

NPFL123 Dialogue Systems

# 4. Language Understanding (non-neural)

<https://ufal.cz/npfl123>

**Ondřej Dušek**, Patrícia Schmidtová, Vojtěch Hudeček & Jan Cuřín

6. 3. 2023



Charles University  
Faculty of Mathematics and Physics  
Institute of Formal and Applied Linguistics



unless otherwise stated

# Natural Language Understanding

- **words → meaning**
  - whatever “meaning” is – can be different tasks
  - typically structured, explicit representation
- alternative names/close tasks:
  - **spoken language understanding**
  - **semantic decoding/parsing**
- integral part of dialogue systems, also explored elsewhere
  - stand-alone semantic parsers
  - other applications:
    - human-robot interaction
    - question answering
    - machine translation (not so much nowadays)

# NLU Challenges

- non-grammaticality

*find something cheap for kids should be allowed*

- disfluencies

- hesitations – pauses, fillers, repetitions
- fragments
- self-repairs (~6%!)

*uhm I want something in the west the west part of town*

*uhm find something uhm something cheap no I mean moderate*

*uhm I'm looking for a cheap*

- ASR errors

*I'm looking for a for a chip Chinese rest or rant*

- synonymy

- out-of-domain utterances

*Chinese city centre*

*uhm I've been wondering if you could find me  
a restaurant that has Chinese food close to  
the city centre please*

*oh yeah I've heard about that place my son was there last month*

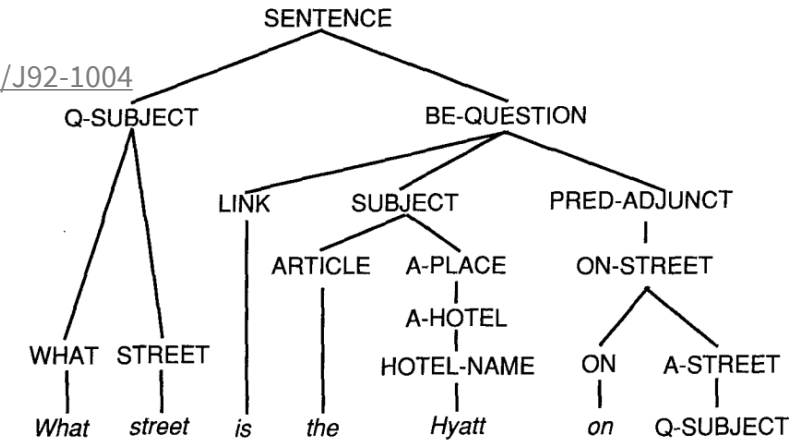
# Semantic representations

- syntax/semantic **trees**
  - typical for standalone semantic parsing
  - different variations
- **frames**
  - technically also trees, but not directly connected to words
  - (mostly older) DSs, some standalone parsers
- **graphs** (AMR)
  - more of a toy task, but popular
- **dialogue acts** = intent + slots & values
  - flat – no hierarchy
  - most DSs nowadays

inform(date=Friday, stay="2 nights")

(Seneff, 1992)

<https://www.aclweb.org/anthology/J92-1004>

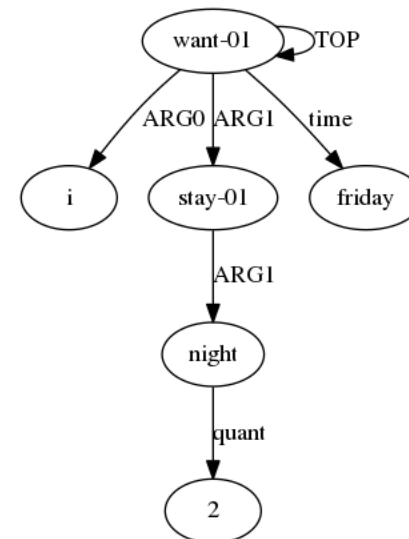


*oui l'hôtel don't le prix ne dépasse pas cent dix euros*

response:	oui
refLink:	co-ref.
	singular
BDObject:	hotel
	room
payment:	amount
	comparative: less
	integer: 110
	unit: euro

(Bonneau-Maynard et al., 2005)

[https://www.isca-speech.org/archive/interspeech\\_2005/i05\\_3457.html](https://www.isca-speech.org/archive/interspeech_2005/i05_3457.html)



I want to stay 2 nights from Friday .

(Damonte et al., 2017)

<https://www.aclweb.org/anthology/E17-1051/>

# NLU basic approaches

For trees/frames/graphs:

- **grammar-based parsing**
  - handwritten/probabilistic grammars & chart parsing algorithms
- **statistical**
  - inducing structure using machine learning
  - grammar is implicit (training treebanks)

For DAs (shallow parsing):

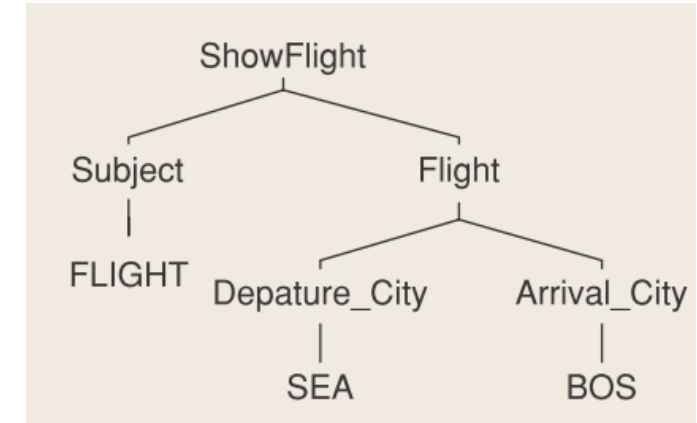
- **classification**
- **sequence labelling**

