CSC 2517: Discrete Mathematical Models of Sentence Structure

- Gerald Penn, PT 283A, gpenn@cs.toronto.edu, Tel: (416)978-7390

- This class will meet during A&S reading week!

- But we will not meet on Wednesday, 22nd March.
CSC 2517: Discrete Mathematical Models of Sentence Structure

- This is an advanced graduate seminar:
  - I will assume that you are familiar with the material of CSC 2501, although graduate seminars do not formally enforce prerequisites.
  - No programming assignments, although your final paper may involve some.
  - Classes will hopefully be more interactive than normal lectures.
  - You will do much of the presenting.

- If any of this doesn’t sound like what you signed up for, then you probably belong in CSC 2511, Natural Language Computing, which is also being offered this term.
This year, presentations will be of papers chosen from among the following topics:

- Algebraic Topology
- Algebraic Invariance
- Graphical/Algebraic Methods for Natural Language
- Geometric Deep Learning
How I will compute your final mark for the class:
- 30% your presentation(s) and participation in the seminar.
- 70% a final paper, due on Friday, 28\textsuperscript{th} April.
- Paper proposals are due on Tuesday, 14\textsuperscript{th} March.
- Auditors are welcome, but they must present, just like everyone else.
- Group papers must be approved in advance. If I approve, everyone in the group will receive the same mark.
- Your final paper must be on the subject of mathematical linguistics, broadly construed. It needn’t be on one of the presentation topics.
- You may submit research you are conducting as part of your thesis or dissertation.
- This is a Methods Area 1 class. In terms of length and style, think of your final papers as MOL or WoLLIC conference papers. Actually submitting to such a conference is encouraged but not required.
CSC 2517: Discrete Mathematical Models of Sentence Structure

- Rules for presentations:
  - Students will pick topics two weeks in advance of their presentation date.
  - I reserve the right to reject papers on the grounds that they are:
    - unsuitably difficult,
    - unsuitably bad,
    - insufficiently related to the topics of this seminar, or
    - excessively devoted to material already covered.
  - Unless invited, you may not present your own research.
  - I will also be offering pre-approved paper(s) to present.
More rules for presentations:
- The papers that we will be reading are highly technical. Expect your presentations to be approximately 1 hour long, or 1 hour of a 2-hour presentation that you will jointly give with another student (for longer papers and collections of related papers).
- Your job as presenter is to teach the material. That not only includes the research presented in the paper, but the background material necessary to understand it.
- Assume that no one has read the paper or understands what we have not discussed in class yet.
- Your job as a non-presenter is to refute this assumption: read the paper, look up the background you are missing and come to class with questions.
- Presenters must defend the work that they are presenting.

Think of these as opportunities not to present a conference paper, but to teach a class – it’s good practice.
Combinatory Categorial Grammar
Combinatory Categorial Grammar (CCG)

- Categorial grammar (CG) is one of the oldest grammar formalisms
- *Combinatory* Categorial Grammar now well established and computationally well founded (Steedman, 1996, 2000)
- Account of syntax; semantics; prosody and information structure; automatic parsers; generation
Combinatory Categorial Grammar (CCG)

- CCG is a lexicalized grammar
- An elementary syntactic structure – for CCG a lexical category – is assigned to each word in a sentence

*walked*: S\NP “give me an NP to my left and I return a sentence”

- A small number of rules define how categories can combine
  - Rules based on the combinators from Combinatory Logic
CCG Lexical Categories

- Atomic categories: S, N, NP, PP, ... (not many more)
- Complex categories are built recursively from atomic categories and slashes, which indicate the directions of arguments
- Complex categories encode subcategorisation information
  - intransitive verb: S \NP walked
  - transitive verb: (S \NP )/NP respected
  - ditransitive verb: ((S \NP )/NP )/NP gave
- Complex categories can encode modification
  - PP nominal: (NP \NP )/NP
  - PP verbal: ((S \NP )\(S \NP ))/NP
Simple CCG Derivation

\[ \text{interleukin} - 10 \quad \underset{NP}{\text{inhibits}} \quad \underset{(S\backslash NP)/NP}{\text{production}} \quad \underset{NP}{S\backslash NP} \quad \underset{S}{\text{}} \]

> forward application
< backward application
Function Application Schemata

- Forward (>), and backward (<) application:

\[
\begin{align*}
X / Y & \quad Y \quad \Rightarrow \quad X \quad (>)
Y & \quad X \backslash Y \quad \Rightarrow \quad X \quad (<)
\end{align*}
\]
Classical Categorial Grammar

- ‘Classical’ Categorial Grammar only has application rules
- Classical Categorial Grammar is context free

Diagram:

```
S
 / \  /
/   \ /   
NP  (S\NP)/NP NP
      /   /
interleukin-10 inhibits production
```
Classical Categorial Grammar

