Dialogue Systems

NPFL123 Dialogové systémy

6. Natural Language Understanding (Non-neural)

Ondřej Dušek & Ondřej Plátek & Jan Cuřín

ufal.cz/npfl123

26. 3. 2019

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
- disfluencies
  - hesitations – pauses, fillers, repetitions
  - fragments
  - self-repairs (~6%!)
- ASR errors
- synonymy
- out-of-domain utterances

find something cheap for kids should be allowed

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

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

Chinese city centre

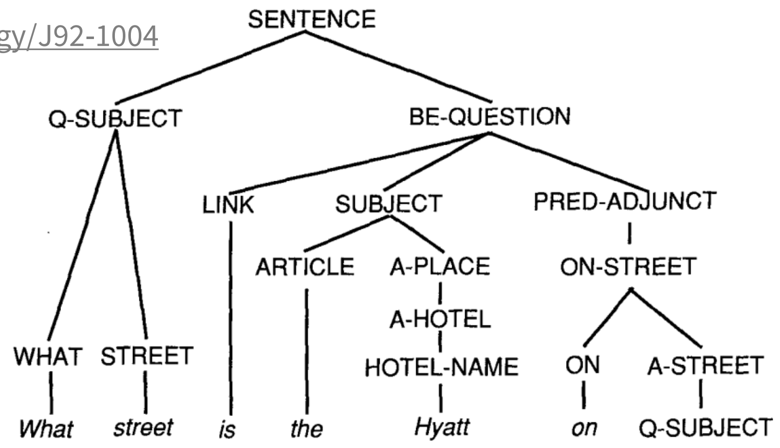
uhm I've been wondering if you could find me a restaurant that has Chinese food close to the city centre please

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

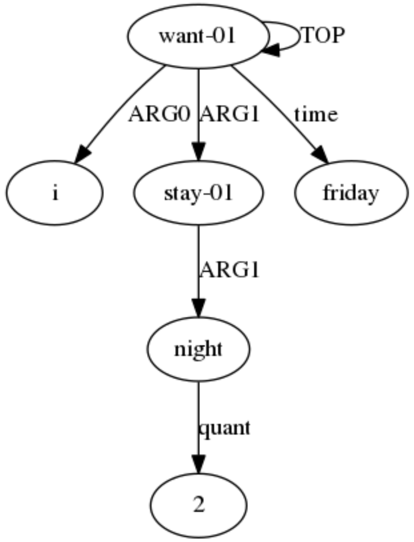
https://www.aclweb.org/anthology/J92-1004



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	

https://www.isca-speech.org/archive/interspeech_2005/i05_3457.html



I want to stay 2 nights from Friday .

http://cohort.inf.ed.ac.uk/amreager.html

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

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):

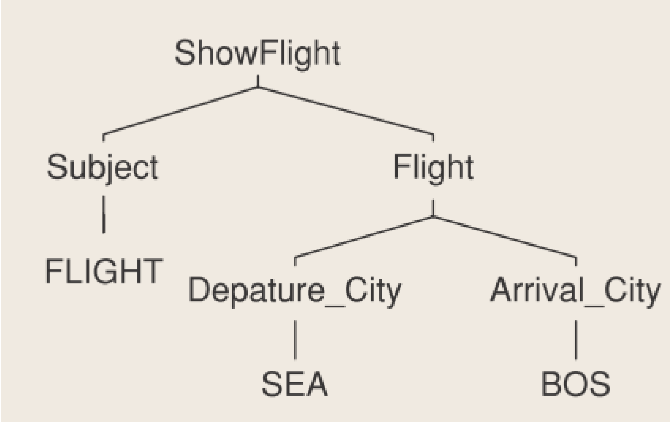
- **classification**
- **sequence labelling**

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



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

inform(from=SEA, to=BOS)

Grammars: CFG
(Context-free Grammar)

