

Artificial Intelligence

EDA132

Lecture 14.2: Phrase-Structure Grammars

Pierre Nugues

Lund University
Pierre.Nugues@cs.lth.se
http://cs.lth.se/pierre_nugues/

March 3, 2017



Applications of Language Processing

Grammar is the focus of natural language processing in the textbook (Russell and Norvig 2010, Chapter 23).

Two main (modern) traditions: constituent grammars (Chomsky, main advocate) and dependency grammars (Tesnière).

Constituent grammars are still dominant for English, although declining.

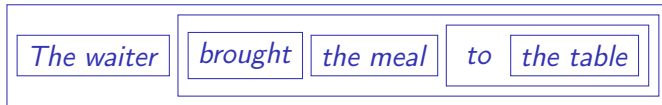


Constituents

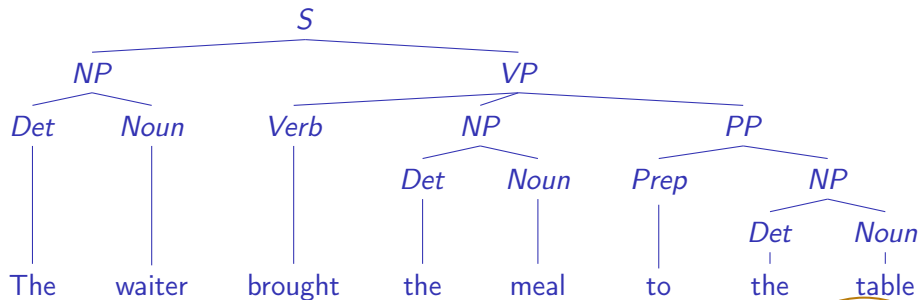
The waiter brought the meal

The waiter brought the meal to the table

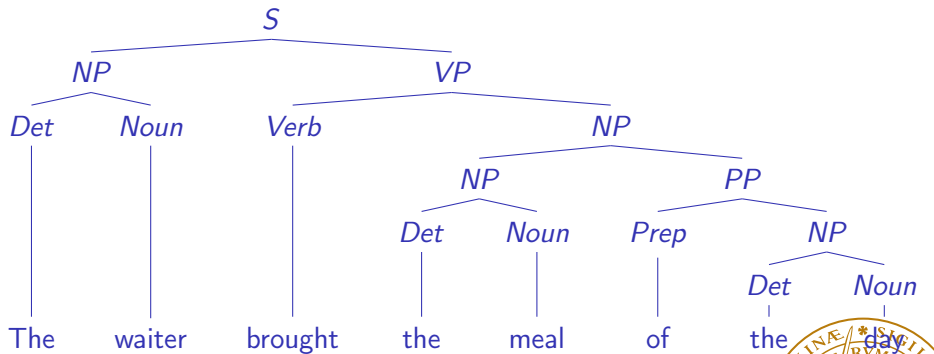
The waiter brought the meal of the day



Syntactic Trees



Syntactic Trees



Lexicon (DCG)

noun --> [stench] ; [breeze] ; [glitter] ; [nothing] ;
 [wumpus] ; [pit] ; [pits] ; [gold] ; [east].
 verb --> [is] ; [see] ; [smell] ; [shoot] ; [feel] ;
 [stinks] ; [go] ; [grab] ; [carry] ; [kill] ; [turn].
 adjective --> [right] ; [left] ; [east] ; [south] ; [dead] ;
 [back] ; [smelly].
 adverb --> [here] ; [there] ; [nearby] ; [ahead] ; [right] ;
 [left] ; [east] ; [south] ; [back].
 pronoun --> [me] ; [you] ; ['I'] ; [it] ; [she] ; [he].
 pnoun --> ['John'] ; ['Mary'] ; ['Boston'] ; ['UCB'] ;
 ['PAJC'].
 article --> [the] ; [a] ; [an].
 preposition --> [to] ; [in] ; [on] ; [near].
 conjunction --> [and] ; [or] ; [but].
 digit --> [0] ; [1] ; [2] ; [3] ; [4] ; [5] ; [6] ; [7] ;
 [8] ; [9].



Grammar Rules (DCG)

```

s --> np, vp. % I + feel a breeze
s --> s, conjunction, s.
np --> pronoun. %I
np --> pnoun.
np --> noun. %pits
np --> article, noun. %the + wumpus
np --> digit, digit. %3 4
np --> np, pp. %the wumpus + to the east
np --> np, rel_clause. %the wumpus + that is smelly
vp --> verb. %stinks
vp --> vp, np. %feel + a breeze
vp --> vp, adjective. %is + smelly
vp --> vp, pp. %turn + to the east
vp --> vp, adverb. %go + ahead
pp --> preposition, np. %to + the east
rel_clause --> [that], vp. %that + is smelly

```



Parsing and Generation

Parsing tells if a sentence is correct according to the grammar

```
?-s([the, wumpus, is, dead], []).
```

yes.

```
?- s([the, wumpus, that, stinks, is, in, 2, 2], []).
```

yes.

