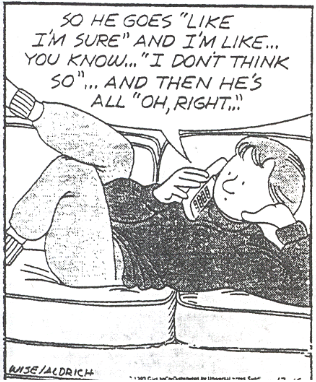
CS325 Artificial Intelligence
Natural Language Processing II (Ch. 23)

Dr. Cengiz Günay, Emory Univ.



So Probabilities Enough for Understanding Language?

He came from out of nowhere.

So Probabilities Enough for Understanding Language?

He came from out of nowhere.
From out of nowhere, he came.

So Probabilities Enough for Understanding Language?

He came from out of nowhere.
From out of nowhere, he came.

- Same meaning but different ordering: non-Markovian.
- How do we understand that both sentences have similar meaning?

So Probabilities Enough for Understanding Language?

He came from out of nowhere.
From out of nowhere, he came.

- Same meaning but different ordering: non-Markovian.
- How do we understand that both sentences have similar meaning?
- Look at sentence structure: "from out of nowhere" and "he came"

So Probabilities Enough for Understanding Language?

He came from out of nowhere.
From out of nowhere, he came.

- Same meaning but different ordering: non-Markovian.
- How do we understand that both sentences have similar meaning?
- Look at sentence structure: "from out of nowhere" and "he came"

Today:

1. Using sentence structure in NLP
2. Machine translation
3. Speech recognition (no time, see textbook)

Entry/Exit Surveys

Exit survey: Natural Language Processing I

- What is a good method for identifying foreign languages?
- How do we improve bag of words to learn word sequences?

Entry survey: Natural Language Processing II (0.25 pts)

- Give some examples of why learning sentence structure may be useful.
- What was the most useful machine translation tool you ever used?

Uses of Sentence Structure in NLP

Can be useful for:

- Disambiguation of phrases

Uses of Sentence Structure in NLP

Can be useful for:

- Disambiguation of phrases
- Understanding meaning

Uses of Sentence Structure in NLP

Can be useful for:

- Disambiguation of phrases
- Understanding meaning
- Translation

Disambiguation

Strike a match.

Disambiguation

Strike a match.

Disambiguation

Strike a match.

How Can We Use the Sentence Structure?

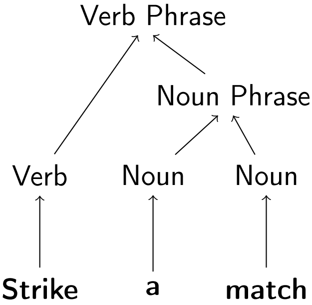
Hint: 

How Can We Use the Sentence Structure?

Hint: 

Strike **a** **match**

How Can We Use the Sentence Structure?



Hint:

How Can We Use the Sentence Structure?

Verb Phrase

Noun Phrase

Verb | Noun | Noun

Strike | **a** | **match**

Noun | Noun | Noun

Noun Phrase

Hint: 

Where Do the Trees Come From?

Where Do the Trees Come From?

From the forest?

Where Do the Trees Come From?

From the forest?

Seriously, from:

The grammar:

$S \rightarrow VP|NP$

$VP \rightarrow V\ NP|V$

$NP \rightarrow N|N\ N|N\ N\ N$

$N \rightarrow \text{strike}|\text{match}$

$V \rightarrow \text{strike}|\text{match}$

Where Do the Trees Come From?

From the forest?

Seriously, from:

The grammar:

S → VP|NP
VP → V NP|V
NP → N|N N|N N N
N → strike|match
V → strike|match

Results in **multiple possible parses** of the same sentence.

Multiple Possible Parsleys

Parses, parsings, or parsleys
(whatever)

Multiple Possible Parsleys

Parses, parsings, or parsleys (whatever)

"strike a match" can be parsed as:

1. verb noun noun
2. noun noun noun
3. noun noun verb

Multiple Possible Parsleys

Parses, parsings, or parsleys (whatever)

"strike a match" can be parsed as:

1. verb noun noun
2. noun noun noun
3. noun noun verb

Problems?