- Simple recursive grammar
  - **rules**: X → 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**

sentence — alternative rules

S →NP VP|N VP|N V|NP V

VP→V NP|V N|VP PP

NP→D N|NP PP|N PP

PP→P NP|P N

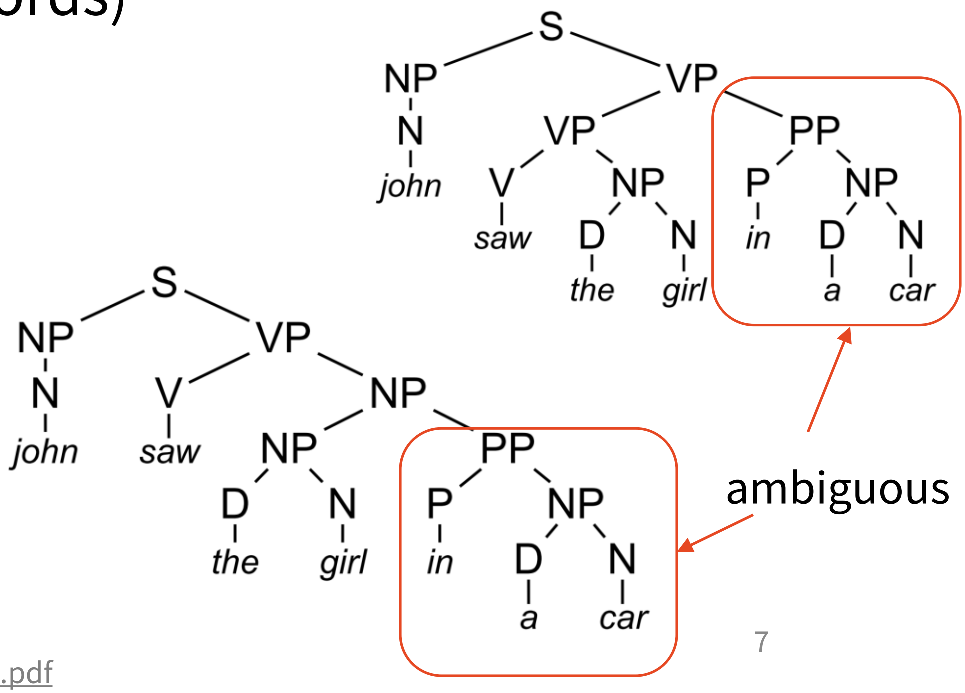
noun phrase — prepositional phrase — verbal phrase

N→*john, girl, car* — noun

V→*saw, walks* — verb

P→*in* — preposition

D→*the, a* — determiner



http://www.dfki.de/compling/pdfs/cfg-slides.pdf

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

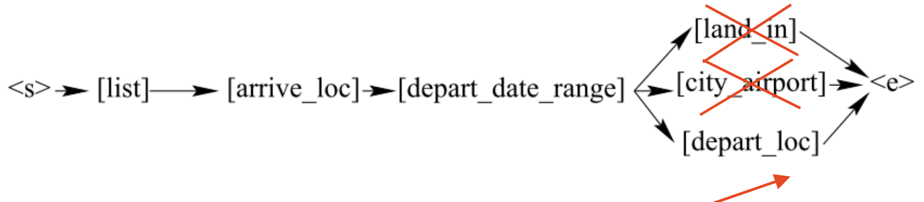
Case Frame

Frame: FlightInfo
Slots:
[List]
[Arrive Location]
[Depart Date Range]
[Depart Location]

[Depart Location] → LEAVE from ENT
LEAVE → leaving | departing | Ø
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))))