The parser can generate all the solutions

```
?- s(L, []).
```

```
L = [me, is] ;
```

```
L = [me, see] ;
```

```
L = [me, smell] ;
```

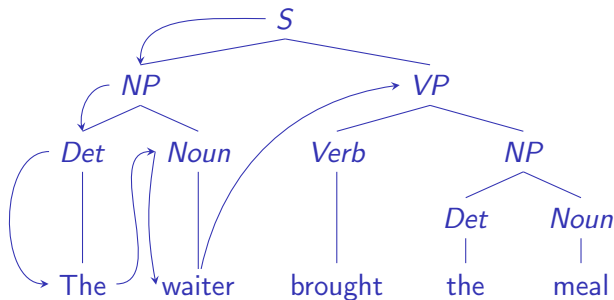
```
L = [me, shoot] ;
```

```
L = [me, feel] ;
```

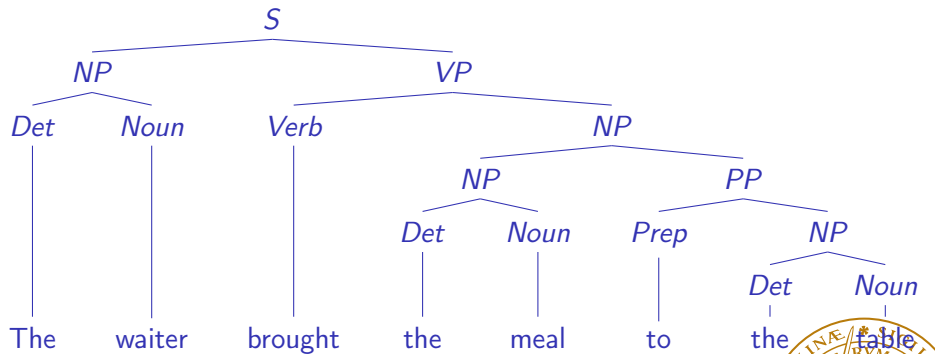
```
L = [me, stinks] ;
```



The Prolog Search



Ambiguity



Left-Recursive Rules

$np \rightarrow np, pp.$

The sentence:

The wumpus in the pit is dead

traps the parser in an infinite recursion.

We can use auxiliary symbols to remove left recursion:

$npx \rightarrow det, noun.$

$np \rightarrow npx.$

$np \rightarrow npx, pp.$



Variables

Overgeneration:

```
?- s(X, []).
X = [me, is] ;
X = [me, see] ;
X = [me, smell] ;
```

Solution: Add variables to differentiate between subject and object pronouns.

```
s --> np(s), vp.
np(Case) --> pronoun(Case).
pronoun(s) --> [you] ; ['I'] ; [it]; [she]; [he].
pronoun(o) --> [me] ; [you] ; [it].
```



Probabilistic Context-Free Grammars

$$P(T, S) = \prod_{rule(i) \text{ producing } T} P(rule(i)).$$

where

$$P(lhs \rightarrow rhs_i | lhs) = \frac{Count(lhs \rightarrow rhs_i)}{\sum_j Count(lhs \rightarrow rhs_j)}.$$



An Example of PCFG

Rules	P	Rules	P
s --> np vp	0.8	det --> the	1.0
s --> vp	0.2	noun --> waiter	0.4
np --> det noun	0.3	noun --> meal	0.3
np --> det adj noun	0.2	noun --> day	0.3
np --> pronoun	0.3	verb --> bring	0.4
np --> np pp	0.2	verb --> slept	0.2
vp --> v np	0.6	verb --> brought	0.4
vp --> v np pp	0.1	pronoun --> he	1.0
vp --> v pp	0.2	prep --> of	0.6
vp --> v	0.1	prep --> to	0.4
pp --> prep np	1.0	adj --> big	1.0



Parse Trees of *Bring the meal of the day*

Parse trees

T1: `vp(verb(bring),
 np(np(det(the), noun(meal)),
 pp(preposition(of), np(det(the), noun(day))))))`

T2: `vp(verb(bring),
 np(np(det(the), noun(meal))),
 pp(preposition(of), np(det(the), noun(day))))`