Multiple Possible Parsleys

Parses, parsings, or parsleys (whatever)

"strike a match" can be parsed as:

1. verb noun noun
2. noun noun noun
3. noun noun verb

Problems?

1. Omitting a good parsley (false negative): #1 above

Multiple Possible Parsleys

Parses, parsings, or parsleys (whatever)

"strike a match" can be parsed as:

1. verb noun noun
2. noun noun noun
3. noun noun verb

Problems?

1. Omitting a good parsley (false negative): #1 above
2. Including a bad parsley (false positive): #2 or #3 above

Multiple Possible Parsleys

Parses, parsings, or parsleys (whatever)

"strike a match" can be parsed as:

1. verb noun noun
2. noun noun noun
3. noun noun verb

Problems?

1. Omitting a good parsley (false negative): #1 above
2. Including a bad parsley (false positive): #2 or #3 above

Solutions?

1. Use probabilities
2. Use word associations
3. Unambiguous grammar

Multiple Possible Parsleys

Parses, parsings, or parsleys (whatever)

"strike a match" can be parsed as:

1. verb noun noun
2. noun noun noun
3. noun noun verb

Problems?

1. Omitting a good parsley (false negative): #1 above
2. Including a bad parsley (false positive): #2 or #3 above

Solutions?

1. **Use probabilities**
2. Use word associations
3. Unambiguous grammar

Multiple Possible Parsleys

Parses, parsings, or parsleys (whatever)

"strike a match" can be parsed as:

1. verb noun noun
2. noun noun noun
3. noun noun verb

Problems?

1. Omitting a good parsley (false negative): #1 above
2. Including a bad parsley (false positive): #2 or #3 above

Solutions?

1. Use probabilities
2. Use word associations
3. Unambiguous grammar

Use Probabilities and Grammar Together

context-free grammar: Words are expanded without context (e.g., $\mathrm{S} \rightarrow \mathrm{VP}|\mathrm{NP}$). Used with programming languages.

context-free grammar: Words are expanded without context (e.g., $\mathrm{S} \rightarrow \mathrm{VP}|\mathrm{NP}$). Used with programming languages.

"strike a match"

The probabilistic grammar:

$\mathrm{S} \rightarrow \mathrm{VP}(0.7)|\mathrm{NP}(0.3)$

Use Probabilities and Grammar Together

context-free grammar: Words are expanded without context (e.g., $\mathrm{S} \rightarrow \mathrm{VP}|\mathrm{NP}$). Used with programming languages.

"strike a match"

The probabilistic grammar:

$\mathrm{S} \rightarrow \mathrm{VP}(0.7)|\mathrm{NP}(0.3)$
$\mathrm{VP} \rightarrow \mathrm{V}\,\mathrm{NP}(0.6)|\mathrm{V}(0.4)$

Use Probabilities and Grammar Together

context-free grammar: Words are expanded without context (e.g., $\mathrm{S} \rightarrow \mathrm{VP}|\mathrm{NP}$). Used with programming languages.

"strike a match"

The probabilistic grammar:

$\mathrm{S} \rightarrow \mathrm{VP}(0.7)|\mathrm{NP}(0.3)$
$\mathrm{VP} \rightarrow \mathrm{V}\,\mathrm{NP}(0.6)|\mathrm{V}(0.4)$
$\mathrm{NP} \rightarrow \mathrm{N}(0.6)|\mathrm{N}\,\mathrm{N}(0.3)|\mathrm{N}\,\mathrm{N}\,\mathrm{N}(0.1)$

Use Probabilities and Grammar Together

context-free grammar: Words are expanded without context (e.g., $\mathrm{S} \rightarrow \mathrm{VP}|\mathrm{NP}$). Used with programming languages.