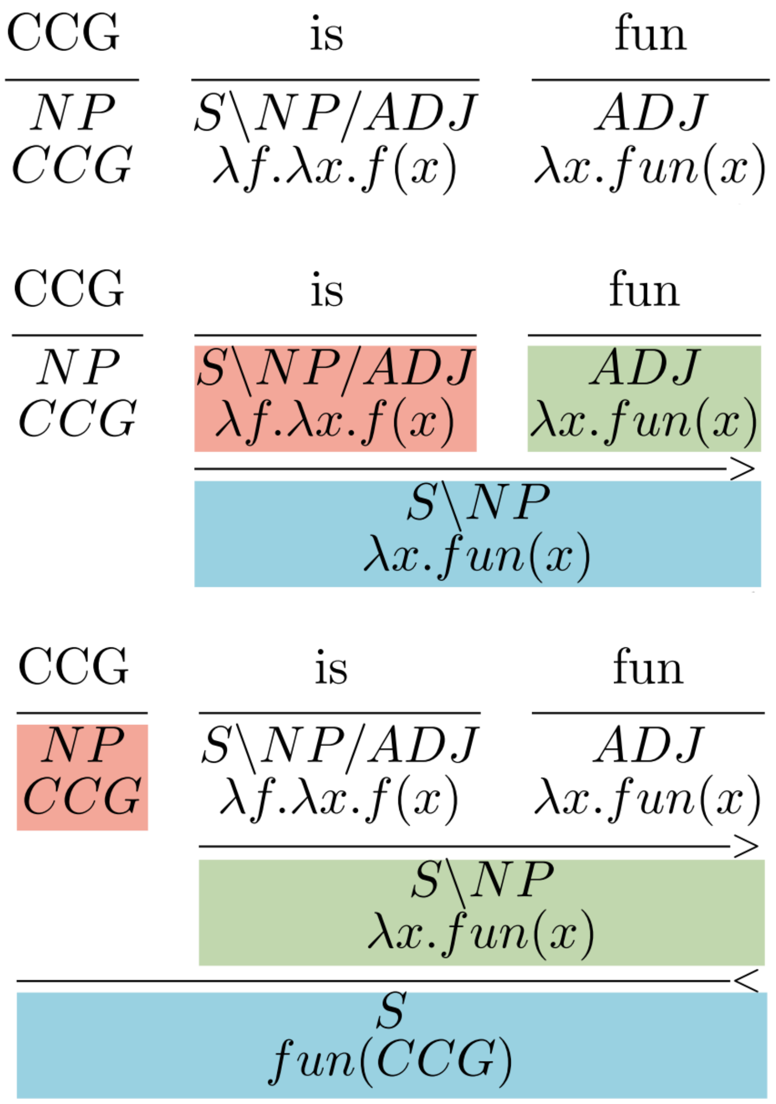
all networks matching
a span added to parse chart,
pruned afterwards

Grammars: CCG
(Combinatory Categorial Grammar)

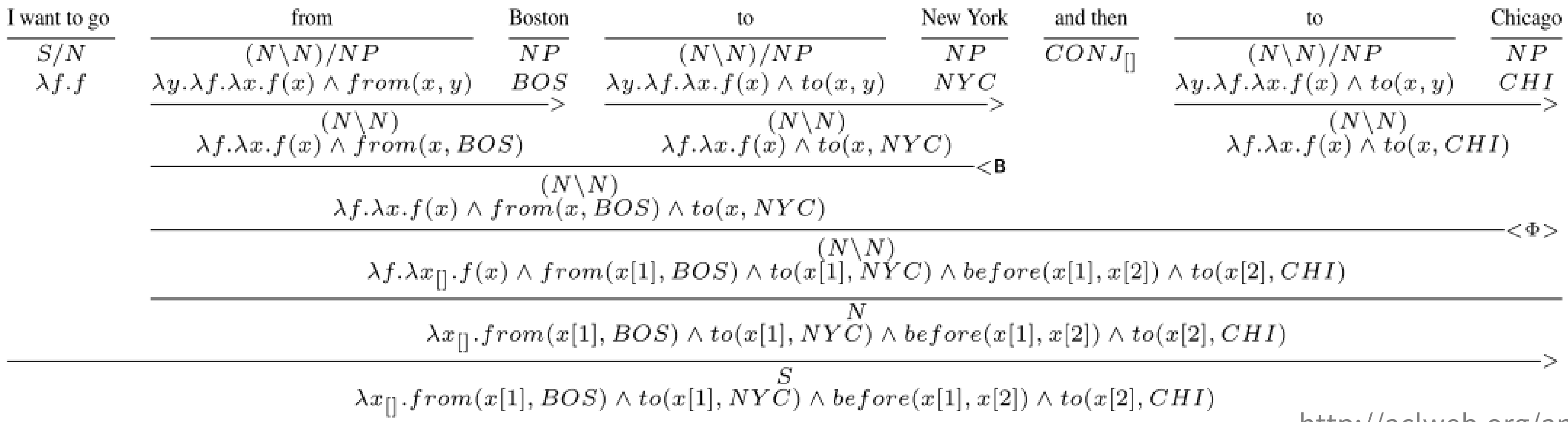
- Grammar based on lambda calculus
  - syntax-bound semantics: lambda meaning in parallel to syntax phrases
- CCG lambda expressions:
  - logical constant: *NYC, BOSTON…*
  - variable: *x, y, z…*
  - literal: *city(AUSTIN), located_in(AUSTIN, TEXAS)*
  - lambda terms – binding variables: *λx.city(x) ~ "x is a city"*
  - quantifiers ∃ ∀, logical operators ∧ ∨ ¬
- CCG categories: syntax + lambda
  - simple: *NOUN : λx.city(x)*
  - complex: *S\NP/NP : λx.f(x)* ("sentence missing an NP to the left and right")
- Lexicon: word + syntax + lambda:
  - *city ⊢ NOUN: λx.city(x), is ⊢ S\NP/NP : λx.f(x)*