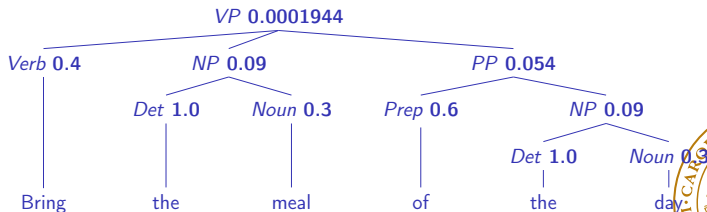
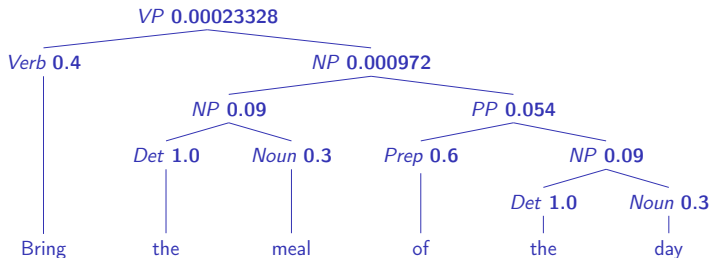
Computing the Probabilities

$$\begin{aligned}
 &P(T_1, \text{Bring the meal of the day}) = \\
 &P(vp \rightarrow v, np) \times P(v \rightarrow \text{Bring}) \times P(np \rightarrow np, pp) \times \\
 &P(np \rightarrow det, noun) \times P(det \rightarrow the) \times P(noun \rightarrow meal) \times \\
 &P(pp \rightarrow prep, np) \times P(preposition \rightarrow of) \times P(np \rightarrow det, noun) \times \\
 &P(det \rightarrow the) \times P(noun \rightarrow day) = \\
 &0.6 \times 0.4 \times 0.2 \times 0.3 \times 1.0 \times 0.3 \times 1.0 \times 0.6 \times 0.3 \times 1.0 \times 0.3 = 0.00023328,
 \end{aligned}$$

$$\begin{aligned}
 &P(T_2, \text{Bring the meal of the day}) = \\
 &P(vp \rightarrow v, np, pp) \times P(v \rightarrow \text{Bring}) \times P(np \rightarrow det, noun) \times \\
 &P(det \rightarrow the) \times P(noun \rightarrow meal) \times P(pp \rightarrow prep, np) \times P(preposition \rightarrow of) \times \\
 &P(np \rightarrow det, noun) \times P(det \rightarrow the) \times P(noun \rightarrow day) = \\
 &0.1 \times 0.4 \times 0.3 \times 1.0 \times 0.3 \times 1.0 \times 0.6 \times 0.3 \times 1.0 \times 0.3 = 0.0001944
 \end{aligned}$$



Computing the Probabilities



Subcategorization Frames

Valence is a model of verb construction. It can be extended to more specific patterns as in the *Oxford Advanced Learner's Dictionary* (OALD).

Verb	Complement structure	Example
<i>slept</i>	None (Intransitive)	<i>I slept</i>
<i>bring</i>	NP	<i>The waiter brought the meal</i>
<i>bring</i>	NP + to + NP	<i>The waiter brought the meal to the patron</i>
<i>depend</i>	on + NP	<i>It depends on the waiter</i>
<i>wait</i>	for + NP + to + VP	<i>I am waiting for the waiter to bring the meal</i>
<i>keep</i>	VP(ing)	<i>He kept working</i>
<i>know</i>	that + S	<i>The waiter knows that the patron loves fish</i>



Semantic Parsing

Converts sentences to first-order logic or predicate-argument structures

Example:

Mr. Schmidt called Bill

to

`called('Mr. Schmidt', 'Bill').`

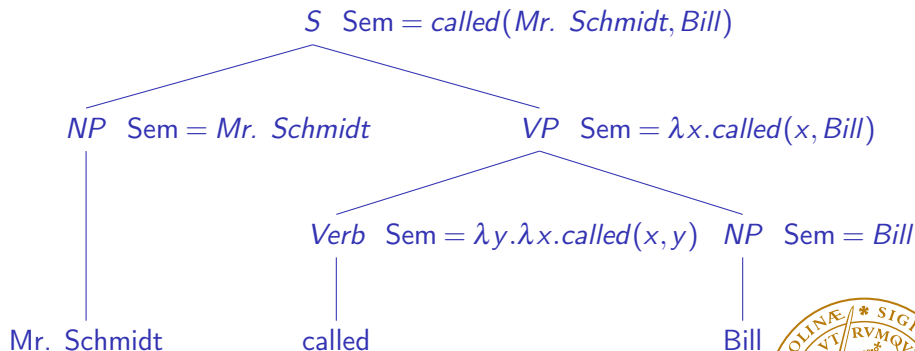
Assumption: We can compose sentence fragments (phrases) into logical forms while parsing

This corresponds to the compositionality principle



Semantic Composition

Semantic composition can be viewed as a parse tree annotation



Getting the Semantic Structure

Bill rushed rushed('Bill') .