"strike a match"

The probabilistic grammar:

$\mathrm{S} \rightarrow \mathrm{VP}(0.7)|\mathrm{NP}(0.3)$
$\mathrm{VP} \rightarrow \mathrm{V}\,\mathrm{NP}(0.6)|\mathrm{V}(0.4)$
$\mathrm{NP} \rightarrow \mathrm{N}(0.6)|\mathrm{N}\,\mathrm{N}(0.3)|\mathrm{N}\,\mathrm{N}\,\mathrm{N}(0.1)$
$\mathrm{N} \rightarrow \mathrm{strike}(0.4)|\mathrm{match}(0.7)$

Use Probabilities and Grammar Together

context-free grammar: Words are expanded without context (e.g., $\mathrm{S} \rightarrow \mathrm{VP}|\mathrm{NP}$). Used with programming languages.

"strike a match"

The probabilistic grammar:

$\mathrm{S} \rightarrow \mathrm{VP}(0.7)|\mathrm{NP}(0.3)$
$\mathrm{VP} \rightarrow \mathrm{V}\,\mathrm{NP}(0.6)|\mathrm{V}(0.4)$
$\mathrm{NP} \rightarrow \mathrm{N}(0.6)|\mathrm{N}\,\mathrm{N}(0.3)|\mathrm{N}\,\mathrm{N}\,\mathrm{N}(0.1)$
$\mathrm{N} \rightarrow \mathrm{strike}(0.4)|\mathrm{match}(0.7)$
$\mathrm{V} \rightarrow \mathrm{strike}(0.6)|\mathrm{match}(0.3)$

Use Probabilities and Grammar Together

context-free grammar: Words are expanded without context (e.g., $\mathrm{S} \rightarrow \mathrm{VP}|\mathrm{NP}$). Used with programming languages.

"strike a match"

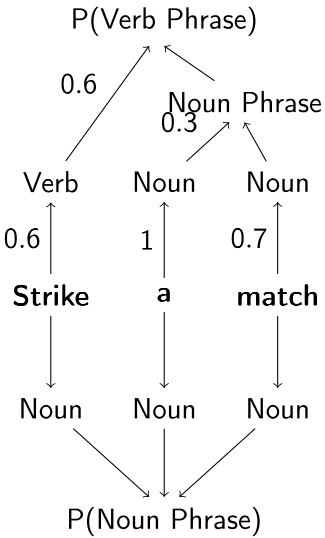
The probabilistic grammar:

$\mathrm{S} \rightarrow \mathrm{VP}(0.7)|\mathrm{NP}(0.3)$
$\mathrm{VP} \rightarrow \mathrm{V}\,\mathrm{NP}(0.6)|\mathrm{V}(0.4)$
$\mathrm{NP} \rightarrow \mathrm{N}(0.6)|\mathrm{N}\,\mathrm{N}(0.3)|\mathrm{N}\,\mathrm{N}\,\mathrm{N}(0.1)$
$\mathrm{N} \rightarrow \mathrm{strike}(0.4)|\mathrm{match}(0.7)$
$\mathrm{V} \rightarrow \mathrm{strike}(0.6)|\mathrm{match}(0.3)$

It's called a **probabilistic context-free grammar (PCFG)**

The probabilistic grammar:

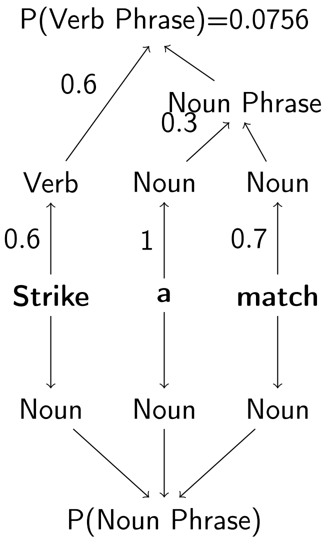
$S \rightarrow VP(0.7)|NP(0.3)$
$VP \rightarrow V\,NP(0.6)|V(0.4)$
$NP \rightarrow N(0.6)|N\,N(0.3)|N\,N\,N(0.1)$
$N \rightarrow strike(0.4)|match(0.7)$
$V \rightarrow strike(0.6)|match(0.3)$



