Lecture 15: Formal grammars of English

Julia Hockenmaier
juliahm@illinois.edu
3324 Siebel Center
Midterm results

22 out of 25 points = 100%
Ugrad median: 15 points = 68.2% ≈ ‘B–’ letter grade
(the same scale will be used for grad students)
Midterm results

Converting **points to percentages:**
- 22 out of 25 points = 100% (you should see both in Compass)

Converting **points/percentages to letter grades:**
The final conversion will be based on the **total percentage** at the end of the semester (MPs, midterm, final, (project))
I use the **undergrads’ performance** as yardstick for everybody

If I had to give letter grades for this midterm, here is a **rough** scale:
- You would need **19 points (~86%)** or more to get an A
- The **undergrad median** (15 points = 68.2%) would correspond to a **B–** letter grade
- You would need **at least 40% (9 points)** to pass the class.
How can you do better?

Come to class, and participate.

Spend time with the material after each lecture.

Read the textbook.

Use Piazza.

Come to office hours.

Let us know if you struggle.
How can we do better?

Please fill in the informal feedback survey, to be sent out by email shortly.
4th credit hour projects

You should have received feedback from me by now.

If I just told you everything is fine, I meant it. (true for most of you)

If you have not heard from me (or if you still have questions), please email me ASAP.

Get started on the actual project!
Back to the lecture
Previous key concepts

NLP tasks dealing with words...
- POS-tagging, morphological analysis

... require finite-state representations,
- Finite-State Automata and Finite-State Transducers

... the corresponding probabilistic models,
- Probabilistic FSAs and Hidden Markov Models
- Estimation: relative frequency estimation, EM algorithm

... and appropriate search algorithms
- Dynamic programming: Forward, Viterbi, Forward-Backward
The next key concepts

NLP tasks dealing with sentences...
- Syntactic parsing and semantic analysis

... require (at least) context-free representations,
- Context-free grammars, unification grammars

... the corresponding probabilistic models,
- Probabilistic Context-Free Grammars, Loglinear models
  - Estimation: Relative Frequency estimation, EM algorithm, etc.

... and appropriate search algorithms
- Dynamic programming: chart parsing, inside-outside algorithm
Dealing with ambiguity

- Search Algorithm (e.g. Viterbi)
- Structural Representation (e.g. FSA)
- Scoring Function (Probability model, e.g. HMM)
Today’s lecture

Introduction to natural language syntax (‘grammar’):

Constituency and dependencies
Context-free Grammars
Dependency Grammars
A simple CFG for English
What is grammar?

No, not really, not in this class
What is grammar?

Grammar formalisms
(= linguists’ programming languages)
A precise way to define and describe the structure of sentences.
(N.B.: There are many different formalisms out there, which each define their own data structures and operations)

Specific grammars
(= linguists’ programs)
Implementations (in a particular formalism) for a particular language (English, Chinese,....)
Can we define a program that generates all English sentences?

The number of sentences is infinite. But we need our program to be finite.
Overgeneration:

John saw Mary.
I ate sushi with tuna.
I want you to go there.
Did you go there?
I ate the cake that John had made for me yesterday.
John made some cake.

Undergeneration:

John Mary saw.
with tuna sushi ate I.
Did you went there?
.....
Basic sentence structure

I eat sushi.

- **Subject** (Noun): I
- **Verb** (Head): eat
- **Object** (Noun): sushi
This is a dependency graph:

I eat sushi.

I eat sushi.
A finite-state-automaton (FSA)
A Hidden Markov Model (HMM)

- **Noun (Subject)**: I, you, ....
- **Verb (Head)**: eat, drink
- **Noun (Object)**: sushi, ...

Diagram: A pathway showing the flow from Noun (Subject) to Verb (Head) to Noun (Object) with examples of phrases.
Words take arguments

- I eat sushi. ✔
- I eat sushi you. ???
- I sleep sushi  ???
- I give sushi  ???
- I drink sushi  ?

Subcategorization
(purely syntactic: what set of arguments do words take?)

**Intransitive verbs** *(sleep)* take only a subject.

**Transitive verbs** *(eat)* take also one (direct) object.

**Ditransitive verbs** *(give)* take also one (indirect) object.

Selectional preferences
(semantic: what types of arguments do words tend to take)

The object of eat should be edible.
A better FSA

Noun (Subject) -> Transitive Verb (Head) -> Noun (Object)

Intransitive Verb (Head)
Language is recursive

the ball
the big ball
the big, red ball
the big, red, heavy ball
....