Grammars: CCG

- parsing = combining categories (function application)
  - much fewer operations than CFG
    - **>, <** function application – *B : g + A\B : f → A : f(g)*
    - **>B, <B** function composition – *A/B : f + B/C : g → A/C : λx.f(g(x))*
    - **<Φ>** coordination (2 identical categories → 1)
    - **–** category change
  - similar algorithms to CFG
  - statistical parsers available



https://yoavartzi.com/tutorial/



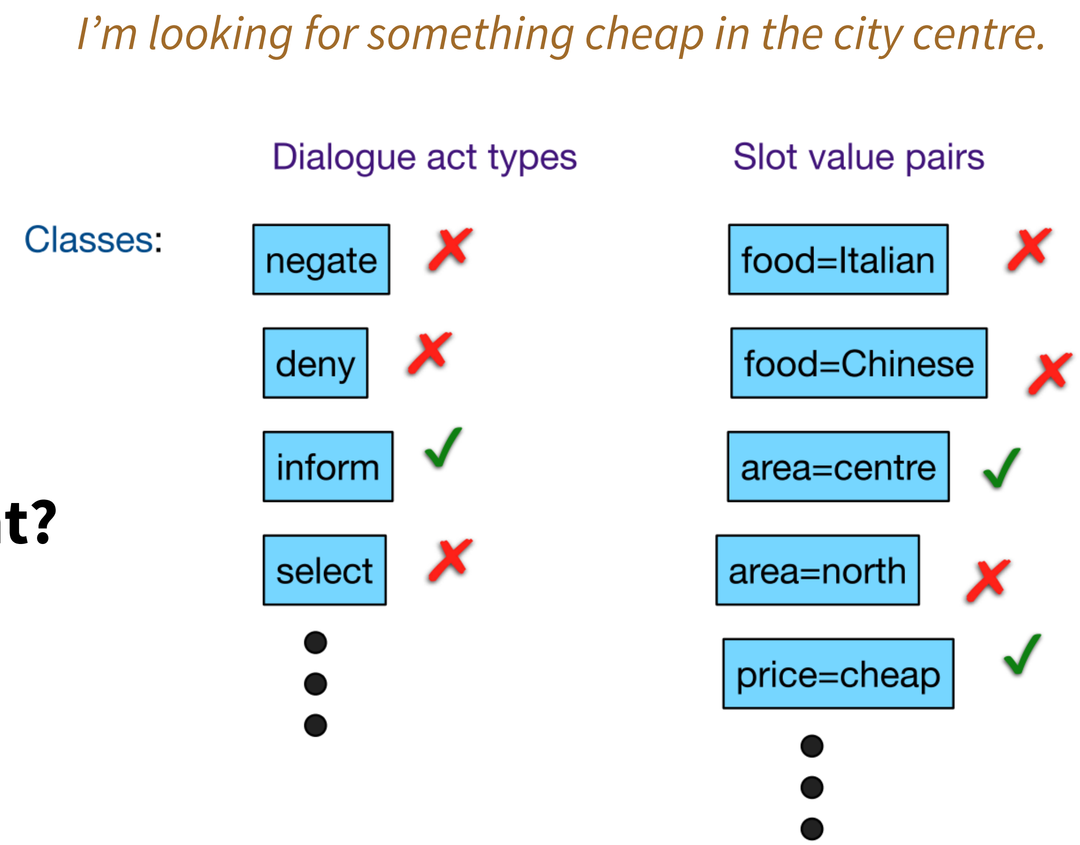
http://aclweb.org/anthology/D11-1039

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



(from Milica Gašić's slides)

NER + delexicalization

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>.