PCFG Example

The probabilistic grammar:

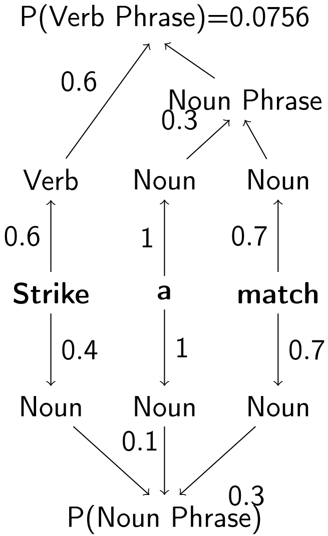
$S \rightarrow VP(0.7) | NP(0.3)$
$VP \rightarrow V\, NP(0.6) | V(0.4)$
$NP \rightarrow N(0.6) | N\, N(0.3) | N\, N\, N(0.1)$
$N \rightarrow \text{strike}(0.4) | \text{match}(0.7)$
$V \rightarrow \text{strike}(0.6) | \text{match}(0.3)$



PCFG Example

The probabilistic grammar:

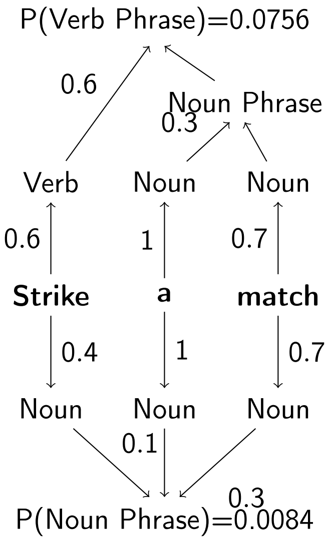
S $\rightarrow$ VP(0.7)|NP(0.3)
VP $\rightarrow$ V NP(0.6)|V(0.4)
NP $\rightarrow$ N(0.6)|N N(0.3)|N N N(0.1)
N $\rightarrow$ strike(0.4)|match(0.7)
V $\rightarrow$ strike(0.6)|match(0.3)



PCFG Example

The probabilistic grammar:

$S \rightarrow VP(0.7)|NP(0.3)$
$VP \rightarrow V\ NP(0.6)|V(0.4)$
$NP \rightarrow N(0.6)|N\ N(0.3)|N\ N\ N(0.1)$
$N \rightarrow \text{strike}(0.4)|\text{match}(0.7)$
$V \rightarrow \text{strike}(0.6)|\text{match}(0.3)$



How to Get Grammar Probabilities?

I made them up :)
Can we count them?

How to Get Grammar Probabilities?

I made them up :)

Can we count them? **No**, they are ambiguous out in the wild.

How to Get Grammar Probabilities?

I made them up :)

Can we count them? **No**, they are ambiguous out in the wild.

First need a model of grammar, but problems:

How to Get Grammar Probabilities?

I made them up :)
Can we count them? **No**, they are ambiguous out in the wild.
First need a model of grammar, but problems:

- Grammars are biologically evolved

How to Get Grammar Probabilities?

I made them up :)
Can we count them? **No**, they are ambiguous out in the wild.
First need a model of grammar, but problems:

- Grammars are biologically evolved
- They are complex and rough

How to Get Grammar Probabilities?

I made them up :)
Can we count them? **No**, they are ambiguous out in the wild.
First need a model of grammar, but problems:

- Grammars are biologically evolved
- They are complex and rough
- Neat rules all have exceptions

How to Get Grammar Probabilities?

I made them up :)

Can we count them? **No**, they are ambiguous out in the wild.

First need a model of grammar, but problems:

- Grammars are biologically evolved
- They are complex and rough
- Neat rules all have exceptions

Solution?

How to Get Grammar Probabilities?

