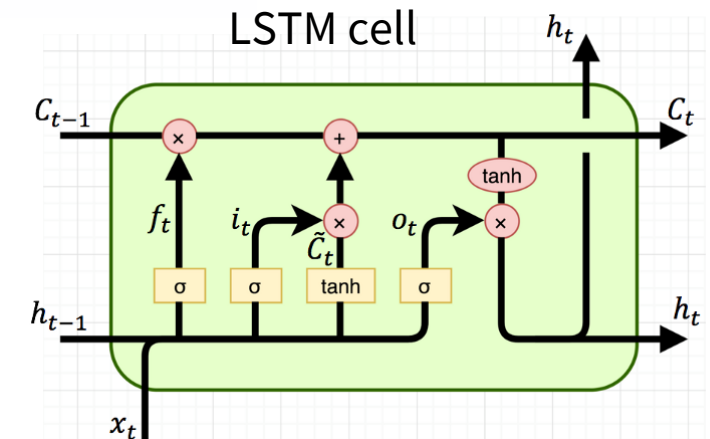
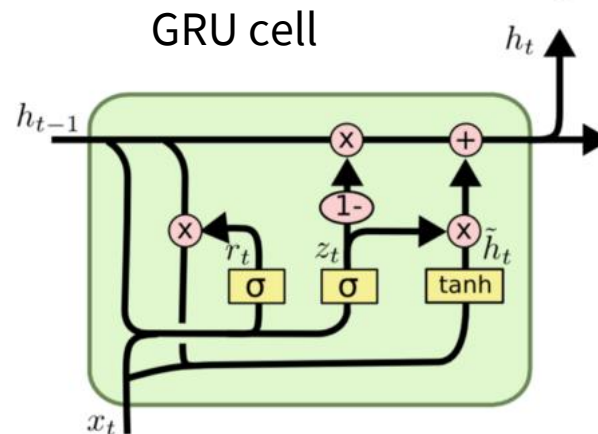
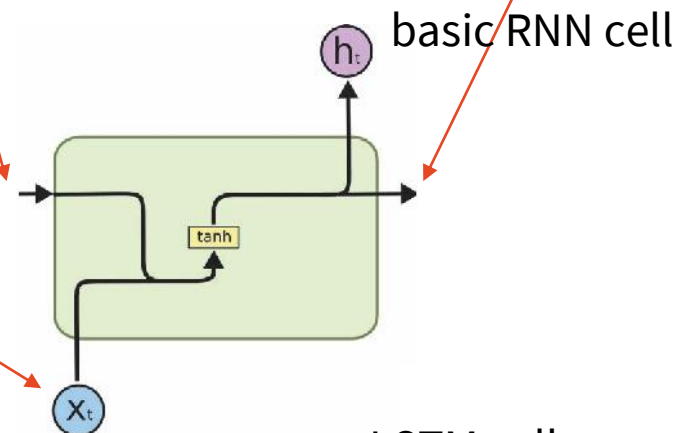
<http://ruder.io/word-embeddings-2017/>

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

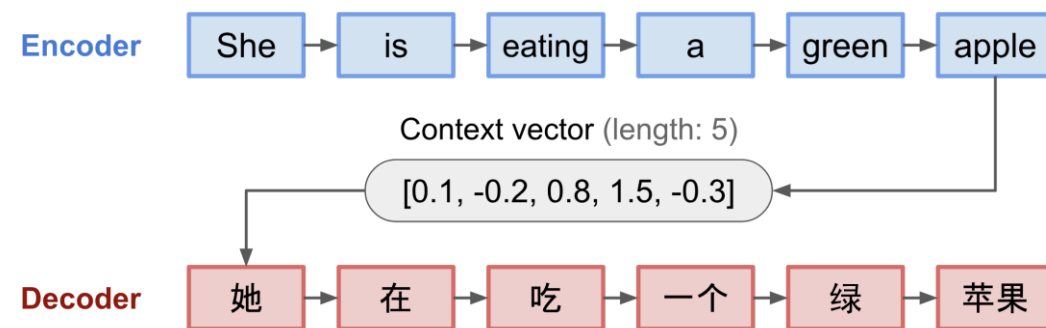
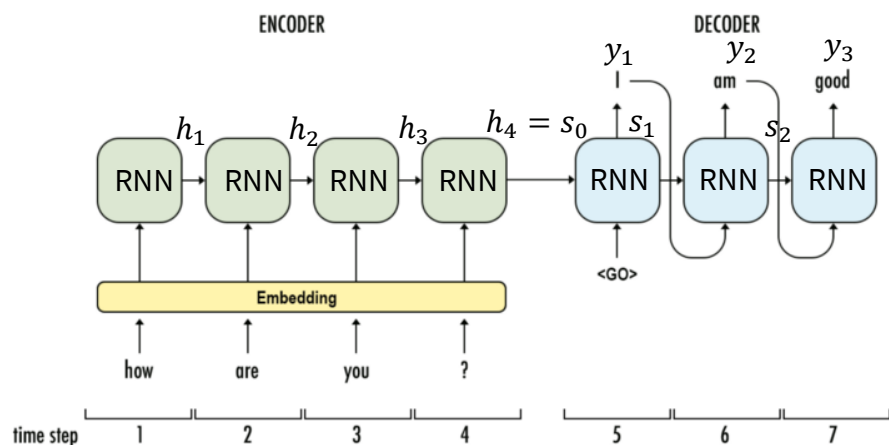
$$h_0 = \mathbf{0}$$