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)

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

Logistic Regression
(Maximum Entropy Classifier)

binary, for $y \in \{-1, +1\}$

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

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

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

normalization

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")

$y \in \{0,1\}$ vs. $\{-1, +1\}$: https://stats.stackexchange.com/questions/229645/
formula equivalence: http://ufal.mff.cuni.cz/~odusek/var/upload/docs/msc_thesis.pdf, page 30

Support-Vector Machines (SVMs)

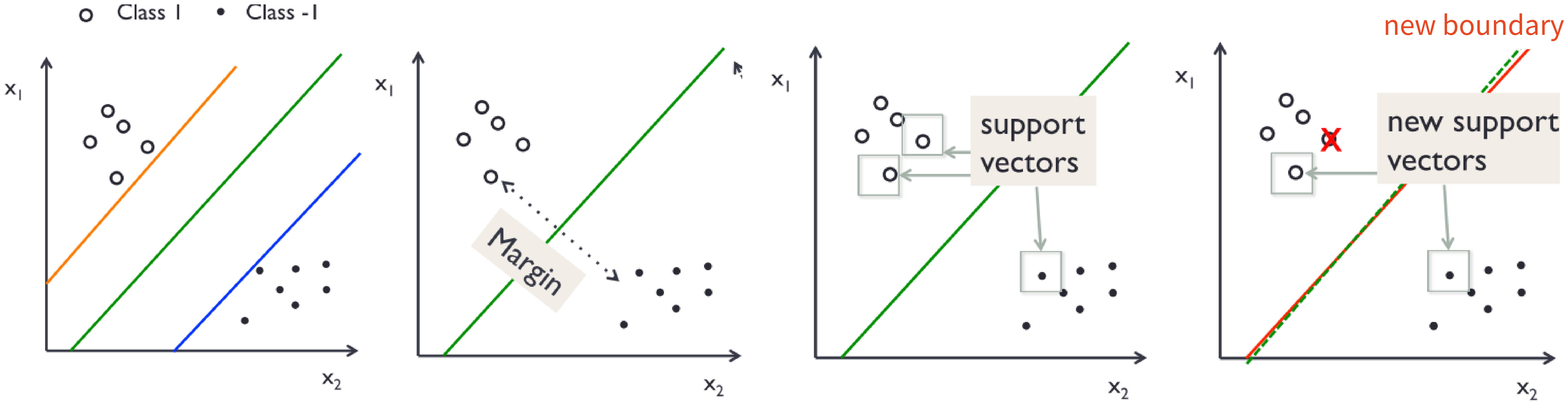
- separate classes with **maximum margin** (=best generalization)
- **decision boundary** defined by **support vectors** (closest instances)

there are many possible separation boundaries between classes in feature space

boundary farthest away from both classes = maximum margin

instances closest to the boundary = **support vectors**

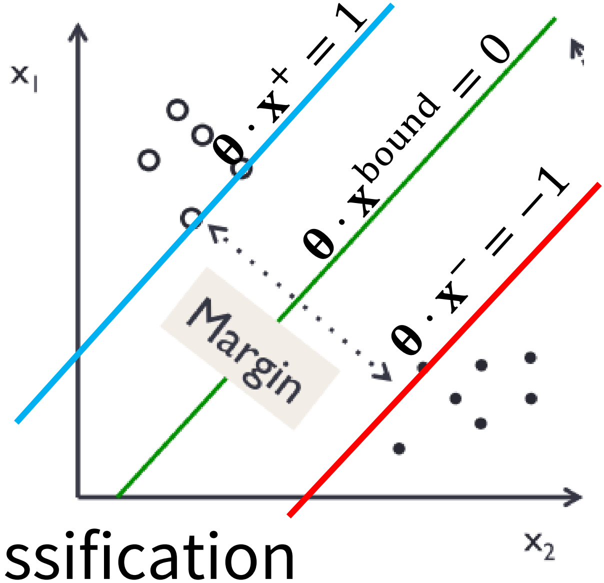
removing a support vector changes the boundary



(from Aikaterini Tzompanaki's slides)