The verb *rushed* is represented as a lambda expression: $\lambda x.\text{rushed}(x)$

Beta reduction: $\lambda x.\text{rushed}(x)(\text{Bill}) = \text{rushed}(\text{Bill})$

Lambda expressions are represented in Prolog as $X^{\text{rushed}}(X)$.

<i>The patron ordered a meal</i>	$\text{ordered}(\text{patron}, \text{meal})$
<i>ordered a meal</i>	$X^{\text{ordered}}(X, \text{meal})$
<i>ordered</i>	$Y^X^{\text{ordered}}(X, Y)$



Getting the Semantic Structure

```

s(Semantics) --> np(Subject), vp(Subject^Semantics).
np(X) --> det, noun(X).
vp(Subject^Predicate) --> verb(Subject^Predicate).
vp(Subject^Predicate) -->
verb(Object^Subject^Predicate), np(Object).
noun(waiter) --> [waiter].
noun(patron) --> [patron].
noun(meal) --> [meal]. det --> [a].
det --> [the].

```

```

verb(X^rushed(X)) --> [rushed].
verb(Y^X^ordered(X, Y)) --> [ordered].
verb(Y^X^brought(X, Y)) --> [brought].

```

```

?- s(Semantics, [the, patron, ordered, a, meal], []).
Semantics = ordered(patron, meal)

```



Statistical Semantic Parsing: Annotation

- ① [_{<Avenger>} His brothers] **avenged** [_{<Injured_party>} him].
- ② With this, [_{<Avenger>} El Cid] at once **avenged** [_{<Injury>} the death of his son].
- ③ [_{<Avenger>} Hook] tries to **avenge** [_{<Injured_party>} himself] [_{<Offender>} on Peter Pan] [_{<Punishment>} by becoming a second and better father].

FrameNet uses three annotation levels: Frame elements, Phrase types (categories), and grammatical functions.

GFs are specific to the target's part-of-speech (i.e. verbs, adjectives, prepositions, and nouns).

For the verbs, three GFs: Subject (Ext), Object (Obj), Complement (Dep) and Modifier (Mod), i.e. modifying adverbs ended by *-ly* or indicating manner



Automatic Frame-semantic Analysis (Johansson, 2008)

Given a sentence:

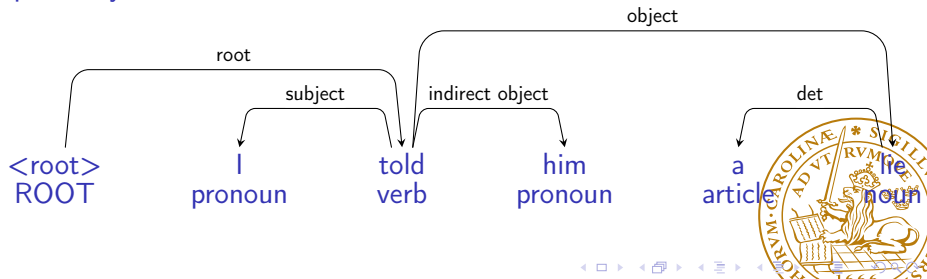
I told him a lie

and a target word – **tell** –, find the semantic arguments.

Semantic analysis often uses Propbank instead of Framenet because of Propbank's larger annotated corpus

In Propbank, the possible arguments of **tell.01** are *speaker* (Arg0), *utterance* (Arg1), and *hearer* (Arg2)

Input: a syntax tree



Classification of Semantic Arguments (Johansson, 2008)

Two steps:

- Find the arguments,
- Determine the name of each argument

The identification of semantic arguments can be modeled as a statistical classification problem.

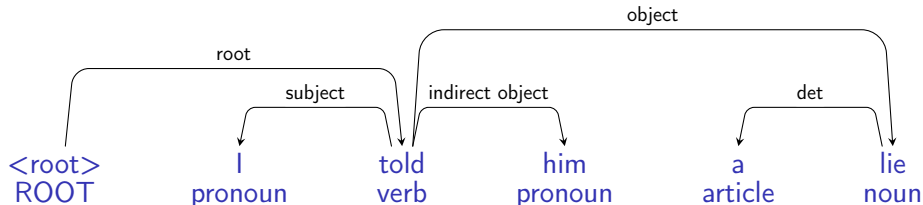
What features are useful for this task? Examples:

- Grammatical function: subject, object, ...
- Voice: *I told a lie* / *I was told a lie*
- Semantic classes: *I told him* / *the note told him*
- Semantic class usually not available: use word instead



Feature Extraction (Johansson, 2008)

Given a dependency tree:



We select the three dependents of *told* and we extract features to determine if it is a semantic argument and its name.