# Grammars vs. shallow parsing

## Grammars are:

- more expressive
  - hierarchical structure better captures relations
- harder to maintain
  - sparser
  - harder to build rules by hand
  - statistical parsers need more data
  - training data is harder to get
- more hardware-hungry
  - chart parsing:  $O(n^3)$ , shallow:  $O(n)$  for simplest approaches
- more brittle
  - shallow parsing is typically less sensitive to ASR errors, variation, etc.

*Show me flights from Seattle to Boston*



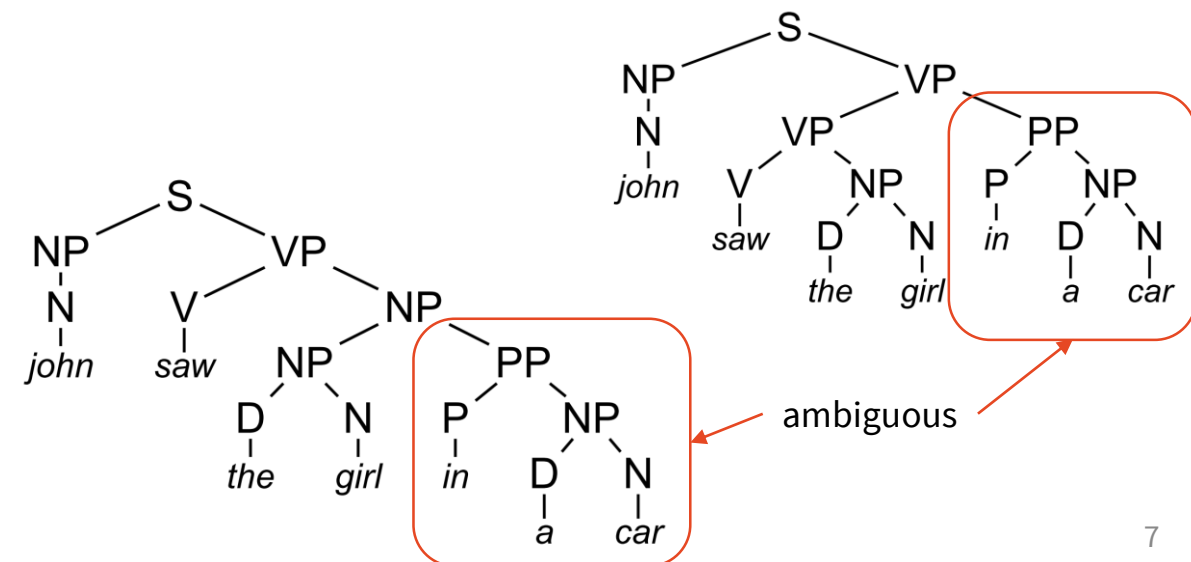
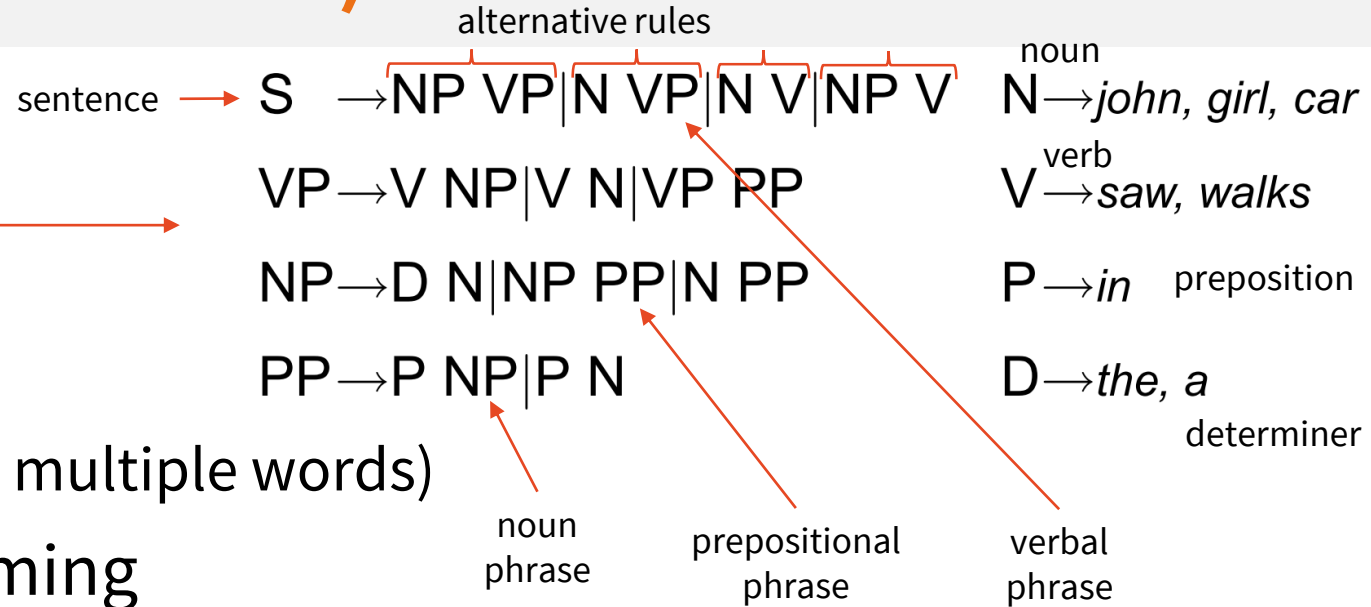
(Wang et al., 2005)