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

$$s_0 = h_T$$

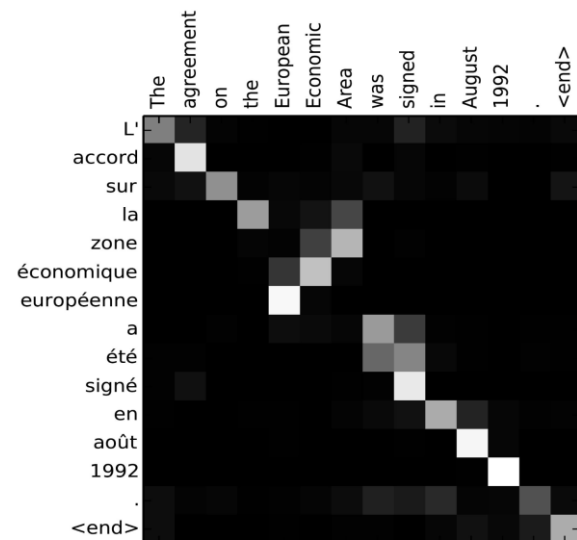
$$p(y_t | y_1, \dots, y_{t-1}, \mathbf{x}) = \text{softmax}(s_t)$$

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



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**

t = decoder step

$1 \dots n$ = encoder steps

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

encoder hidden state

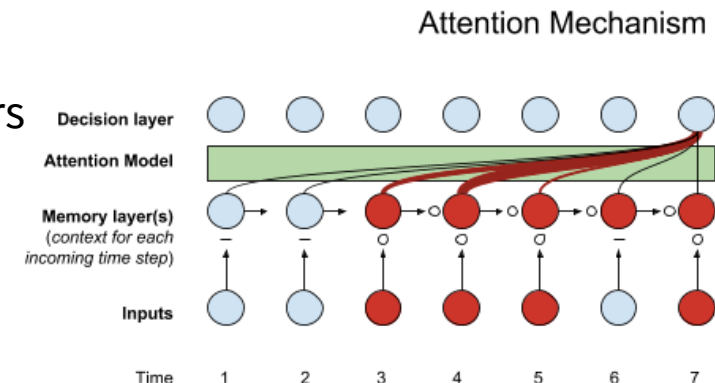
attention weights

= **alignment model**

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

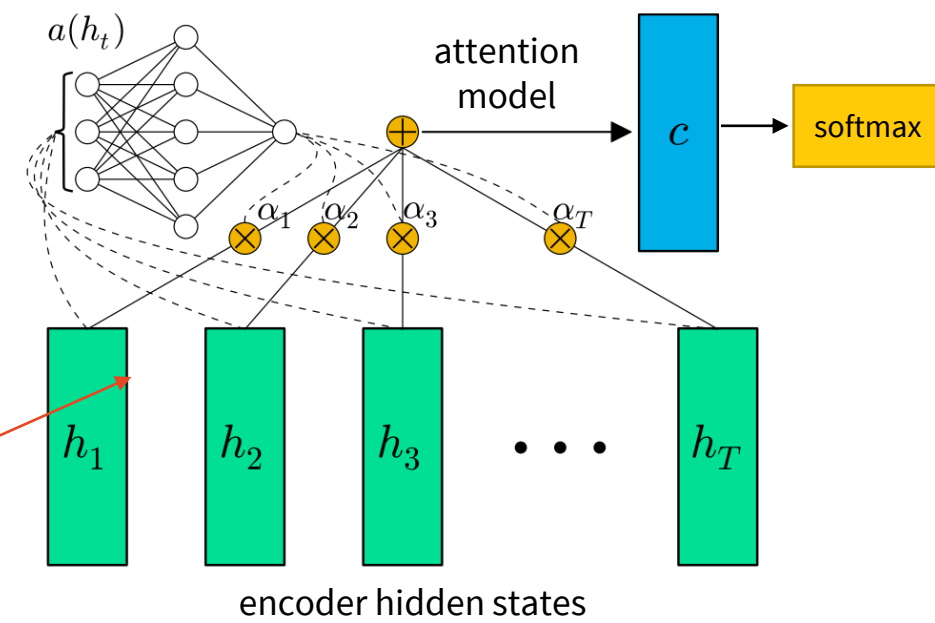
decoder state trained parameters

- Self-attention** – over previous decoder steps

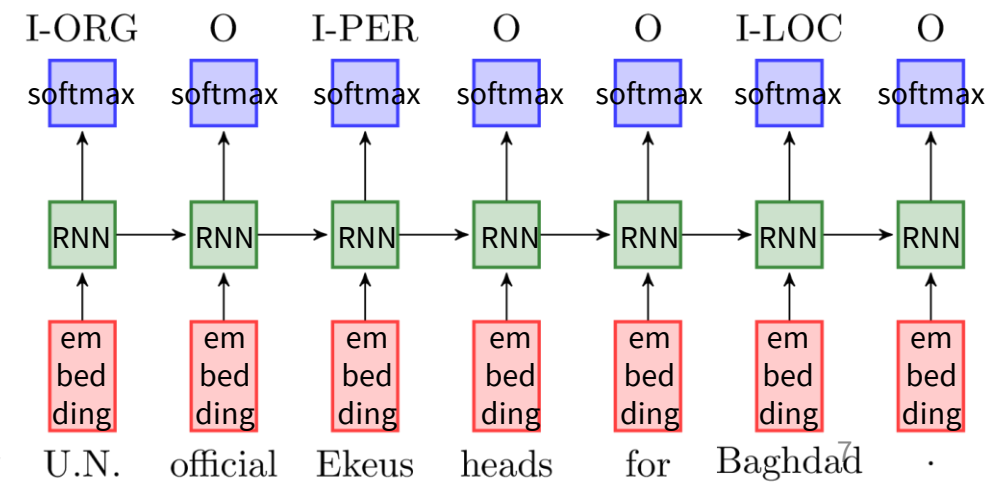


Neural NLU

- Various architectures possible
- Classification
 - feed-forward NN
 - RNN + attention weight \rightarrow softmax
 - convolutional networks
- 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