Adjectives can modify nouns.
The number of modifiers (aka adjuncts) a word can have is (in theory) unlimited.
Another FSA
Recursion can be more complex

the ball
the ball in the garden
the ball in the garden behind the house
the ball in the garden behind the house next to the school

....
Yet another FSA

So, why do we need anything beyond regular (finite-state) grammars?
What does this mean?

There is an attachment ambiguity.
FSAs do not generate hierarchical structure
What is the structure of a sentence?

Sentence structure is **hierarchical**:

A sentence consists of **words** (I, eat, sushi, with, tuna) ..which form phrases or **constituents**: “sushi with tuna”

Sentence structure defines **dependencies** between words or phrases:

```
[I[ eat[ sushi [with tuna]] ]] ]
```
Strong vs. weak generative capacity

Formal language theory:
- defines language as string sets
- is only concerned with generating these strings (weak generative capacity)

Formal/Theoretical syntax (in linguistics):
- defines language as sets of strings with (hidden) structure
- is also concerned with generating the right *structures* (strong generative capacity)
Context-free grammars (CFGs) capture recursion

Language has complex constituents
   (“the garden behind the house”)

Syntactically, these constituents behave just like simple ones.
   (“behind the house” can always be omitted)

CFGs define nonterminal categories to capture equivalent constituents.
Context-free grammars

A CFG is a 4-tuple \( \langle N, \Sigma, R, S \rangle \) consisting of:
- A set of nonterminals \( N \)
  (e.g. \( N = \{ S, NP, VP, PP, Noun, Verb, \ldots \} \))
- A set of terminals \( \Sigma \)
  (e.g. \( \Sigma = \{ I, you, he, eat, drink, sushi, ball, \} \))
- A set of rules \( R \)
  \( R \subseteq \{ A \rightarrow \beta \text{ with left-hand-side (LHS) } A \in N \text{ and right-hand-side (RHS) } \beta \in (N \cup \Sigma)^* \} \)
- A start symbol \( S \in N \)
An example

DT → \{the, a\}
N → \{ball, garden, house, sushi\}
P → \{in, behind, with\}
NP → DT N
NP → NP PP
PP → P NP

N: noun
P: preposition
NP: “noun phrase”
PP: “prepositional phrase”
CFGs define parse trees

N → \{sushi, tuna\}
P → \{with\}
V → \{eat\}
NP → N
NP → NP PP
PP → P NP
VP → V NP
CFGs and center embedding

The mouse ate the corn.
The mouse that the snake ate ate the corn.
The mouse that the snake that the hawk ate ate ate the corn.
....
CFGs and center embedding

Formally, these sentences are all grammatical, because they can be generated by the CFG that is required for the first sentence:

\[
\begin{align*}
S & \rightarrow \text{NP} \text{ VP} \\
\text{NP} & \rightarrow \text{NP} \text{ RelClause} \\
\text{RelClause} & \rightarrow \text{that} \text{ NP ate}
\end{align*}
\]

**Problem:** CFGs are not able to capture **bounded recursion.** (‘only embed one or two relative clauses’).

To deal with this discrepancy between what the model predicts to be grammatical, and what humans consider grammatical, linguists distinguish between a speaker’s **competence** (grammatical knowledge) and **performance** (processing and memory limitations).
CFGs are equivalent to Pushdown automata (PDAs)

PDAs are FSAs with an additional stack:
Emit a symbol and push/pop a symbol from the stack

This is equivalent to the following CFG:

\[
\begin{align*}
S & \to a \ X \ b \quad S \to a \ b \\
X & \to a \ X \ b \quad X \to a \ b
\end{align*}
\]
Generating $a^n b^n$

<table>
<thead>
<tr>
<th>Action</th>
<th>Stack</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Push $x$ on stack. Emit $a$.</td>
<td>$x$</td>
<td>$a$</td>
</tr>
<tr>
<td>2. Push $x$ on stack. Emit $a$.</td>
<td>$xx$</td>
<td>$aa$</td>
</tr>
<tr>
<td>3. Push $x$ on stack. Emit $a$.</td>
<td>$xxx$</td>
<td>$aaa$</td>
</tr>
<tr>
<td>4. Push $x$ on stack. Emit $a$.</td>
<td>$xxxx$</td>
<td>$aaaa$</td>
</tr>
<tr>
<td>5. Pop $x$ off stack. Emit $b$.</td>
<td>$xxx$</td>
<td>$aaaab$</td>
</tr>
<tr>
<td>6. Pop $x$ off stack. Emit $b$.</td>
<td>$xx$</td>
<td>$aaaabb$</td>
</tr>
<tr>
<td>7. Pop $x$ off stack. Emit $b$.</td>
<td>$x$</td>
<td>$aaaabbb$</td>
</tr>
<tr>
<td>8. Pop $x$ off stack. Emit $b$.</td>
<td></td>
<td>$aaaabbbb$</td>
</tr>
</tbody>
</table>
Defining grammars for natural language
Two ways to represent structure