<http://ieeexplore.ieee.org/document/1511821/>

`inform(from=SEA, to=BOS)`

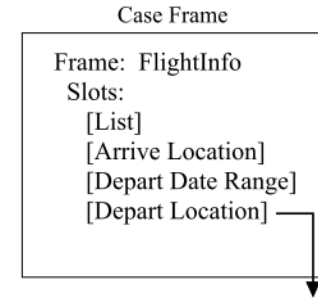
# Grammars: CFG (Context-free Grammar)

- Simple recursive grammar
  - **rules:**  $X \rightarrow A B C$ 
    - splitting a phrase into adjacent parts
  - **terminals** = words
  - **non-terminals** = phrases (spanning multiple words)
- parsable using dynamic programming
  - (chart parsing)
- too simple for full natural language
  - but may be OK for a limited domain
  - especially with **probabilistic extensions**



# CFG: Phoenix Parser (ATIS, 90's)

- CFG hierarchy based on semantic frames
  - Frames → slots / other frames
  - multiple CFGs, one per slot
- Robustness attempts
  - ignore stuff not belonging to any frame
- Chart parsing
  - left to right
  - maximize coverage
  - minimize # of different slots

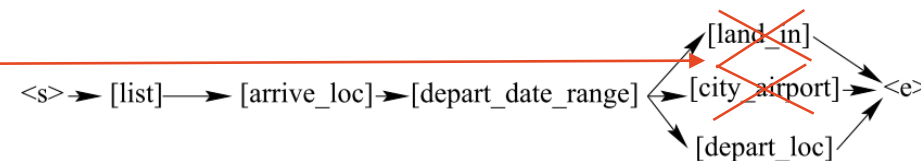


[Depart Location]	→ LEAVE from ENT
LEAVE	→ leaving   departing   ∅
ENT	→ <city>   <airport>

*I would like to go to Boston tomorrow from San Francisco*

```
[List] ( I WOULD LIKE TO )
[Arrive Location] ( GO TO [arrive_loc] ( [city ( [cityname] ( BOSTON ))) )
[Depart Date Range] ( [depart_date_range] ( [on_date] ( [date]
    ( [day_of_week] ( [dayname] ( TOMORROW ))))) )
[Depart Location] ( FROM [depart_loc] ( [city] ( [cityname] ( SAN FRANCISCO ))) )
```

all networks matching  
a span are added to parse chart,  
they're pruned afterwards





# NLU as classification

- using DAs – treating them as a **set of semantic concepts**
  - concepts:
    - intent
    - slot-value pair
  - binary classification: is concept Y contained in utterance X?
  - independent for each concept
- consistency problems
  - no conflicting intents (e.g. *affirm* + *negate*)
  - no conflicting values (e.g. *kids-allowed=yes* + *kids-allowed=no*)
  - need to be solved externally, e.g. based on classifier confidence

# NLU as classification

- classification:  
features → labels (classes)
  - here: classes are **binary** (-1/1 or 0/1)
  - **one classifier per concept**
- features
  - **binary** – is X present?  
or **count** – how many X's are present?
  - words
  - n-grams
  - word pairs/triples  
(position-independent)
  - regex
  - presence of named entities

*I'm looking for something cheap in the city centre.*

	Dialogue act types	Slot value pairs
Classes:	negate ✗	food=Italian ✗
	deny ✗	food=Chinese ✗
	inform ✓	area=centre ✓
	select ✗	area=north ✗
	•	price=cheap ✓
	•	•
	•	•

(from Milica Gašić's slides)

# NER + delexicalization

Approach:

**1) identify** slot values/named entities

**2) delexicalize** = replace them with placeholders (indicating entity type)

- or add the NE tags as more features for classification
- generally needed for NLU as classification
  - otherwise in-domain data is too sparse
  - this can vastly reduce the number of concepts to classify & classifiers
- NER is a problem on its own
  - but general-domain NER tools may need to be adapted
    - added gazetteers with in-domain names
  - in-domain gazetteers alone may be enough
  - NE supplemented by NE linking/disambiguation (usually not needed in DS)