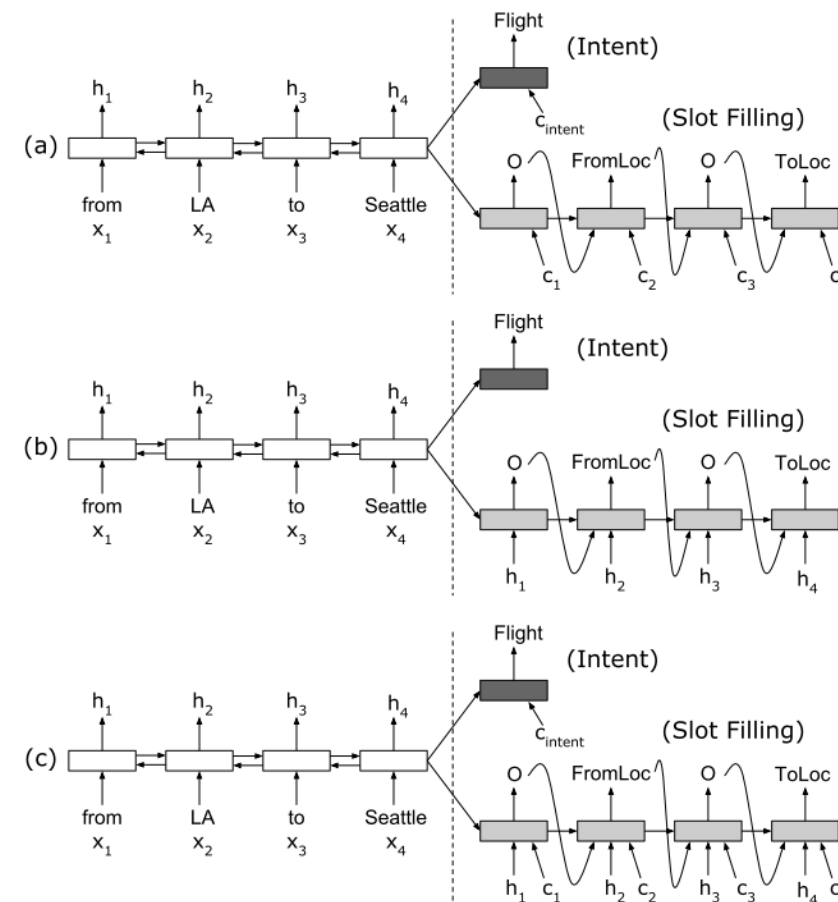
(Raffel & Ellis, 2016)
<https://colinraffel.com/publications/iclr2016feed.pdf>



NN NLU – Joint Intent & Slots

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

- 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)

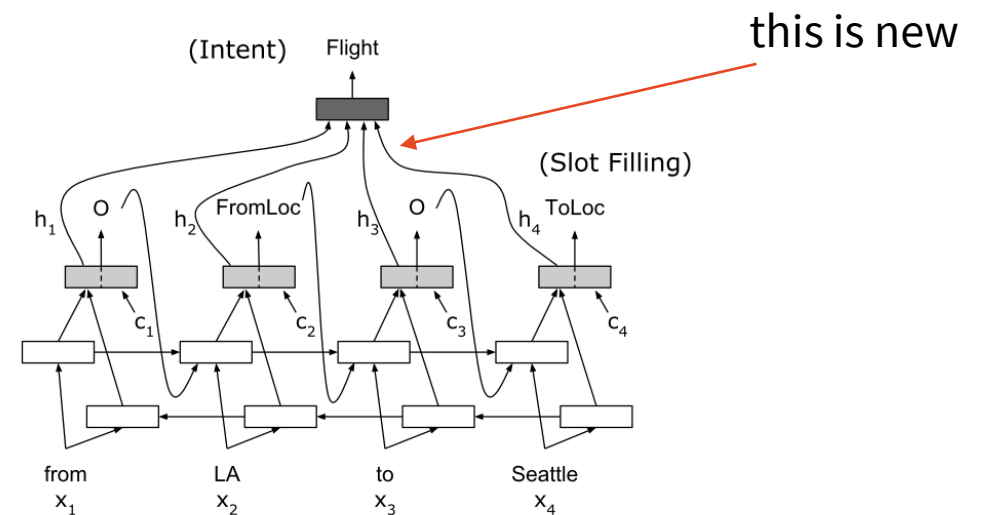


NN NLU – Joint Intent & Slots

(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



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)

<https://ai.google/research/pubs/pub44018>

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

- Past **system actions**

- information provided
 - slots and values
 - list of venues offered

U: Give me the address of the first one you talked about.

U: Is there any other place in this area?

- slots confirmed

S: OK, Chinese food. [...]

- slots requested

S: What time would you like to leave?