Phrase structure trees

```
VP
 /   
NP    PP
  /   
NP    NP

V eat NP sushi PP with tuna NP
```

```
VP
 /   
NP    PP
  /   
NP    NP

V eat NP sushi PP with chopsticks NP
```

Dependency trees

```
eat sushi with tuna
```

```
eat sushi with chopsticks
```
Structure (syntax) corresponds to meaning (semantics)

Correct analysis

Incorrect analysis
Dependency grammar

DGs describe the structure of sentences as a directed acyclic graph.

The **nodes** of the graph are the **words**

The **edges** of the graph are the **dependencies**.

Typically, the graph is assumed to be a **tree**.

Note: the relationship between DG and CFGs:

If a CFG phrase structure tree is translated into DG, the resulting dependency graph has no crossing edges.
Constituents:
Heads and dependents

There are different kinds of constituents:

**Noun phrases:** the man, a girl with glasses, Illinois
**Prepositional phrases:** with glasses, in the garden
**Verb phrases:** eat sushi, sleep, sleep soundly

Every phrase has a **head:**

**Noun phrases:** the *man*, a *girl* with glasses, *Illinois*
**Prepositional phrases:** *with* glasses, *in* the garden
**Verb phrases:** *eat* sushi, *sleep*, *sleep* soundly

The other parts are its **dependents.**
Dependents are either **arguments** or **adjuncts**
Is string $\alpha$ a constituent?

He talks [in class].

Substitution test:
Can $\alpha$ be replaced by a single word?
He talks [there].

Movement test:
Can $\alpha$ be moved around in the sentence?
[In class], he talks.

Answer test:
Can $\alpha$ be the answer to a question?
Where does he talk? - [In class].
Arguments are obligatory

Words subcategorize for specific sets of arguments:
  Transitive verbs (sbj + obj): [John] likes [Mary]

All arguments have to be present:
  *[John] likes.  *likes [Mary].

No argument can be occupied multiple times:
  *[John] [Peter] likes [Ann] [Mary].

Words can have multiple subcat frames:
  Transitive eat (sbj + obj): [John] eats [sushi].
Adjuncts are optional

Adverbs, PPs and adjectives can be adjuncts:

Adverbs: John runs [fast].
          a [very] heavy book.
PPs:     John runs [in the gym].
          the book [on the table]
Adjectives: a [heavy] book

There can be an arbitrary number of adjuncts:

John saw Mary.
John saw Mary [yesterday].
John saw Mary [yesterday] [in town]
John saw Mary [yesterday] [in town] [during lunch]
Perhaps] John saw Mary [yesterday] [in town] [during lunch]
A context-free grammar for a fragment of English
Noun phrases (NPs)

Simple NPs:
[He] sleeps. (pronoun)
[John] sleeps. (proper name)
[A student] sleeps. (determiner + noun)

Complex NPs:
[A tall student] sleeps. (det + adj + noun)
[The student in the back] sleeps. (NP + PP)
[The student who likes MTV] sleeps. (NP + Relative Clause)
The NP fragment

NP → Pronoun
NP → ProperName
NP → Det Noun

Det → \{a, the, every\}
Pronoun → \{he, she,...\}
ProperName → \{John, Mary,...\}
Noun → AdjP Noun
Noun → N
NP → NP PP
NP → NP RelClause
Adjective phrases (AdjP) and prepositional phrases (PP)

\[
\begin{align*}
\text{AdjP} & \rightarrow \text{Adj} \\
\text{AdjP} & \rightarrow \text{Adv AdjP} \\
\text{Adj} & \rightarrow \{\text{big, small, red,...}\} \\
\text{Adv} & \rightarrow \{\text{very, really,...}\}
\end{align*}
\]

\[
\begin{align*}
\text{PP} & \rightarrow \text{P NP} \\
\text{P} & \rightarrow \{\text{with, in, above,...}\}
\end{align*}
\]
The verb phrase (VP)

*He [eats].*
*He [eats sushi].*
*He [gives John sushi].*
*He [eats sushi with chopsticks].*

\[
\begin{align*}
\text{VP} & \rightarrow \text{V} \\
\text{VP} & \rightarrow \text{V NP} \\
\text{VP} & \rightarrow \text{V NP PP} \\
\text{VP} & \rightarrow \text{VP PP} \\
\text{V} & \rightarrow \{\text{eats, sleeps gives,...}\}
\end{align*}
\]
Capturing subcategorization