*What is the phone number for Golden Dragon?*

*What is the phone number for **<restaurant-name>**?*

*I'm looking for a Japanese restaurant in Notting Hill.*

*I'm looking for a **<food>** restaurant in **<area>**.*

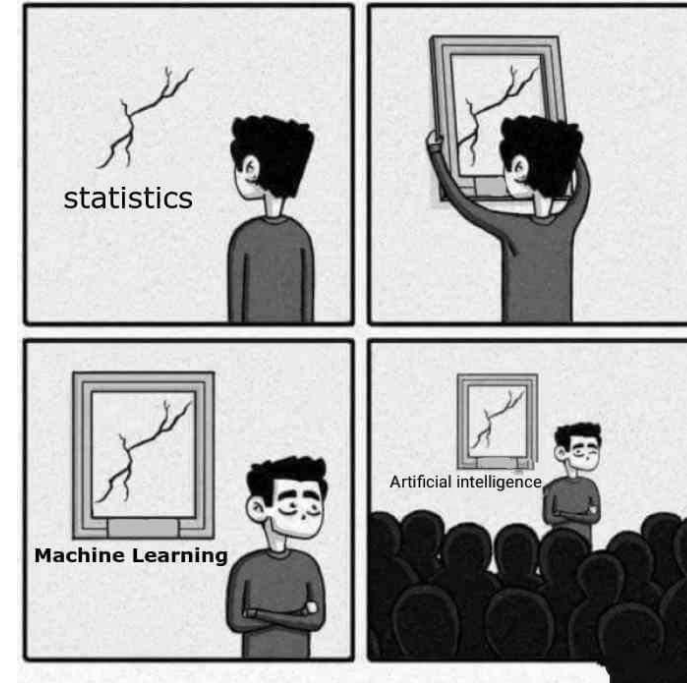
# NLU Classifiers

- note that data is usually scarce!
- **handcrafted / rules**
  - simple mapping: word/n-gram/regex match → concept
  - can work really well for a limited domain
  - no training data, no retraining needed (tweaking on the go)
- **logistic regression**
- **SVM** (support vector machine)
- **neural nets**
  - different, “automatic” features (embeddings, see later)
  - only applicable if a lot of data is available

# Machine Learning (Grossly Oversimplified)

ML is basically function approximation

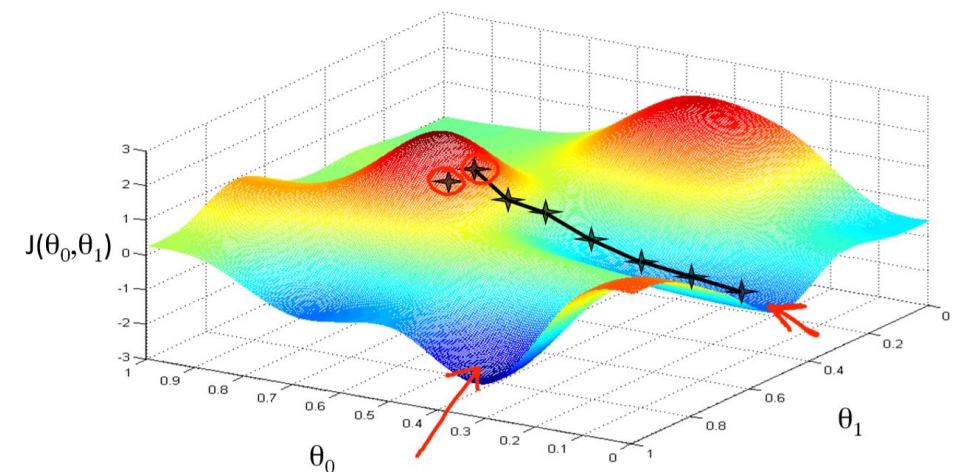
- function: data (**features**) → **labels**
  - discrete labels = **classification**
  - continuous labels = **regression**
- function shape
  - this is where different algorithms differ
  - neural nets: complex functions, composed of simple building blocks (linear, sigmoid, tanh...)
- **training/learning** = adjusting function parameters to minimize error
  - **supervised** learning = based on data + labels given in advance
  - **reinforcement** learning = based on exploration & rewards given online



<https://towardsdatascience.com/no-machine-learning-is-not-just-glorified-statistics-26d3952234e3>

# Machine Learning (Grossly Oversimplified)

- training– **gradient descent** methods
  - minimizing a **cost function**  
(notion of error – given system output, how far off are we?)
  - calculus: derivative = steepness/slope
  - follow the slope to find the minimum – derivative gives the direction
  - **learning rate** = how fast do we go (needs to be tuned)
- gradient typically computed over **mini-batches**
  - random bunches of a few training instances
  - not as erratic as using just 1 instance, not so slow as computing over whole data
  - **stochastic gradient descent**
  - improvements: AdaGrad, Adam [...]
    - cleverly adjusting the learning rate

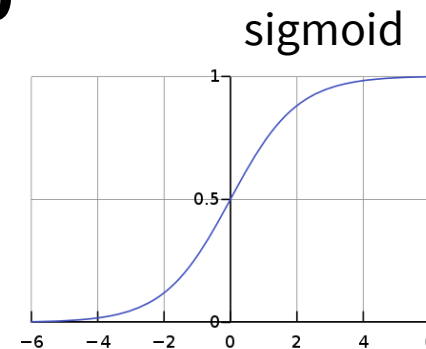


# Logistic Regression (LR, also called Maximum Entropy Classifier)

- modeling using the sigmoid (logistic) function with parameters  $\theta$

target class is binary, i.e.  $y \in \{-1, +1\}$

$$p(y|\mathbf{x}) = \text{sigmoid}(-y(\theta \cdot \mathbf{x})) = \frac{1}{1 + \exp(-y(\theta \cdot \mathbf{x}))}$$



equivalent form  
– maximum entropy style  
(works for **multiclass**, too!)