SVMs

margin width is $\frac{2}{||\boldsymbol{\theta}||}$



- Decision boundary: $\boldsymbol{\theta} \cdot \mathbf{x}^{\text{bound}} = 0$
- Support vectors: $\boldsymbol{\theta} \cdot \mathbf{x}^{\text{sv}} = y^{\text{sv}} \;\; (y^{\text{sv}} \in \{-1, +1\})$
- Maximum margin: $\max \frac{2}{||\boldsymbol{\theta}||} \sim \min \frac{1}{2} ||\boldsymbol{\theta}||^2$ with correct classification
  - constrained optimization – quadratic programming (Lagrange multipliers)
- **SVM Score**: $g(\mathbf{x}) = \boldsymbol{\theta} \cdot \mathbf{x} = \sum_{i=1}^{s} y_i \alpha_i \underbrace{\mathbf{x_i} \cdot \mathbf{x}}$

optimal decision boundary

sum over support vectors

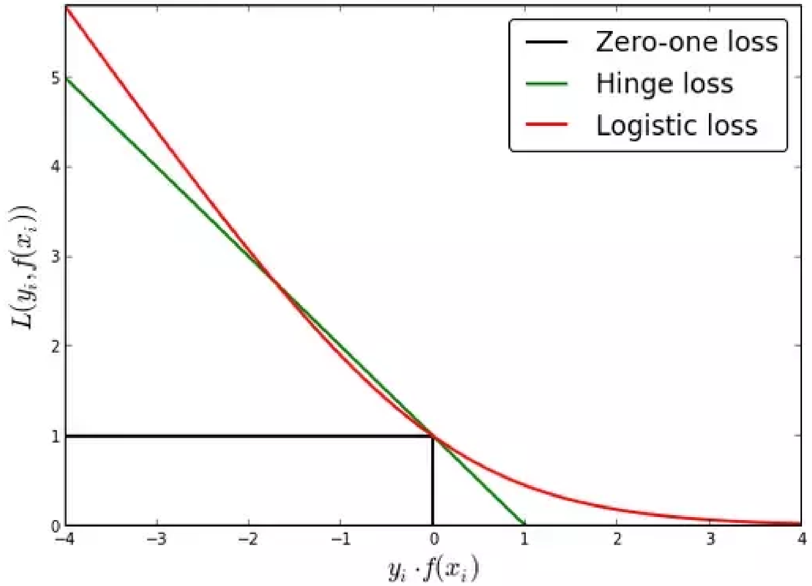
sup. vec. label (-1/+1)

sup. vec. weight in feature space (Lagrange multiplier)

kernel – dot product of features (linear SVM)

- classification:
  - $y = \text{sign}(g(\mathbf{x}))$
- probability: Platt scaling
  - logistic regression with $g(\mathbf{x})$ as feature

SVM vs. Logistic Regression



- **soft-margin SVM** – for non-separable cases
  - non-separable = no perfect decision boundary
  - "soft" = weighing correct classification (**hinge loss**) & margin size
  - model: $\min_{\boldsymbol{\theta}} \lambda ||\boldsymbol{\theta}||^2 + \sum_i \max\{0, 1 - y_i \boldsymbol{\theta} \cdot \mathbf{x}_i\}$

regularization weight

- **regularized logistic regression** – for better generalization
  - preventing overfitting to training data – trying to keep parameter values low
  - **logistic loss**
  - model: $\min_{\boldsymbol{\theta}} \lambda ||\boldsymbol{\theta}||^2 + \sum_i \log(1 + \exp(1 - y_i \boldsymbol{\theta} \cdot \mathbf{x}_i))$
- the main difference is the loss
  - hinge loss should be marginally better for classification, but it depends

https://towardsdatascience.com/support-vector-machine-vs-logistic-regression-94cc2975433f

Classification example

features ($\mathbf{x}$)	
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	$\boldsymbol{\theta}_{\mathrm{SF}}$
intent=request_price	$\boldsymbol{\theta}_{\mathrm{RP}}$
…	
from_airport=<airport-1>	$\boldsymbol{\theta}_{\mathrm{FA1}}$
….	

weights define different classifiers

SVM: $\boldsymbol{\theta}_{\mathrm{FA1}} \cdot \mathbf{x} = +3.4347$ → found from_airport=Newark
LR: $\mathrm{sigmoid}(\boldsymbol{\theta}_{FA1} \cdot \mathbf{x}) = 0.883$ → 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

I	*need*	*a*	*flight*	*from*	*Boston*	*to*	*New*	*York*	*tomorrow*
O	**O**	**O**	**O**	**O**	**B-dept**	**O**	**B-arr**	**I-arr**	**B-date**