Word	Grammatical function	Voice	Argument
<i>I</i>	Subject	Active	<i>speaker</i> (Arg0)
<i>him</i>	Indirect object	Active	<i>hearer</i> (Arg2)
<i>lie</i>	Direct object	Active	<i>utterance</i> (Arg1)



Events

Research on the representation of time, events, and temporal relations dates back the beginning of logic.

It resulted in an impressive number of formulations and models.

A possible approach is to **reify** events: turn them into objects, quantify them existentially, and connect them using predicates







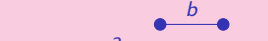
John saw Mary in London on Tuesday

$$\exists \varepsilon [\text{saw}(\varepsilon, \text{John}, \text{Mary}) \wedge \text{place}(\varepsilon, \text{London}) \wedge \text{time}(\varepsilon, \text{Tuesday})],$$

where ε represents the event.



Temporal Representation of Events (Allen 1983)

#	Relations	#	Inverse relations	Graphical representations
1.	before(a, b)	2.	after(b, a)	
3.	meets(a, b)	4.	met_by(b, a)	
5.	overlaps(a, b)	6.	overlapped_by(b, a)	
7.	starts(a, b)	8.	started_by(b, a)	
9.	during(b, a)	10.	contains(a, b)	
11.	finishes(b, a)	12.	finished_by(a, b)	
13.	equals(a, b)			

Determiners

A caterpillar is eating

$\exists x, \text{caterpillar}(x) \wedge \text{eating}(x)$, or
 $\text{exists}(X, \text{caterpillar}(X), \text{eating}(X))$

Every caterpillar is eating

$\forall x, \text{caterpillar}(x) \Rightarrow \text{eating}(x)$, or
 $\text{all}(X, \text{caterpillar}(X), \text{eating}(X))$

A caterpillar is eating a hedgehog

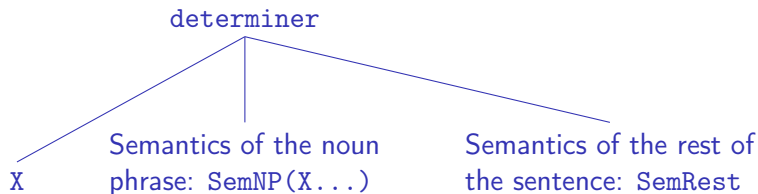
$\exists x, \text{caterpillar}(x) \wedge (\exists y, \text{hedgehog}(y) \wedge \text{eating}(x, y))$, or
 $\text{exists}(X, \text{caterpillar}(X), \text{exists}(Y, \text{hedgehog}(Y), \text{eating}(X, Y)))$

Every caterpillar is eating a hedgehog

$\forall x, \text{caterpillar}(x) \Rightarrow (\exists y, \text{hedgehog}(y) \wedge \text{eating}(x, y))$, or
 $\text{all}(X, \text{caterpillar}(X), \text{exists}(Y, \text{hedgehog}(Y), \text{eating}(X, Y)))$



General Representation of Determiners



Representation:

$(X \wedge NP) \wedge (X \wedge Rest) \wedge a(X, NP, Rest)$



Compositionality

Some rules to generate the logical form:

`noun(X^dog(X)) --> [dog].`

`noun(X^wumpus(X)) --> [wumpus].`

`determiner((X^NP)^exists(X, NP)) --> [a].`

`np(SemDet) --> determiner((X^NP)^SemDet), noun(X^NP).`

`?- np(Semantics, [a, wumpus], []).`

`Semantics = exists(_4,wumpus(_4))`



Machine Translation

Natural language processing was born with machine translation
Massive advance when the US government decided to fund large-scale translation programs to have a quick access to documents written in Russian
IBM teams pioneered statistical models for machine translation in the early 1990s
Their work that used the English (e) and French (f) parallel versions of the Canadian Hansards is still the standard reference in the field.