- **Other** semantic context

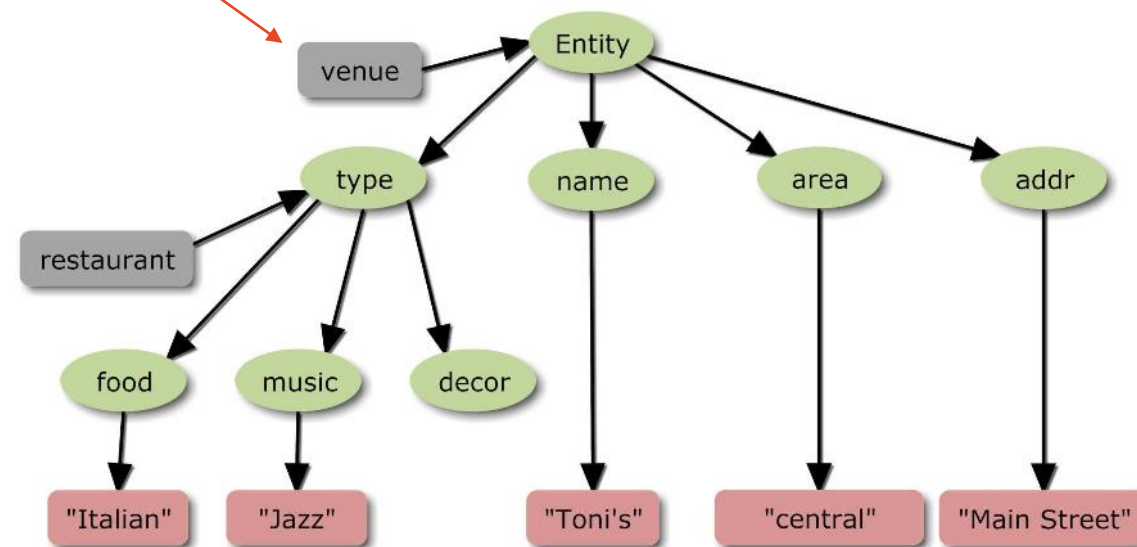
- user/system utterance: bye, thank you, repeat, restart etc.

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)

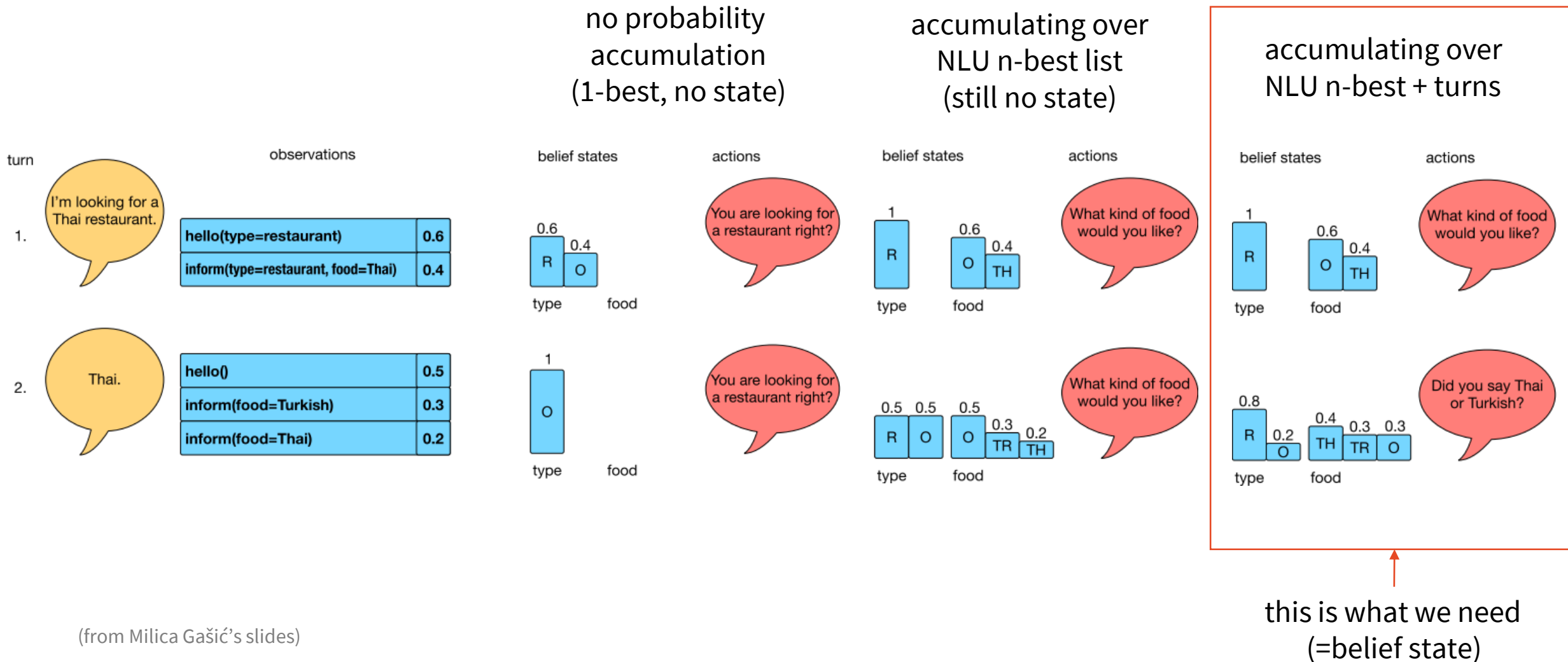
Problems with Dialogue State

- NLU is unreliable
 - takes unreliable ASR output → ASR: 0.5 I'm looking for an expensive hotel
0.5 I'm looking for inexpensive hotels
 - 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 → NLU: 0.3 inform(type=restaurant, stars=5)
 - what's "low"?
 - what if we ignore 10x the same thing?
- Better solution: make the state probabilistic – **belief state**
 - 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