- ‘Classical’ Categorial Grammar only has application rules
- Classical Categorial Grammar is context free

```
NP        V        NP
interleukin-10 inhibits production
```

```
S
  VP
    NP
      interleukin-10
    V
    NP
      inhibits
      production
```
Extraction out of a Relative Clause

The company which Microsoft bought

NP/N  N  (NP\NP)/(S/NP)  NP  (S\NP)/NP
Extraction out of a Relative Clause

The company which Microsoft bought

NP/N N (NP\NP)/(S/NP) NP (S\NP)/NP

S/(S\NP) > T

> T type-raising
Extraction out of a Relative Clause

The company which Microsoft bought

NP/N N (NP\NP)/(S/NP) NP (S\NP)/NP

S/(S\NP) >T S/NP >B

> T type-raising

> B forward composition

Stephen Clark

Practical Linguistically Motivated Parsing

JHU, June 2009
Extraction out of a Relative Clause

The company which Microsoft bought

\[
\begin{align*}
\text{NP} / \text{N} & \quad \text{NP} / \text{NP} & \quad \text{NP} / (S / \text{NP}) \\
\text{N} & \quad (NP \backslash NP) & \quad (S \backslash NP) / \text{NP} \\
\text{NP} \backslash \text{NP} & \quad S / (S \backslash \text{NP}) & \quad \text{NP} / \text{NP}
\end{align*}
\]
Extraction out of a Relative Clause

The company which Microsoft bought

NP/N N (NP\NP)/(S/NP) NP (S\NP)/NP

NP

\[ \frac{\text{NP/N}}{\text{NP}} \]

NP

NP

\[ \frac{\text{S}/(S\NP)}{\text{NP}} \]

NP

\[ \frac{\text{S}/\NP}{\text{NP}} \]

NP

\[ \frac{\text{NP}/\NP}{\text{NP}} \]

Stephen Clark

Practical Linguistically Motivated Parsing

JHU, June 2009
Forward Composition and Type-Raising

- Forward composition ($\succ_B$):
  
  \[ X/Y \ Y/Z \Rightarrow X/Z \ (\succ_B) \]

- Type-raising ($T$):
  
  \[ X \Rightarrow T/(T\backslash X) \ (\succ_T) \]
  \[ X \Rightarrow T\backslash(T/X) \ (\prec_T) \]

- Extra combinatory rules increase the weak generative power to mild context-sensitivity
“Non-constituents” in CCG – Right Node Raising

Google sells but Microsoft buys shares

\[
\begin{align*}
\text{Google} & \quad \text{sells} & \quad \text{but} & \quad \text{Microsoft} & \quad \text{buys} & \quad \text{shares} \\
NP & \quad (S\backslash NP)/NP & \quad \text{conj} & \quad NP & \quad (S\backslash NP)/NP & \quad \text{NP} \\
S/(S\backslash NP) & \quad \xrightarrow{T} & \quad & \quad S/(S\backslash NP) & \quad \xrightarrow{T} \\
\end{align*}
\]

> \text{T type-raising}
“Non-constituents” in CCG – Right Node Raising

Google sells but Microsoft buys shares

\[
\begin{align*}
\text{Google} & \quad \text{sells} & \quad \text{but} & \quad \text{Microsoft} & \quad \text{buys} & \quad \text{shares} \\
NP & \quad (S\backslash NP)/NP & \quad \text{conj} & \quad NP & \quad (S\backslash NP)/NP & \quad NP \\
S/(S\backslash NP) & \quad \xrightarrow{T} & \quad S/\text{NP} & \quad \xrightarrow{T} & \quad S/\text{NP} & \quad \xrightarrow{T} \\
\end{align*}
\]

\[S/\text{NP} \xrightarrow{B} \]

\[S/\text{NP} \xrightarrow{B} \]

\[> T \quad \text{type-raising} \]

\[> B \quad \text{forward composition} \]
"Non-constituents" in CCG – Right Node Raising

Google sells but Microsoft buys shares

NP (S\NP) / NP conj NP (S\NP) / NP NP

S/(S\NP) >T S/NP B S/(S\NP) >T S/NP B

S/NP <Φ>
“Non-constituents” in CCG – Right Node Raising

\[
\begin{align*}
\text{Google} & \quad \text{sells} \quad \text{but} \quad \text{Microsoft} \quad \text{buys} \quad \text{shares} \\
\text{NP} & \quad \frac{(S\backslash NP)/NP}{\text{conj}} \quad \frac{NP}{(S\backslash NP)/NP} \quad \frac{NP}{NP} \\
S/(S\backslash NP) & \quad \Rightarrow \text{T} \quad \frac{S/(S\backslash NP)}{\text{conj}} \quad \frac{S/(S\backslash NP)}{\text{T}} \\
S/NP & \quad \Rightarrow \text{B} \quad S/NP \quad \Rightarrow \text{B} \\
\Phi & \quad \frac{S/NP}{\text{conj}} \quad \frac{S/NP}{\text{T}} \\
S & \quad \Rightarrow
\end{align*}
\]