$$p(y|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp(\theta \cdot \mathbf{f}(\mathbf{x}, y))$$

normalization

input data/features

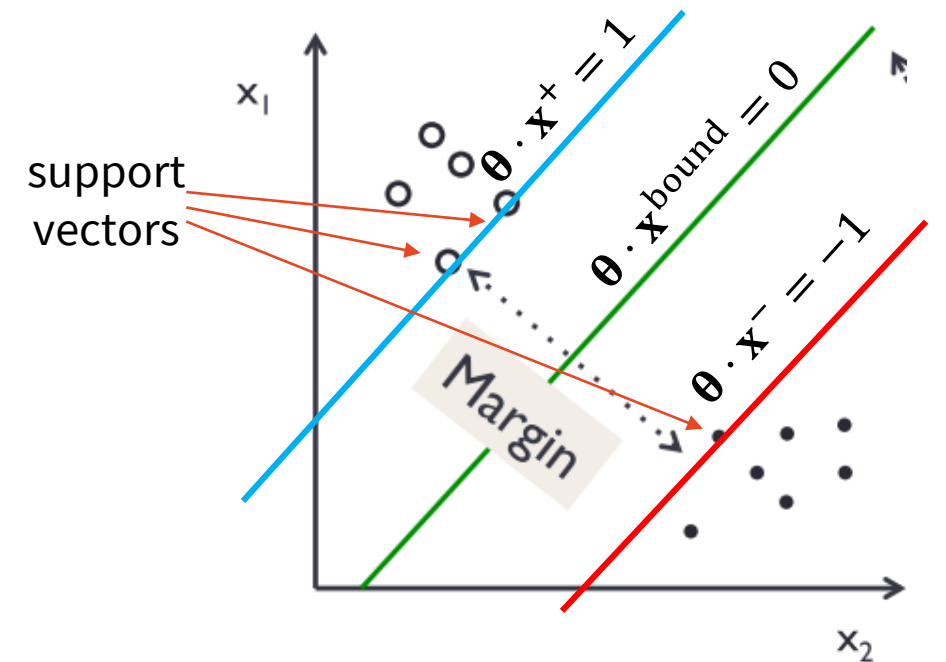
generalization: **feature functions** vector  
(some fire for each value of  $y$ )

- despite the name, it's a classifier
- very basic, but powerful with the right features
- trained by gradient descent (logistic/cross entropy loss)
- maximum entropy estimate (“most uniform model given data”)

# Support-Vector Machines (SVMs)

- geometric intuition: features ~ coordinates in multidimensional space
- trying to separate classes with a hyperplane (**decision boundary**)
- idea: let's find a boundary with **maximum margin**
  - i.e. maximize distance between classes → best generalization
    - most likely to classify new example correctly
  - this boundary is given by **support vectors** (instances that are closest to it)
- margin width is  $\frac{2}{\|\theta\|}$  → we minimize  $\|\theta\|^2$
- SVM score:  $g(\mathbf{x}) = \theta \cdot \mathbf{x}$ 
  - 0 at the boundary, +1/-1 for support vectors
  - sign of the score gives the class (positive/negative)

(from Aikaterini Tzompanaki's slides)



$x_1, x_2$  = features      o = positive class

• = negative class



# SVM vs. Logistic Regression

- **soft-margin SVM** – for non-separable cases

- non-separable = messy data, can't separate with a hyperplane
- “soft” = weighing correct classification (**hinge loss**) & margin size

- model:  $\min_{\theta} \lambda \|\theta\|^2 + \sum_i \max\{0, 1 - y_i \theta \cdot \mathbf{x}_i\}$   
regularization  $\lambda$  weight  $\theta$

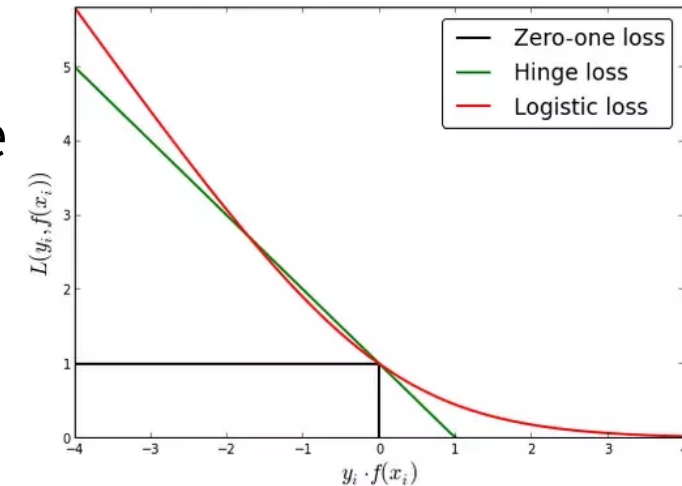
- **regularized logistic regression** – for better generalization

- preventing overfitting to training data – trying to keep parameter values low
- **logistic loss**

- model:  $\min_{\theta} \lambda \|\theta\|^2 + \sum_i \log(1 + \exp(1 - y_i \theta \cdot \mathbf{x}_i))$