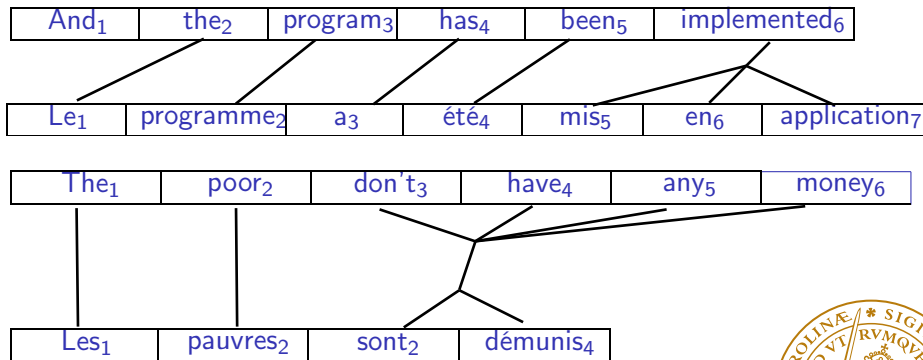
Parallel Corpora (Swiss Federal Law)

German	French	Italian
Art. 35 Milchtransport	Art. 35 Transport du lait	Art. 35 Trasporto del latte
<p>1 Die Milch ist schonend und hygienisch in den Verarbeitungsbetrieb zu transportieren. Das Transportfahrzeug ist stets sauber zu halten. Zusammen mit der Milch dürfen keine Tiere und milchfremde Gegenstände transportiert werden, welche die Qualität der Milch beeinträchtigen können.</p>	<p>1 Le lait doit être transporté jusqu'à l'entreprise de transformation avec ménagement et conformément aux normes d'hygiène. Le véhicule de transport doit être toujours propre. Il ne doit transporter avec le lait aucun animal ou objet susceptible d'en altérer la qualité.</p>	<p>1 Il latte va trasportato verso l'azienda di trasformazione in modo accurato e igienico. Il veicolo adibito al trasporto va mantenuto pulito. Con il latte non possono essere trasportati animali e oggetti estranei, che potrebbero pregiudicare la qualità.</p>



Alignment (Brown et al. 1993)

Canadian Hansard



Machine Translation Algorithms

A statistical model:

$$P(f, d|e) = \prod_i P(f_i|e_i)P(d_i),$$

where d measures the distortion, how much reassembling is needed from English to French.

Distortion has the form of a right-to-left or left-to-right shift.

Steps to build a machine translation system:

- Build parallel corpora
- Segment and align sentences
- Align phrases
- Extract distortions
- Improve estimates



Speech Recognition

Conditions to take into account:

- Number of speakers
- Fluency of speech.
- Size of vocabulary
- Syntax
- Environment



Structure of Speech Recognition

Words:

$$W = w_1, w_2, \dots, w_n.$$

Acoustic symbols:

$$A = a_1, a_2, \dots, a_m,$$

$$\hat{W} = \arg \max_W P(W|A).$$

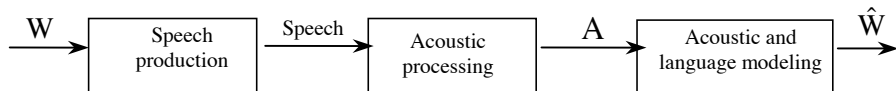
Using Bayes' formula,

$$P(W|A) = \frac{P(A|W)P(W)}{P(A)}.$$



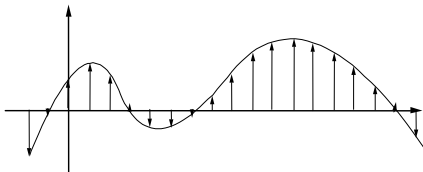
Two-Step Recognition

$$\hat{W} = \arg \max_W P(A|W)P(W).$$

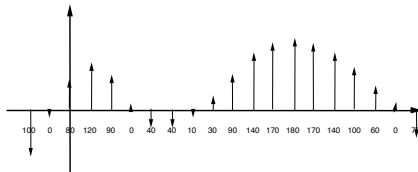


Signals

Sampling



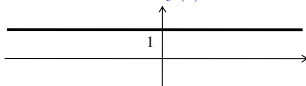
Digitization



Fourier Transforms

Time domain

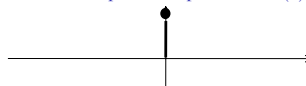
Unit constant function: $f(x) = 1$



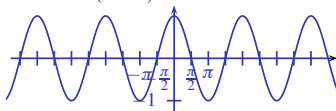
Frequency domain

(Fourier Transforms)

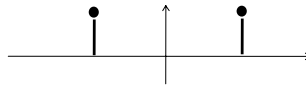
Delta function, perfect impulse at 0: $\delta(x)$