He [eats]. ✔
He [eats sushi]. ✔
He [gives John sushi]. ✔
He [eats sushi with chopsticks]. ✔
*He [eats John sushi]. ????

\[
\begin{align*}
VP & \rightarrow V_{\text{intrans}} \\
VP & \rightarrow V_{\text{trans}} \ NP \\
VP & \rightarrow V_{\text{ditrans}} \ NP \ NP \\
VP & \rightarrow VP \ PP \\
V_{\text{intrans}} & \rightarrow \{eats, sleeps\} \\
V_{\text{trans}} & \rightarrow \{eats\} \\
V_{\text{trans}} & \rightarrow \{gives\}
\end{align*}
\]
Sentences

[He eats sushi].
[Sometimes, he eats sushi].
[In Japan, he eats sushi].

S → NP VP
S → AdvP S
S → PP S

He says [he eats sushi].
VP → Vcomp S
Vcomp → {says, think, believes}
Sentences redefined

[He eats sushi]. ✔
*[I eats sushi].  ???
*[They eats sushi].  ???

S → NP\(_{3\text{sg}}\) VP\(_{3\text{sg}}\)
S → NP\(_{1\text{sg}}\) VP\(_{1\text{sg}}\)
S → NP\(_{3\text{pl}}\) VP\(_{3\text{pl}}\)

We need features to capture agreement:
(number, person, case,...)
Complex VPs

In English, simple tenses have separate forms:

present tense: the girl eats sushi
simple past tense: the girl ate sushi

Complex tenses, progressive aspect and passive voice consist of auxiliaries and participles:

past perfect tense: the girl has eaten sushi
future perfect: the girl will have eaten sushi
passive voice: the sushi was eaten by the girl
progressive: the girl is/was/will be eating sushi
VPs redefined

*He [has [eaten sushi]].*

*The sushi [was [eaten by him]].*

\[
\begin{align*}
\text{VP} & \rightarrow \text{V} \text{have } \text{VP}_{\text{pastPart}} \\
\text{VP} & \rightarrow \text{V} \text{be } \text{VP}_{\text{pass}} \\
\text{VP}_{\text{pastPart}} & \rightarrow \text{V}_{\text{pastPart}} \text{ NP} \\
\text{VP}_{\text{pass}} & \rightarrow \text{V}_{\text{pastPart}} \text{ PP} \\
\text{V} \text{have} & \rightarrow \{\text{has}\} \\
\text{V}_{\text{pastPart}} & \rightarrow \{\text{eaten, seen}\}
\end{align*}
\]

We need more nonterminals (e.g. \(\text{VP}_{\text{pastPart}}\)).

N.B.: We call \(\text{VP}_{\text{pastPart}}, \text{VP}_{\text{pass}},\) etc. ‘untensed’ VPs
Coordination

[He eats sushi] and [she drinks tea]
[John] and [Mary] eat sushi.
He [eats sushi] and [drinks tea]

\[ S \rightarrow S \text{ conj } S \]
\[ NP \rightarrow NP \text{ conj } NP \]
\[ VP \rightarrow VP \text{ conj } VP \]

He says [he eats sushi].
\[ VP \rightarrow V_{\text{comp}} S \]
\[ V_{\text{comp}} \rightarrow \{\text{says, think, believes}\} \]
Relative clauses

Relative clauses modify a noun phrase:
the girl [that eats sushi]

Relative clauses lack a noun phrase, which is understood to be filled by the NP they modify:
‘the girl that eats sushi’ implies ‘the girl eats sushi’

There are subject and object relative clauses:
subject: ‘the girl that eats sushi’
object: ‘the sushi that the girl eats’
Yes/No questions

Yes/no questions consist of an auxiliary, a subject and an (untensed) verb phrase:

does she eat sushi?
have you eaten sushi?

YesNoQ → Aux  NP  VP_{inf}
YesNoQ → Aux  NP  VP_{pastPart}
Wh-questions

Subject wh-questions consist of an wh-word, an auxiliary and an (untensed) verb phrase:

*Who has eaten the sushi?*

Object wh-questions consist of an wh-word, an auxiliary, an NP and an (untensed) verb phrase:

*What does Mary eat?*
Today’s key concepts

Natural language syntax
  Constituents
  Dependencies
  Context-free grammar
  Arguments and modifiers
  Recursion in natural language
Today’s reading

Textbook:
   Jurafsky and Martin, Chapter 12, sections 1-7