I made them up :)
Can we count them? **No**, they are ambiguous out in the wild.
First need a model of grammar, but problems:

- Grammars are biologically evolved
- They are complex and rough
- Neat rules all have exceptions

Solution?

- Machine learning

How to Get Grammar Probabilities?

I made them up :)

Can we count them? **No**, they are ambiguous out in the wild.

First need a model of grammar, but problems:

- Grammars are biologically evolved
- They are complex and rough
- Neat rules all have exceptions

Solution?

- Machine learning

But where's the data?

How to Get Grammar Probabilities?

I made them up :)
Can we count them? **No**, they are ambiguous out in the wild.
First need a model of grammar, but problems:

- Grammars are biologically evolved
- They are complex and rough
- Neat rules all have exceptions

Solution?

- Machine learning

But where's the data?

- Need to pay people to build databases (e.g., Penn Tree Bank)

How to Get Grammar Probabilities?

I made them up :)

Can we count them? **No**, they are ambiguous out in the wild.

First need a model of grammar, but problems:

- Grammars are biologically evolved
- They are complex and rough
- Neat rules all have exceptions

Solution?

- Machine learning

But where's the data?

- Need to pay people to build databases (e.g., Penn Tree Bank)

Can you think of a better solution?

How to Get Grammar Probabilities?

I made them up :)
Can we count them? **No**, they are ambiguous out in the wild.
First need a model of grammar, but problems:

- Grammars are biologically evolved
- They are complex and rough
- Neat rules all have exceptions

Solution?

- Machine learning

But where's the data?

- Need to pay people to build databases (e.g., Penn Tree Bank)

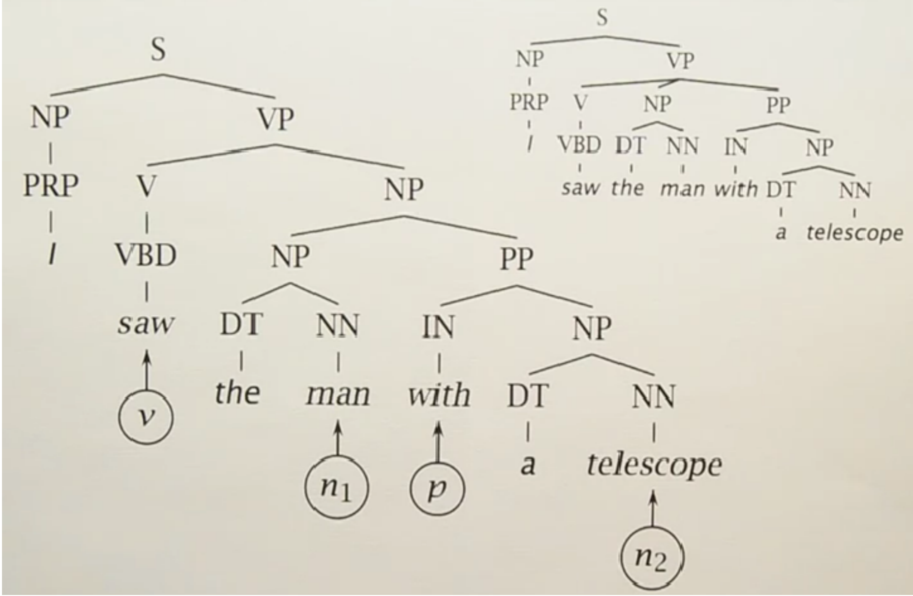
Can you think of a better solution?

- Understand context first?