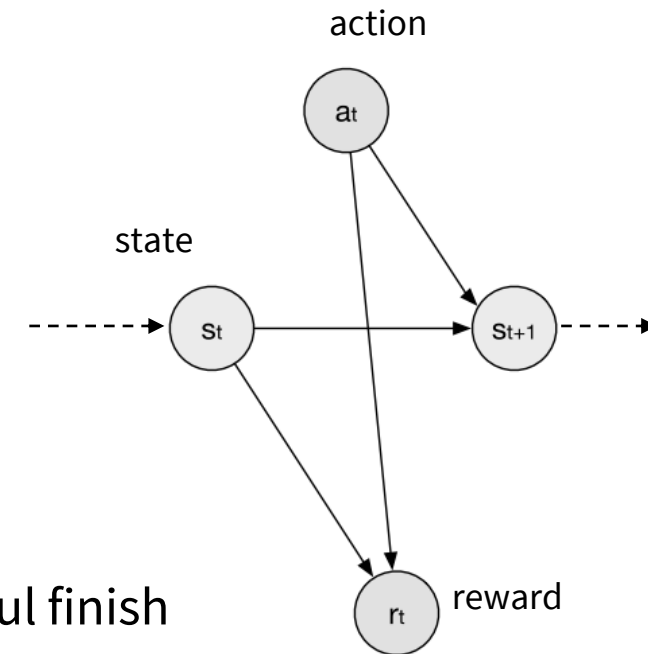
Belief State



(from Milica Gašić's slides)

Dialogue as a Markov Decision Process

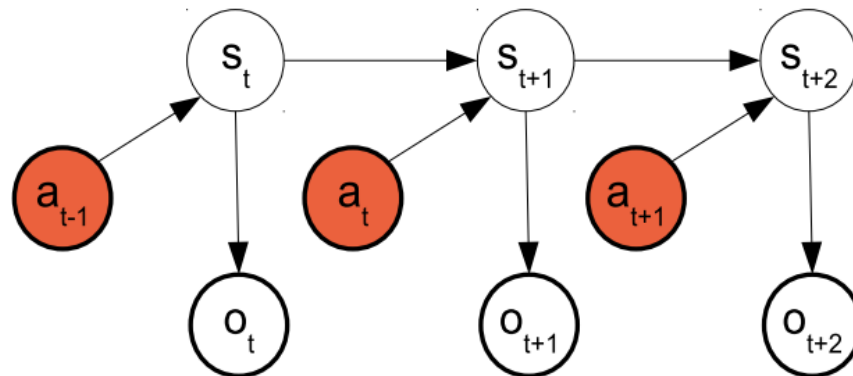
- MDP = probabilistic control process
 - model – Dynamic Bayesian Network
 - random variables & dependencies in a graph/network
 - “dynamic” = structure repeats over each time step t
 - s_t – dialogue **states** = what the user wants
 - a_t – **actions** = what the system says
 - r_t – **rewards** = 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



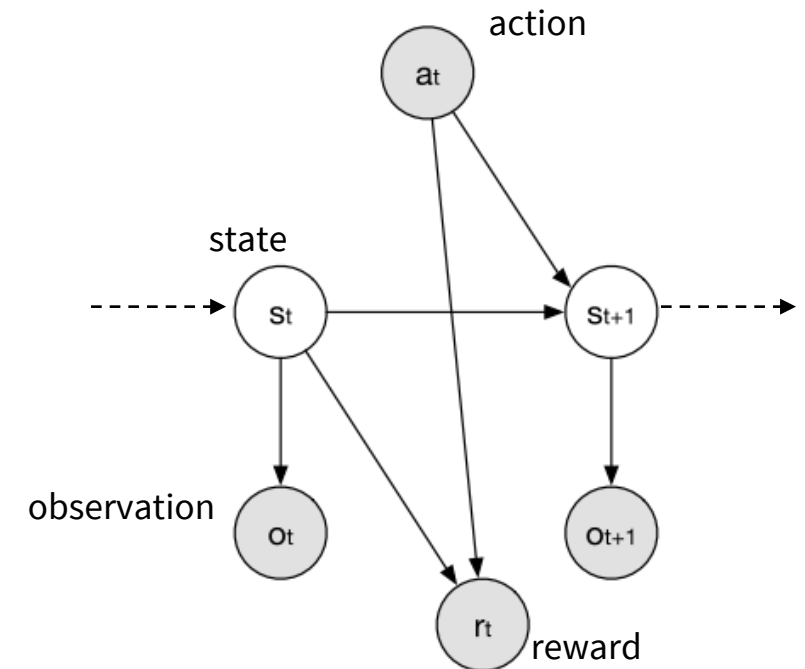
(from Milica Gašić’s slides)

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)



grey = observed
white = unobserved (from Milica Gašić'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 $p(y), p(x|y)$
- 2) Estimate parameters of $p(y), p(x|y)$ 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 ^UFAL



Example: elephants vs. dogs

<http://cs229.stanford.edu/notes/cs229-notes2.pdf>

- **Discriminative:**

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

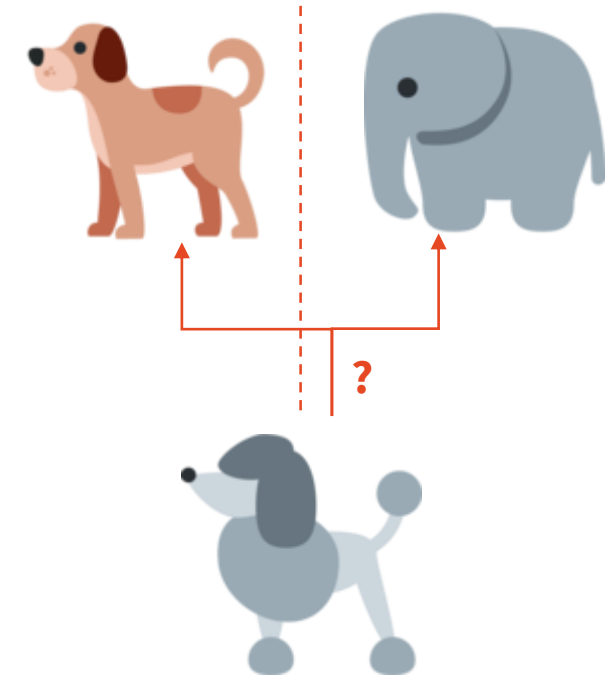
- **Generative**

- ~ 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



Naiïve Generative Belief Tracking

(= Belief Monitoring)