- 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
  - 1st order MM: just the last one (←this is what we show here)
  - *n*th order MM: *n* most recent ones
- still not modelling the sequence globally

looking at the whole input

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

for the whole sequence

time steps – independent except for y_{t-1}

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

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 $p(\mathbf{y}|\mathbf{x})$

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

Hidden Markov Model

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

transition

observation

hide the actual probabilities as weights (in logarithm)

just indicators (binary features)

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

subsume transition & observation under **feature functions**, θ_k is θ_{ij} & μ_{oi}

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

just the current word

conditional probability

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

vector notation

normalization is global

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: **O O O O O O B-from_airport O**

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

features ($\mathbf{x}$):

in_sent=I	1	*cur*=London	1	*prev_tag*=O	1
in_sent=want	1	*cur*=him	0	*prev_tag*=B-price	0
in_sent=to	3	…		…	
in_sent=go	1	*prev*=to	1		
…		*prev*=want	0		
in_sent=him	0	*prev*=price	0		
in_sent=price	0	…			
…		*cur*=to London	1		
in_sent=I want	1	*prev*=Newark to	1		
in_sent=want to	1	…			
in_sent=to go	1				

HMM considers only these

using y_{t-1}

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**

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

```
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)

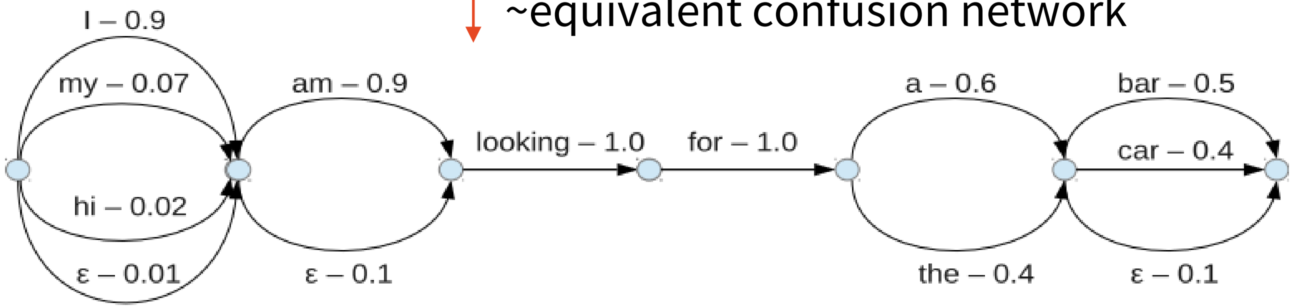
Handling ASR noise

- 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
```

n-best list

~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
am looking	0.9
a bar	0.3
a car	0.24
…	

should be normalized by n-gram length

(from Filip Jurčíček's slides)

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

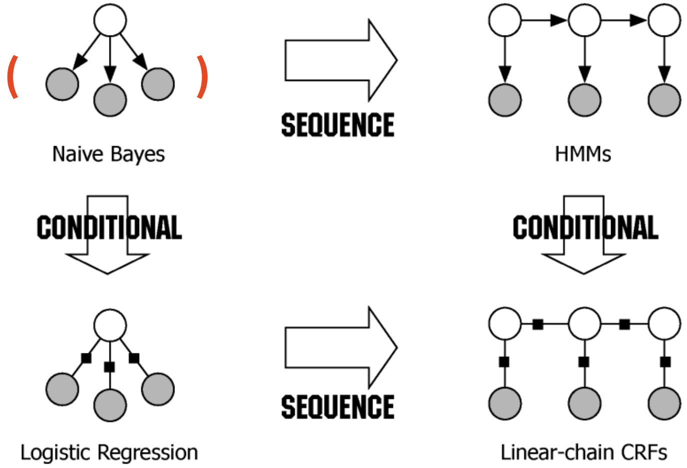
U: I'm looking for flights from JFK.
S: Where would you like to go?
U: Atlanta.

inform(**??**=Atlanta)
inform(**from**=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



https://arxiv.org/abs/1011.4088

Thanks

Contact me:

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

Labs tomorrow
9:00 SU1

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
- Raymond Mooney's slides (University of Texas Austin): https://www.cs.utexas.edu/~mooney/ir-course/
- Filip Jurčíček'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
- 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 Massachusets Amherst): https://people.cs.umass.edu/~mccallum/courses/inlp2007/