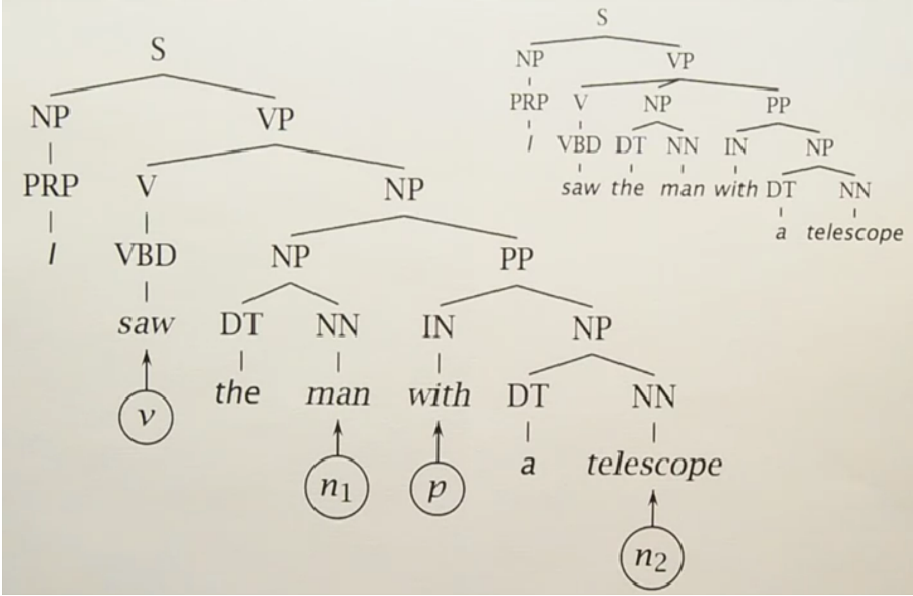
Example Grammar

```
( (S
  (NP-SBJ (DT The) (NN move))
  (VP (VBD followed)
   (NP
    (NP (DT a) (NN round))
    (PP (IN of)
     (NP
      (NP (JJ similar) (NNS increases))
      (PP (IN by)
       (NP (JJ other) (NNS lenders)))
      (PP (IN against)
       (NP (NNP Arizona) (JJ real) (NN estate) (NNS loans))))))
   (, ,)
   (S-ADV
    (NP-SBJ (-NONE- *))
    (VP (VBG reflecting)
     (NP
      (NP (DT a) (VBG continuing) (NN decline))
      (PP-LOC (IN in)
```

Back to Disambiguation with Learned Grammar



Back to Disambiguation with Learned Grammar



Lexicalized PCFG (LPCFG)

OMG! That's a long acronym. 

Lexicalized PCFG (LPCFG)

OMG! That's a long acronym. 

Probabilities based on actual words:

$$
\begin{aligned}
P(\mathrm{VP} \rightarrow \mathrm{V\,NP\,NP} | \mathrm{V} = \mathrm{gave}) &= 0.8\,(\mathrm{common:\ gave\,me\,something}) \\
P(\mathrm{VP} \rightarrow \mathrm{V\,NP\,NP} | \mathrm{V} = \mathrm{kiss}) &= 0.1\,(\mathrm{rare:\ kiss\,me\,goodbte})
\end{aligned}
$$

Lexicalized PCFG (LPCFG)

OMG! That's a long acronym. 

Probabilities based on actual words:

$$\begin{aligned} P(\mathrm{VP} \rightarrow \mathrm{V\,NP\,NP} | \mathrm{V} = \mathrm{gave}) &= 0.8\,(\mathrm{common: \ gave\,me\,something}) \\ P(\mathrm{VP} \rightarrow \mathrm{V\,NP\,NP} | \mathrm{V} = \mathrm{kiss}) &= 0.1\,(\mathrm{rare: \ kiss\,me\,goodbte}) \end{aligned}$$

But telescope example still hard to solve. But we can use:

- Smoothing
- Abstractions

Putting Them Together: Parsing Trees with LPCFGs

So we have all the information now. How to parse language into trees?

Putting Them Together: Parsing Trees with LPCFGs

So we have all the information now. How to parse language into trees? Two options:

1. Start from words (bottom up); like starting from initial state

Putting Them Together: Parsing Trees with LPCFGs

So we have all the information now. How to parse language into trees? Two options:

1. Start from words (bottom up); like starting from initial state
2. Start from sentence (top down); like starting from goal state

Putting Them Together: Parsing Trees with LPCFGs

So we have all the information now. How to parse language into trees? Two options:

1. Start from words (bottom up); like starting from initial state
2. Start from sentence (top down); like starting from goal state

So it becomes like a regular tree search!