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

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

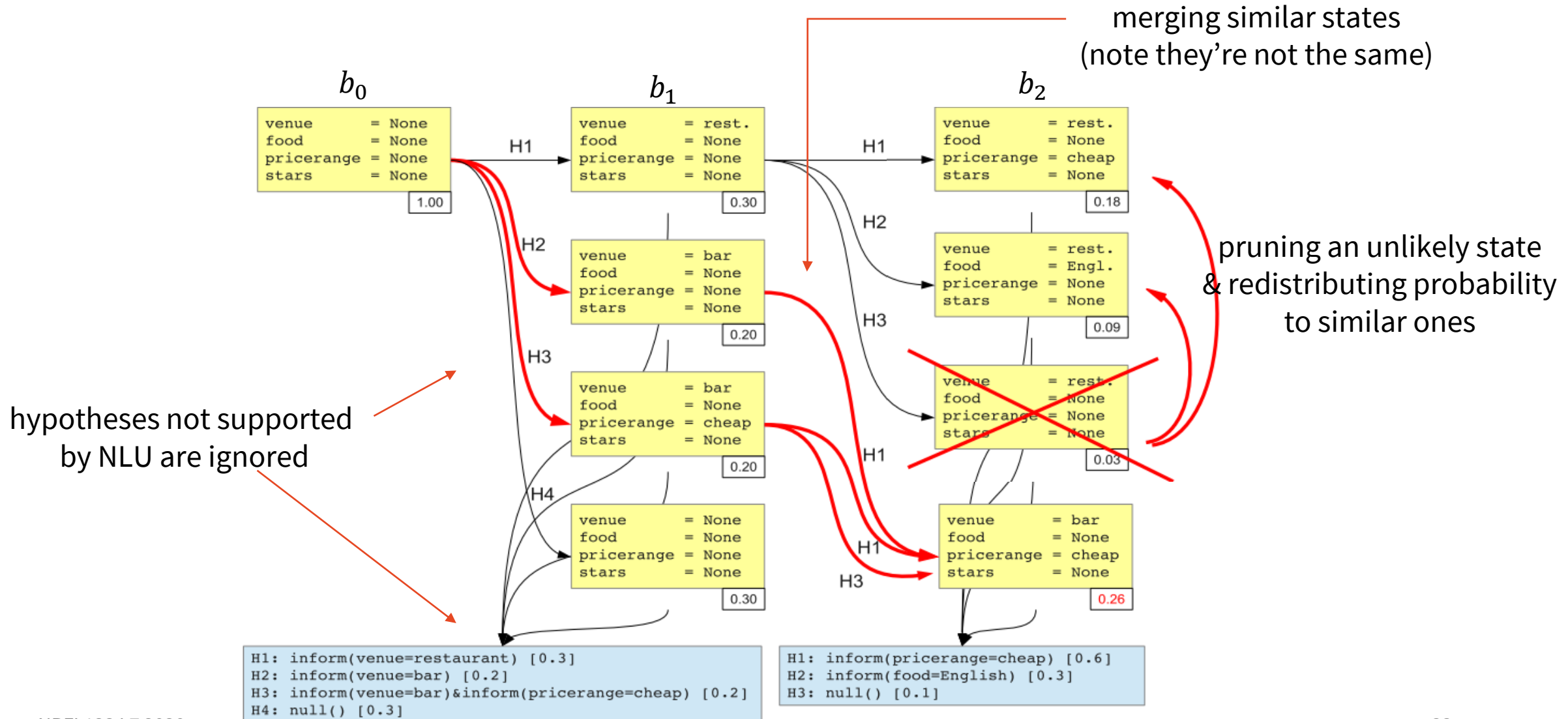
← same as previous

- 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



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 $\mathbf{s} = [s^1, \dots, s^N]$, belief $b(\mathbf{s}_t) = \prod_i b(s_t^i)$
 - other partitions possible – speed/accuracy trade-off

- Per-slot updates:

$$\begin{aligned}
 b(s_t^i) &= \sum_{s_{t-1}, o_t^i} p(s_t^i | a_{t-1}^i, s_{t-1}^i, o_t^i) b(s_{t-1}^i) \\
 &= \sum_{s_{t-1}, o_t^i} \underbrace{p(s_t^i | a_{t-1}^i, s_{t-1}^i)}_{\text{transition probability}} \underbrace{p(o_t^i | s_t^i)}_{\text{observation probability}} \underbrace{b(s_{t-1}^i)}_{\text{last belief}}
 \end{aligned}$$

per-slot dependencies only

Generative BT: Parameter Tying

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

transition probabilities:

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

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

observation probabilities:

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

$\theta_o \sim$ confidence in NLU

$p(o_t^i)$ = NLU output

i.e. believe in value given by NLU with θ_o ,
distribute rest of probability equally

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

NLU output

“user mentioned this value”

$$p(s_t^i | a_{t-1}^i, s_{t-1}^i, o_t^i) = \begin{cases} p(o_t^i) & \text{if } s_t^i = o_t^i \wedge o_t^i \neq \text{🗣️} \\ p(o_t^i) & \text{if } s_t^i = s_{t-1}^i \wedge o_t^i = \text{🗣️} \\ 0 & \text{otherwise} \end{cases}$$