- the main difference is the loss

- hinge loss should be marginally better for classification, but it depends



# Classification example

## features (x)

I	1
want	1
to	3
go	1
from	2
<airport-1>	1
...	
him	0
price	0
tell	0
...	
I want	1
want to	1
to go	1
....	
from <airport-1>	1

ASR: *I want to go from from Newark to London City next Friday*

Delex: *I want to go from from <airport-1> to <airport-2> next <day-1>*

## weights:

intent=search\_flights

$\theta_{SF}$

intent=request\_price

$\theta_{RP}$

...

from\_airport=<airport-1>

$\theta_{FA1}$

....

weights define  
different classifiers

SVM:  $\theta_{FA1} \cdot \mathbf{x} = +3.4347$

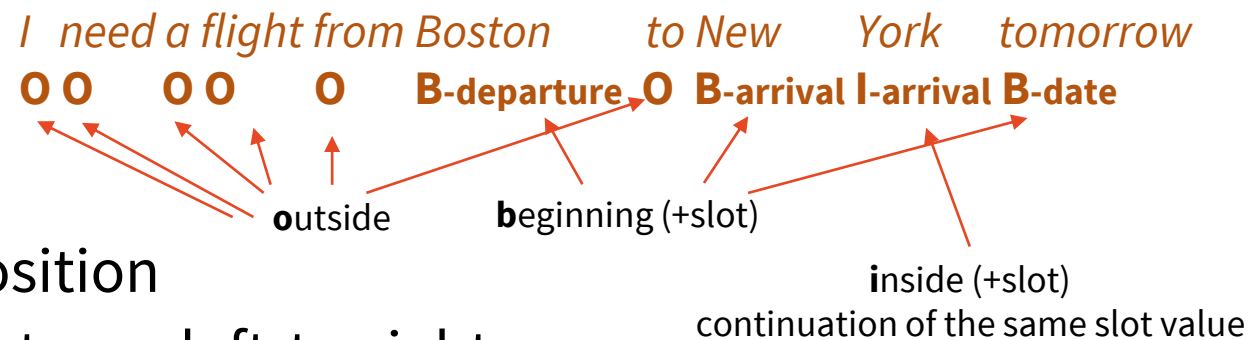
→ found from\_airport=Newark

LR:  $\text{sigmoid}(\theta_{FA1} \cdot \mathbf{x}) = 0.883$

→ found from\_airport=Newark (conf. = 0.883)

# Slot filling as sequence tagging

- get slot values directly – “automatic” delexicalization
  - each word classified
  - classes = slots & **IOB format** (inside-outside-beginning)
  - slot values taken from the text (where a slot is tagged)
  - NER-like approach
- rules + classifiers kinda still work
  - a) keywords/regexes found at specific position
  - b) apply classifier to each word in the sentence left-to-right
  - problem: overall consistency
    - slots found elsewhere in the sentence might influence what’s classified now
- solution: **structured/sequence prediction**



# Maximum Entropy Markov Model (MEMM)

- Looking at past classifications when making next ones
  - LR + a simple addition to the feature set
- Whole history would be too sparse/complex
  - **Markov assumption**: only the most recent matters
    - 1<sup>st</sup> order MM: just the last one (←this is what we show here)
    - $n^{\text{th}}$  order MM:  $n$  most recent ones
- still not modelling the sequence globally

$$p(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^T \frac{1}{Z(y_{t-1}, \mathbf{x})} \exp(\boldsymbol{\theta} \cdot \mathbf{f}(y_t, y_{t-1}, \mathbf{x}))$$

for the whole sequence

time steps are independent except for  $y_{t-1}$

$y_{t-1}$  is the main addition compared to LR

looking at the whole input

# Hidden Markov Model (HMM)

- Modelling the **sequence as a whole**
- Very basic model:
  - “**tag depends on word + previous tag**”
- Markov assumption, again
- “Hidden” – reverse viewpoint:
  - “tags are hidden, but they influence the words on the surface”
- Inference – Viterbi algorithm
  - we can get the **globally best tagging**

HMM is a **generative model** – models **joint distribution**  $p(\mathbf{y}, \mathbf{x})$ , not just conditional  $p(\mathbf{y}|\mathbf{x})$

$$p(\mathbf{y}, \mathbf{x}) = \prod_{t=1}^T \underbrace{p(y_t | y_{t-1})}_{\substack{\text{transition} \\ \text{probability} \\ \text{prev. tag} \rightarrow \text{tag}}} \underbrace{p(x_t | y_t)}_{\substack{\text{observation} \\ \text{probability} \\ \text{tag} \rightarrow \text{word}}}$$

for the whole sequence

# HMM vs. MEMM

- MEMM:
  - any feature functions, as in LR
  - local normalization – does not model whole sequences, just locally
  - **label bias** problem
    - training: you know the correct labels
    - inference: one error can lead to a series of errors
- HMM:
  - global normalization for  $p(\mathbf{y}|\mathbf{x})$  over all  $\mathbf{y}$ 's
    - modelling sequences as a whole
  - **very** boring & limited feature functions
- how about best of both?