Putting Them Together: Parsing Trees with LPCFGs

So we have all the information now. How to parse language into trees? Two options:

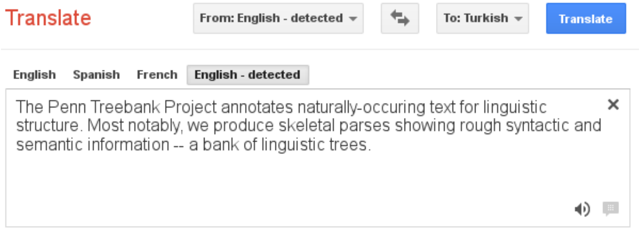
1. Start from words (bottom up); like starting from initial state
2. Start from sentence (top down); like starting from goal state

So it becomes like a regular tree search!

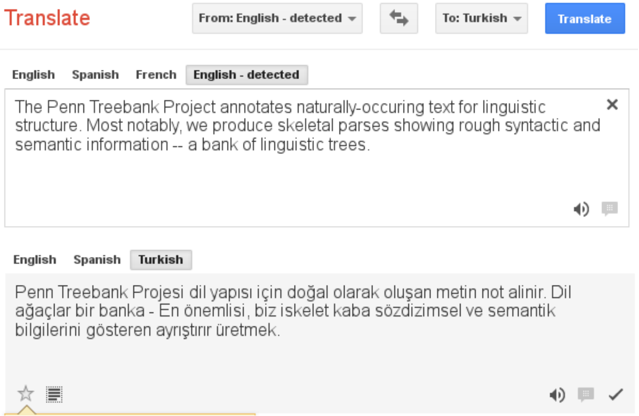
Note:

- Context-free grammars have advantage of parsing parts of the tree **independent** of the rest. That is, we can **divide and conquer**.

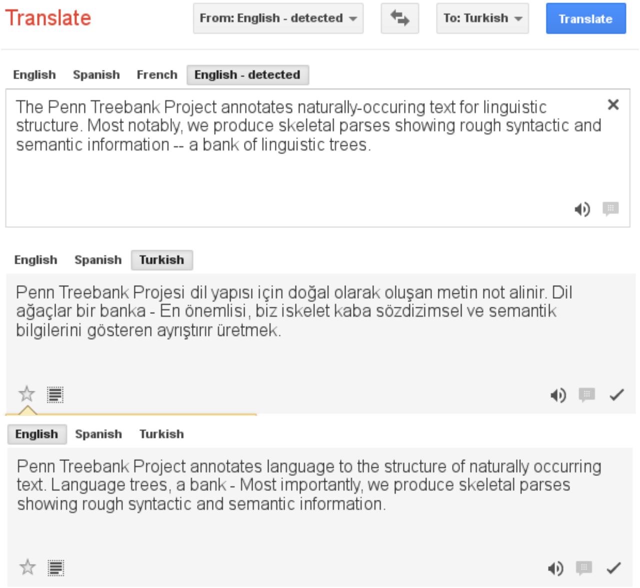
Machine Translation



Machine Translation



Machine Translation



Machine Translation Levels

Multi-level pyramid of machine translation (by Vauquois):