“no change”

user silent about slot i

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

substitution

$$b(s_t^i) = \begin{cases} s_t^i = \text{🗣️}: p(s_{t-1}^i = \text{🗣️}) p(o_t^i = \text{🗣️}) \\ s_t^i \neq \text{🗣️}: p(o_t^i = s_t^i) + p(o_t^i = \text{🗣️}) p(s_t^i = s_{t-1}^i) \end{cases}$$

the rule is now deterministic

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 <http://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
 - **sequence models** – explicitly model dialogue as sequential

Static Discriminative Trackers

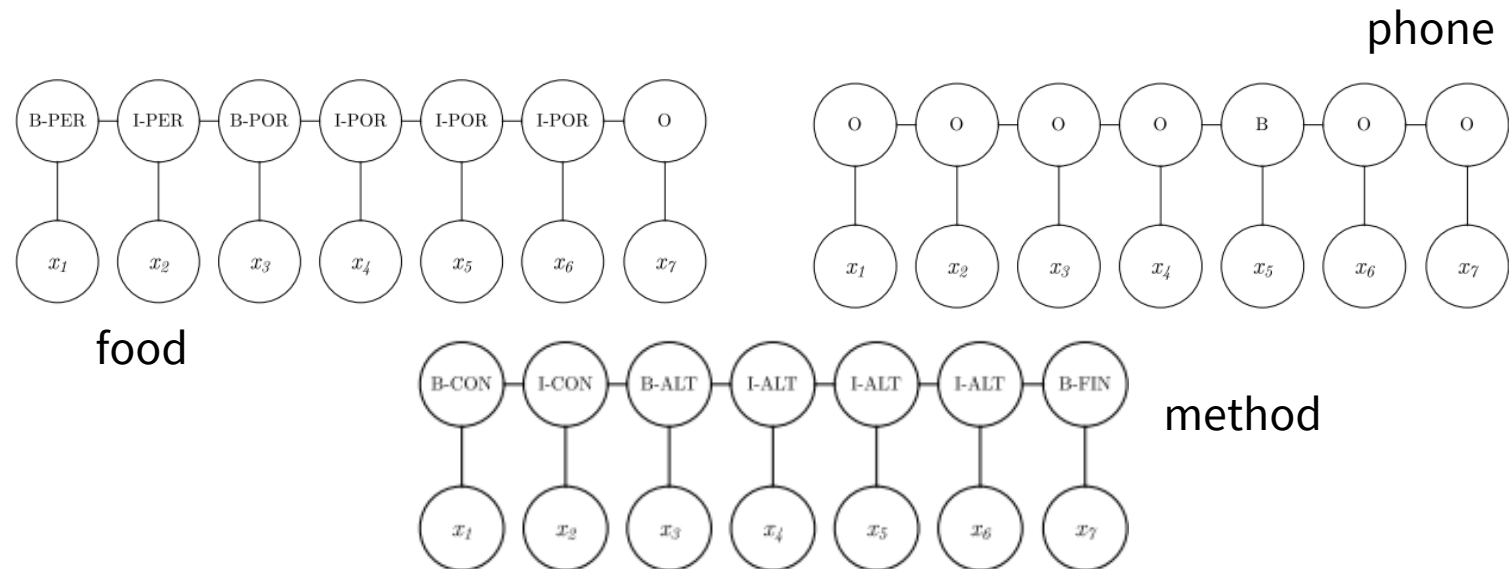


- Generally predict $p(s_t | o_1, a_1, \dots, a_{t-1}, o_t)$
 - any kind of classifier (SVM, LR...)
 - need fixed feature vector from $o_1, a_1, \dots, a_{t-1}, o_t$ (where t is arbitrary)
 - current turn, cumulative, sliding window
 - per-value features & tying weights– some values are too rare
- Global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

Sequence-Based Discriminative Trackers

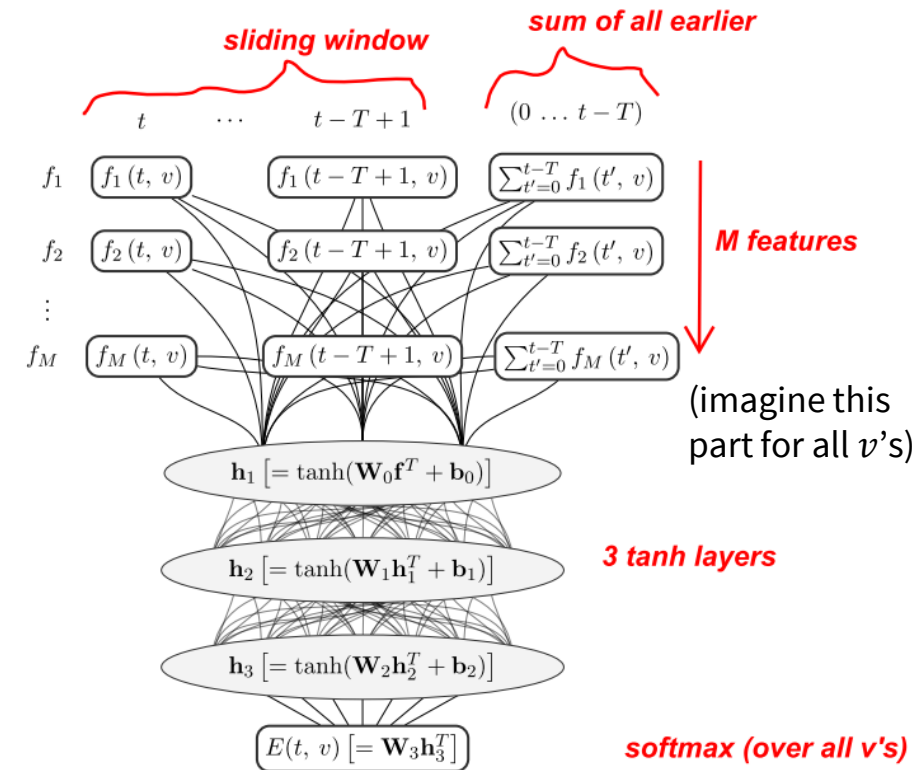
- Dialogue as a sequence $p(s_1, \dots, s_t | o_1, \dots, o_t)$
- **CRF** models
 - similar features as previously – can be current-slot only (CRF will handle it)
 - feature value: NLU score for the given thing (e.g. DA type + slot + value)
 - target: per-slot BIO coding