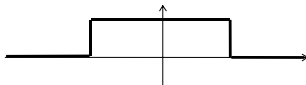
Cosine: $\cos(2\pi\omega x)$



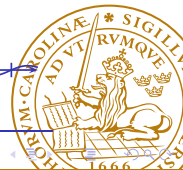
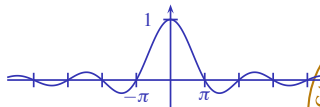
Shifted deltas: $\frac{\delta(x+\omega) + \delta(x-\omega)}{2}$



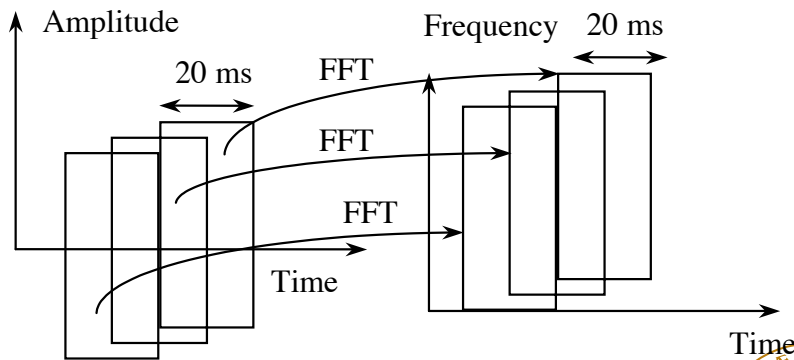
Square pulse : $w_a(x) = \begin{cases} 1 & -\frac{1}{2} \leq x \leq \frac{1}{2} \\ 0 & \text{elsewhere} \end{cases}$



$\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$

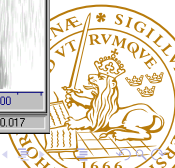
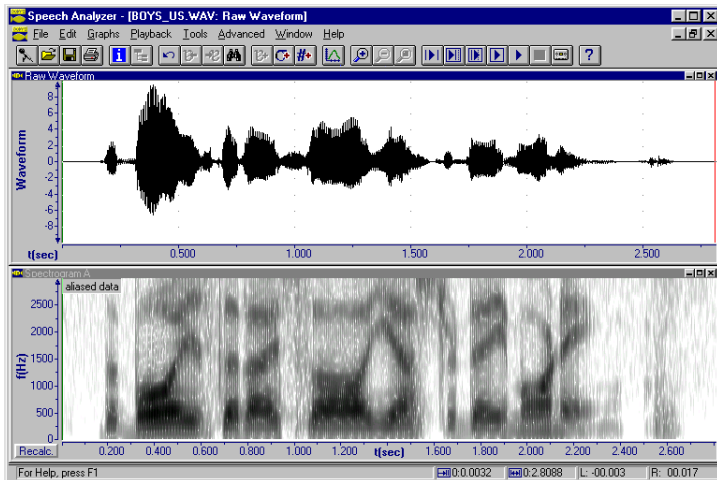


Speech Spectrograms



Speech Signals

The boys I saw yesterday morning



Speech Parameters

Recognition devices derive a set of acoustic parameters from speech frames. Parameters should be related to “natural” features of speech: voiced or unvoiced segments.

A simple parameter giving a rough estimate of it: the energy: the darker the frame, the higher the energy.

$$E(F_k) = \sum_{n=m}^{m+N-1} s^2(n).$$

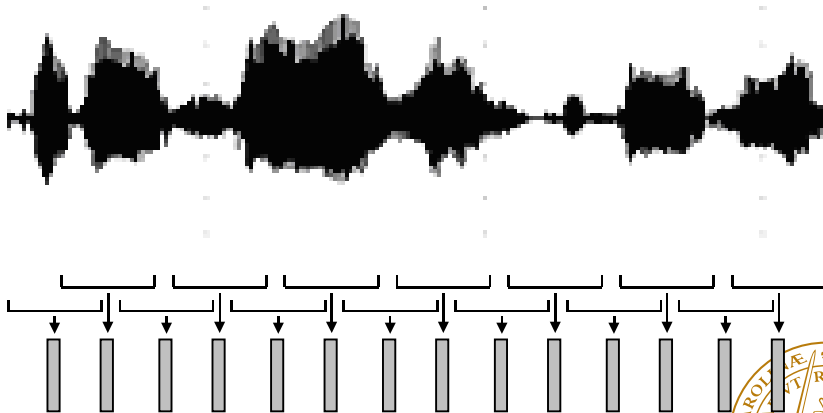
Linear prediction coefficients:

$$\hat{s}(n) = a(1)s(n-1) + a(2)s(n-2) + a(3)s(n-3) + \dots + a(m)s(n-m).$$

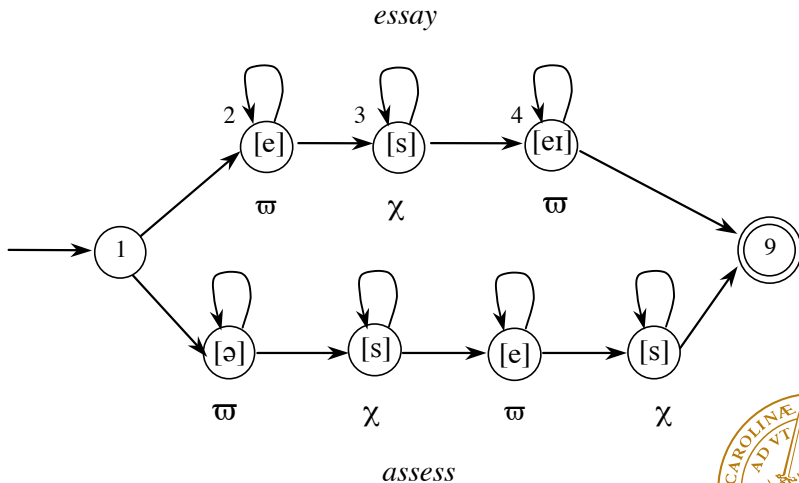


Extraction of Speech Parameters

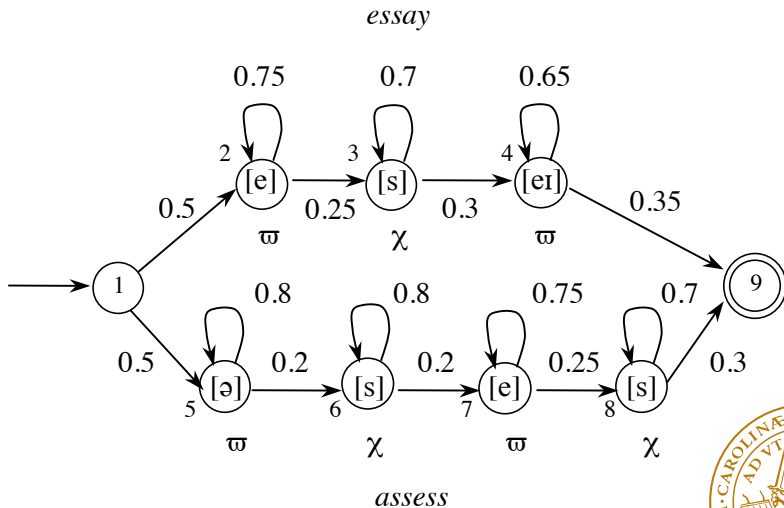
Features are extracted every 10 ms over a 20 s frame



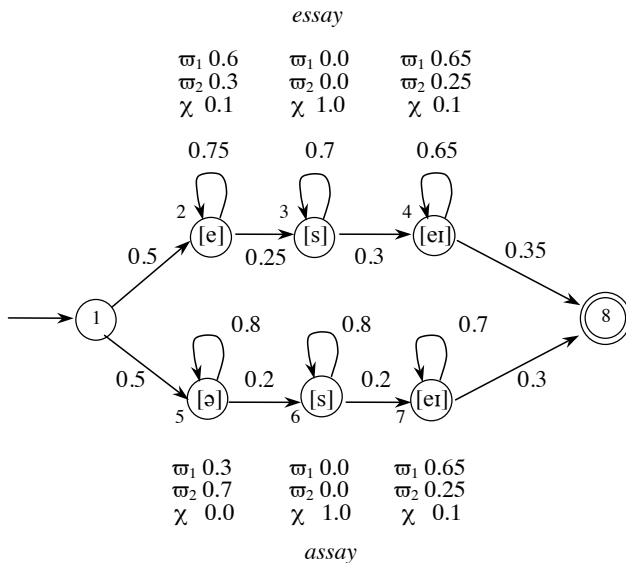
Automata



Markov Chains



Hidden-Markov Models



Solving Problems with Hidden-Markov Models

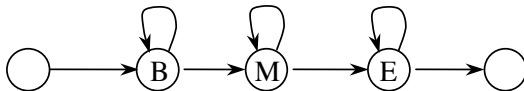
Given a hidden-Markov model, the main problems to solve are to:

- Estimate the probability of an observed sequence. It corresponds to the sum of all the paths producing the observation. It is solved using the forward algorithm.
- Determine the most likely path of an observed sequence. It is a decoding problem. It is solved using the Viterbi algorithm.
- Determine (learn) the parameters given a set of observations. It is used to build models to recognize speech. It is solved using the forward-backward algorithm.



HMM and Phones

Modeling phones:
Simple model



A more complex model due to Lee