# Linear-Chain **Conditional Random Field (CRF)**

- HMM + more complex feature functions
- MEMM + global sequence modelling

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^T \exp(\boldsymbol{\theta} \cdot \mathbf{f}(y_t, y_{t-1}, \mathbf{x}))$$

global normalization  
(otherwise like MEMM)

feature functions  
looking at whole input  
(otherwise looks like HMM)

- state-of-the art for many sequence tagging tasks (incl. NLU)
  - until NNs took over
  - used also in conjunction with NNs
- global normalization makes it slow to train

# Sequence tagging example

ASR: *I want to go from from Newark to London City next Friday*  
Previous tags: **0 0 0 0 0 0 B-from\_airport 0**

**current position:**  
what's the class  
for *London*?

## features (x):

<i>in_sent</i> =I	1	<i>cur</i> =London	1	<i>prev_tag</i> =O	1
<i>in_sent</i> =want	1	<i>cur</i> =him	0	<i>prev_tag</i> =B-price	0
<i>in_sent</i> =to	3	...	...	...	...
<i>in_sent</i> =go	1	<i>prev</i> =to	1		
...		<i>prev</i> =want	0		
<i>in_sent</i> =him	0	<i>prev</i> =price	0		
<i>in_sent</i> =price	0	...			
...		<i>cur</i> =to London	1		
<i>in_sent</i> =I want	1	<i>prev</i> =Newark to	1		
<i>in_sent</i> =want to	1	...			
<i>in_sent</i> =to go	1				

using  $y_{t-1}$

**HMM** considers only these

**MEMM:** looks at *London*, ignores  
that it also needs to tag *City* later  
→ likely to tag as B-to\_city

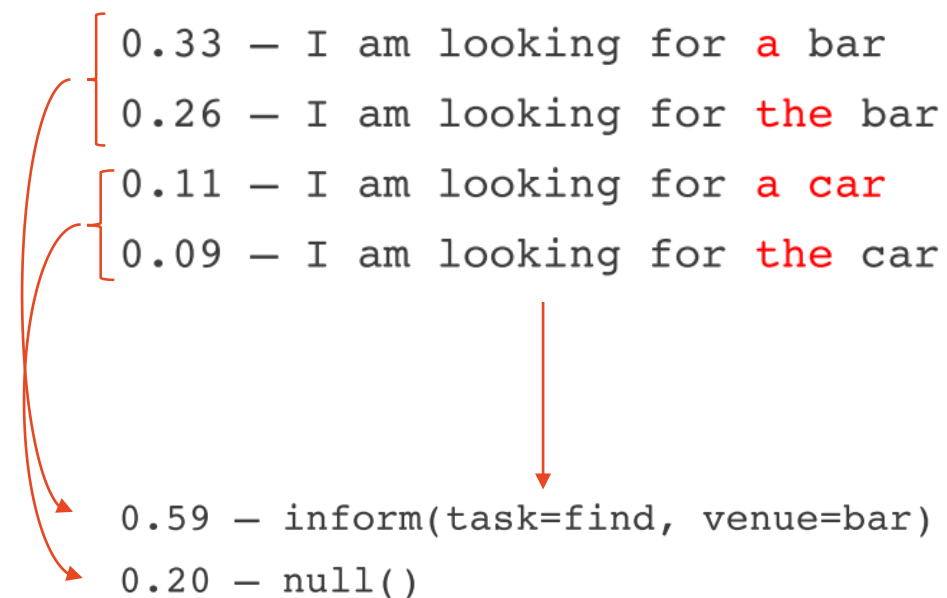
**CRF:** also considers future tags,  
more likely to tag *London City*  
as B-to\_airport I-to\_airport



# 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**

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

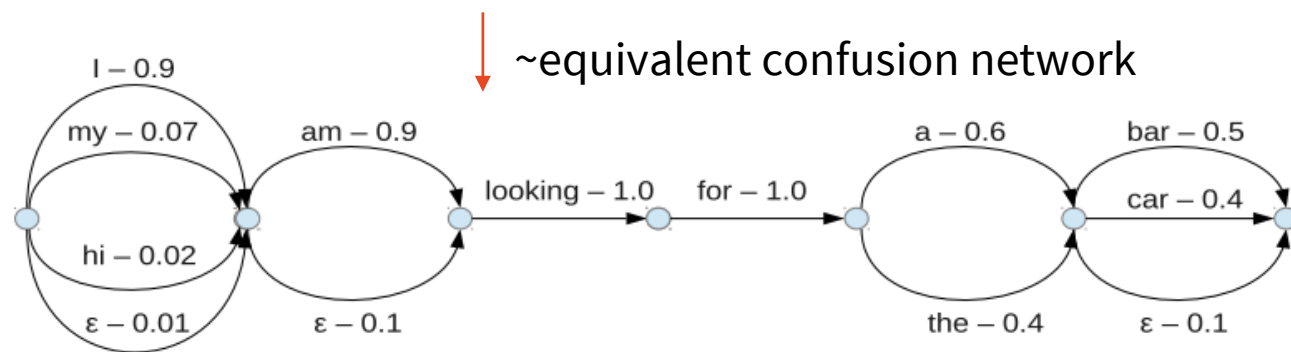


(from Filip Jurčiček's slides)

# 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 n-best list  
0.26 – I am looking for the bar  
0.11 – I am looking for a car  
0.09 – I am looking for the car



(from Filip Jurčiček's slides)

## features:

I	0.9
hi	0.02
am	0.9
looking	1
for	1
...	
I am	0.81
my am	0.063
am looking	0.9
a bar	0.3
a car	0.24
...	