1. Word by word
2. Phrase
3. Tree
4. Meaning (semantic)

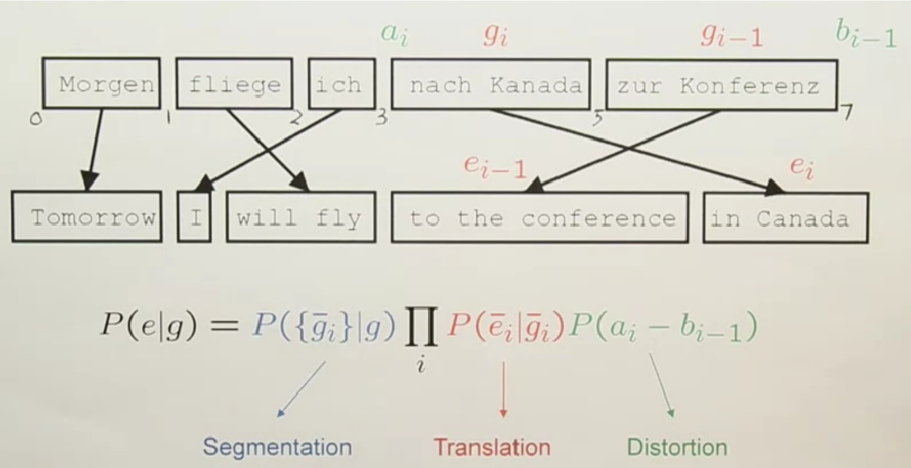
Machine Translation Levels

Multi-level pyramid of machine translation (by Vauquois):

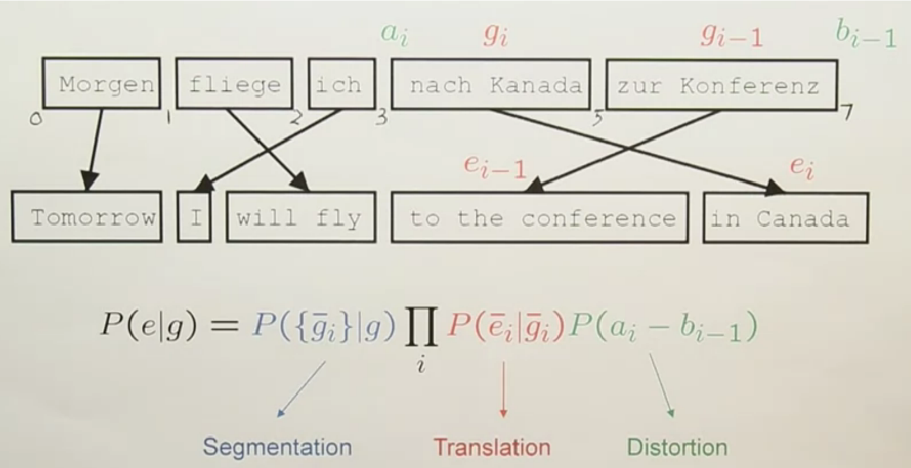
1. Word by word
2. **Phrase**
3. Tree
4. Meaning (semantic)

We'll concentrate on #2, but others are used on the field, too.

Phrase Translation

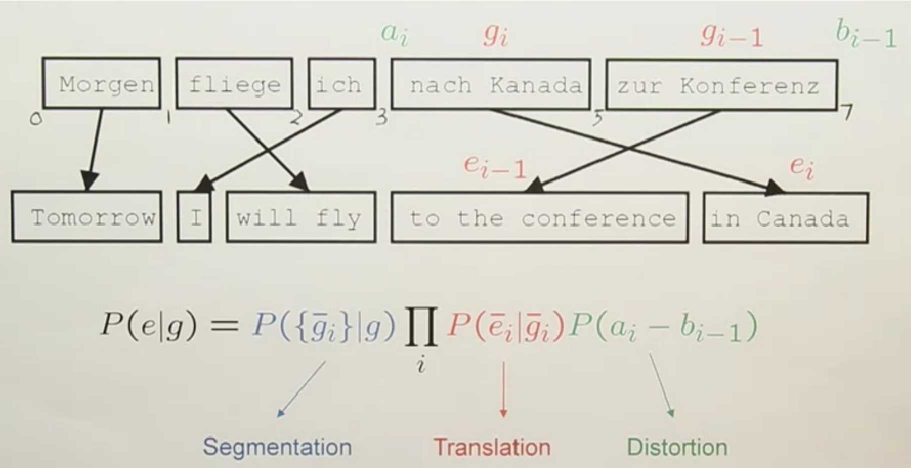


Phrase Translation



What else to improve?

Phrase Translation



What else to improve?

- Calculate $p(e)$ from LPCFG and check if translated sentence is likely.