Utterance		Goals	
		Food	Area
S ₁	Hello, How may I help you?		
U ₁	I need a Persian restaurant in the south part of town.	Persian	South
S ₂	What kind of food would you like?		
U ₂	Persian.	Persian	South
S ₃	I'm sorry but there is no restaurant serving persian food		
U ₃	How about Portuguese food?	Portuguese	South
S ₄	Peking restaurant is a nice place in the south of town.		
U ₄	Is that Portuguese?	Portuguese	South
S ₅	Nandos is a nice place in the south of town serving tasty Portuguese food.		
U ₅	Alright. Whats the phone number?	Portuguese	South
S ₆	The phone number of nandos is 01223 327908 .		
U ₆	And the address?	Portuguese	South
S ₇	Sure, nandos is on Cambridge Leisure Park Clifton Way.		
U ₇	Thank you good bye.		



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)
 - SLU score of v
 - n-best rank of v
 - user & system act type
 - ... – domain-independent, low-level NLU outputs
 - 3 tanh layers
 - output – softmax (= probability distribution over values)
 - static: sliding window
 - current time t
 - few steps back
 - \sum previous

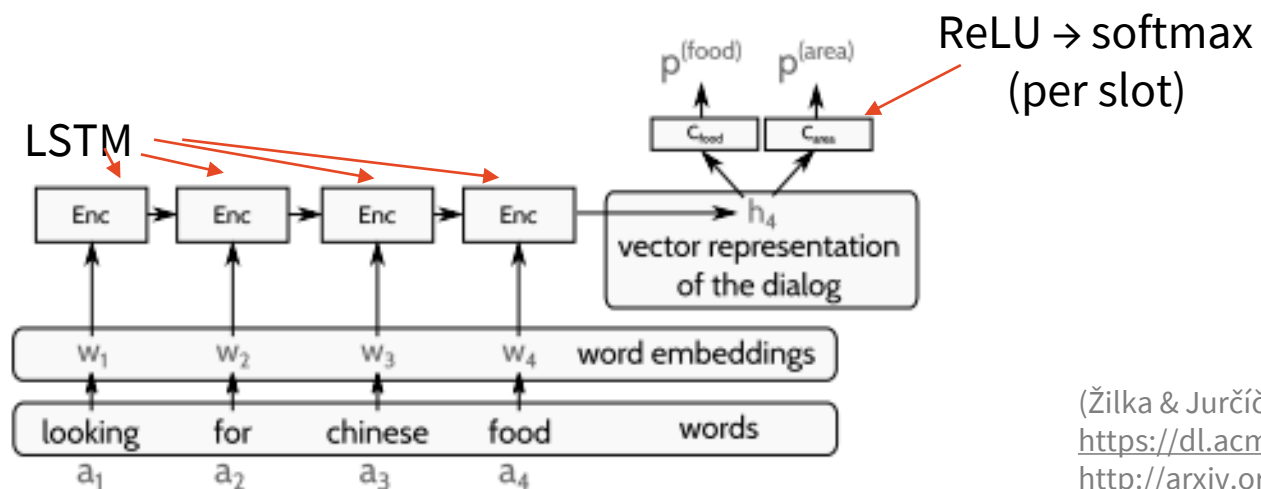


$$\begin{aligned} \mathbb{P}(s = v) &= e^{E(t, v)} / Z \\ \mathbb{P}(s \notin S_{t, s}) &= e^B / Z \\ Z &= e^B + \sum_{v' \in S_{t, s}} e^{E(t, v')} \end{aligned}$$

(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)
<https://dl.acm.org/citation.cfm?id=2955040>
<http://arxiv.org/abs/1507.03471>

Summary



- 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

Thanks



Contact us:

odusek@ufal.mff.cuni.cz

hudecek@ufal.mff.cuni.cz

Slack

Get these slides here:

<http://ufal.cz/npfl123>

References/Inspiration/Further:

- Filip Jurčiček's slides (Charles University): <https://ufal.mff.cuni.cz/~jurcicek/NPFL099-SDS-2014LS/>
- Milica Gašić's slides (Cambridge University): <http://mi.eng.cam.ac.uk/~mg436/teaching.html>
- Henderson (2015): Machine Learning for Dialog State Tracking: A Review <https://ai.google/research/pubs/pub44018>
- Žilka et al. (2013): Comparison of Bayesian Discriminative and Generative Models for Dialogue State Tracking <https://aclweb.org/anthology/W13-4070> (+David Marek's MSc. thesis <https://is.cuni.cz/webapps/zzp/detail/122733/>)
- Liu & Lane (2016): Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling <http://arxiv.org/abs/1609.01454>
- Kim & Banchs (2014): Sequential Labeling for Tracking Dynamic Dialog States <https://www.aclweb.org/anthology/W14-4345>