↑  
should be normalized  
by n-gram length

# Context

- 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

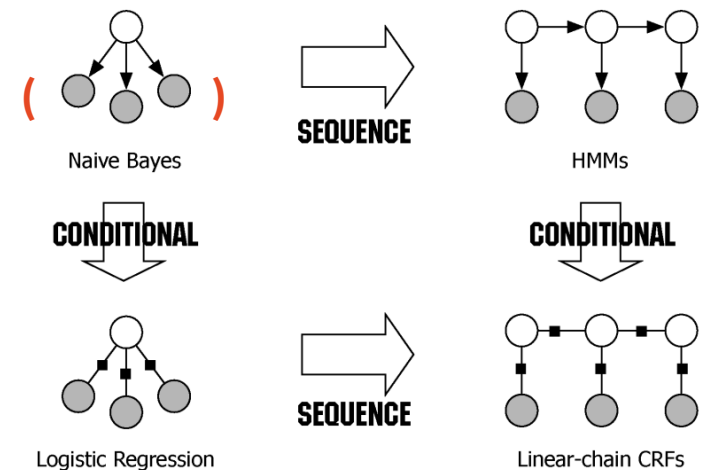
*U: I'm looking for flights from JFK.  
S: Where would you like to go?  
U: Atlanta.*

*inform(??=Atlanta)  
inform(**to\_city**=Atlanta)*

*x U: Actually I'd rather fly from Newark.*

# Summary

- NLU can be tricky
  - bad grammar, fragments, synonymy, ASR errors ...
- Grammars, frames, graph representation
  - rule-based or statistical structure induction
  - more expressive, but harder – not so much in limited-domain systems
- Shallow parsing
  - dialogue acts: intent + slots & labels
  - rules – keyword spotting, regex
  - classification (LR, SVM)
  - sequence tagging (MEMM, HMM, CRF)
- Next time: neural NLU & dialogue state tracking



(Sutton & McCallum, 2010)  
<https://arxiv.org/abs/1011.4088>

## Contact us:

<https://ufaldsg.slack.com/>

[{odusek,schmidtova,hudecek}@ufal.mff.cuni.cz](mailto:{odusek,schmidtova,hudecek}@ufal.mff.cuni.cz)

Skype/Meet/Zoom (by agreement)

## Labs in 10 mins

## Get the slides here:

<http://ufal.cz/npfl123>

## References/Inspiration/Further:

- Milica Gašić's slides (Cambridge University): <http://mi.eng.cam.ac.uk/~mg436/teaching.html>
- Raymond Mooney's slides (University of Texas Austin): <https://www.cs.utexas.edu/~mooney/ir-course/>
- Filip Jurčiček's slides (Charles University): <https://ufal.mff.cuni.cz/~jurcicek/NPFL099-SDS-2014LS/>
- Hao Fang's slides (University of Washington): [https://hao-fang.github.io/ee596\\_spr2018/syllabus.html](https://hao-fang.github.io/ee596_spr2018/syllabus.html)
- Aikaterini Tzompanaki's slides (University of Cergy-Pontoise): <https://perso-etis.ensea.fr/tzompanaki/teaching.html>
- Pierre Lison's slides (University of Oslo): <https://www.uio.no/studier/emner/matnat/ifi/INF5820/h14/>
- Sutton & McCallum – Introduction to Conditional Random Fields: <https://arxiv.org/abs/1011.4088>
- Andrew McCallum's slides (U. of Massachusetts Amherst): <https://people.cs.umass.edu/~mccallum/courses/inlp2007/>

# Hidden Markov Model vs. MEMM (additional explanation, just FYI, not required)

- Rewrite HMM so it looks more like MEMM + get conditional probability

just indicators (binary features)

transition

observation

hide the actual probabilities as weights (in logarithm)

$$p(\mathbf{y}, \mathbf{x}) = \prod_{t=1}^T \exp\left(\sum_{i,j \in \mathcal{S}} \theta_{ij} \mathbf{1}_{y_t=i} \mathbf{1}_{y_{t-1}=j} + \sum_{i \in \mathcal{S}} \sum_{o \in \mathcal{O}} \mu_{oi} \mathbf{1}_{y_t=i} \mathbf{1}_{x_t=o}\right)$$

subsume transition & observation under **feature functions**,  $\theta_k$  is  $\theta_{ij}$  &  $\mu_{oi}$

conditional probability:

just the current word

$$p(\mathbf{y}, \mathbf{x}) = \prod_{t=1}^T \exp\left(\sum_{k=1}^K \theta_k f_k(y_t, y_{t-1}, x_t)\right)$$

vector notation

$$p(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{y}, \mathbf{x})}{\sum_{\mathbf{y}'} p(\mathbf{y}', \mathbf{x})} = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^T \exp\left(\sum_{k=1}^K \theta_k f_k(y_t, y_{t-1}, x_t)\right) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^T \exp(\boldsymbol{\theta} \cdot \mathbf{f}(y_t, y_{t-1}, x_t))$$

normalization is global