Stephen Clark
Practical Linguistically Motivated Parsing
JHU, June 2009
Combinatory Categorial Grammar

- \textit{CCG} is \textit{mildly} context sensitive
- Natural language is provably non-context free
- Constructions in Dutch and Swiss German (Shieber, 1985) require more than context free power for their analysis
  - these have crossing dependencies (which \textit{CCG} can handle)

\begin{center}
\begin{tikzpicture}
  \node (type0) at (0,0) {Type 0 languages};
  \node (context敏感) at (0,-2) {Context sensitive languages};
  \node (context自由) at (0,-4) {Context free languages};
  \node (regular) at (0,-6) {Regular languages};
  \node (自然) at (0,-3) {Mildly context sensitive languages = natural languages (?)};

  \draw (type0) -- (context敏感) -- (context自由) -- (regular);
  \draw (自然) -- (context敏感) -- (自然);
\end{tikzpicture}
\end{center}
CCG Semantics

- Categories encode argument sequences
- Parallel syntactic combinator operations and lambda calculus semantic operations

\[
\begin{align*}
John &\vdash NP : john' \\
shares &\vdash NP : shares' \\
buys &\vdash (S\setminus NP)/NP : \lambda x.\lambda y.\text{buys}' xy \\
sleeps &\vdash S\setminus NP : \lambda x.\text{sleeps}' x \\
well &\vdash (S\setminus NP)\setminus(S\setminus NP) : \lambda f.\lambda x.\text{well}'(fx) \\
\end{align*}
\]
# CCG Semantics

<table>
<thead>
<tr>
<th>Left arg.</th>
<th>Right arg.</th>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>(X/Y : f)</td>
<td>(Y : a)</td>
<td>Forward application</td>
<td>(X : f(a))</td>
</tr>
<tr>
<td>(Y : a)</td>
<td>(X/Y : f)</td>
<td>Backward application</td>
<td>(X : f(a))</td>
</tr>
<tr>
<td>(X/Y : f)</td>
<td>(Y/Z : g)</td>
<td>Forward composition</td>
<td>(X/Z : \lambda x.f(g(x)))</td>
</tr>
<tr>
<td>(X : a)</td>
<td></td>
<td>Type raising</td>
<td>(T/(T\backslash X) : \lambda f.f(a))</td>
</tr>
</tbody>
</table>

etc.
Tree Adjoining Grammar
TAG Building Blocks

- Elementary trees (of many depths)
- Substitution at ↓
- Tree Substitution Grammar equivalent to CFG

\[
\alpha_3 \quad \text{NP} \quad \text{peanuts} \\
\alpha_1 \quad \text{NP} \quad \text{Harry} \\
\alpha_2 \quad \text{S} \quad \text{NP} \downarrow \quad \text{VP} \quad \text{V} \quad \text{likes} \quad \text{NP} \downarrow
\]
TAG Building Blocks

- Auxiliary trees for *adjunction*
- Adds extra power beyond CFG

\[
\begin{align*}
\alpha_1 & : NP \downarrow \text{Harry} \\
\alpha_2 & : S \downarrow \text{VP} \\
\alpha_3 & : NP \downarrow \text{peanuts} \\
\beta & : VP \downarrow \text{Adv} \downarrow \text{passionately}
\end{align*}
\]
Harry likes peanuts passionately.

Semantics

$$Harry(x) \land likes(e, x, y) \land peanuts(y) \land passionately(e)$$
Why Supertag?

- If lexical items have more description associated with them, parsing is easier
  - Only useful if the supertag space is not huge

- Straightforward to compile parse from accurate supertagging
  - But impossible if there are any supertag errors
    - We can account for *some* supertag errors
    - Don’t always want a full parse anyway
WHAT IS SUPERTAGGING?

- Systematic assignment of supertags
- Supertags are:
  - Statistically selected
    - Robust
    - Tends to work
  - Linguistically motivated
    - This makes sense
What is supertagging?

- Many supertags for each word
  - Extended Domain of Locality
    - Each lexical item has one supertag for every syntactic environment it appears in
    - Inspiration comes from LTAG, lexicalized tree-adjoining grammars, in which all dependencies are localized.
    - Generally, agreement features such as number and tense, are not part of the supertag.
HOW TO SUPERTAG

“Alice opened her eyes and saw.”

Supertags:
- Verb
  - Transitive verb
  - Intransitive verb
  - Infinitive verb
  - ...
- Noun
  - Noun phrase (subject)
  - Nominal predicative
  - Nominal modifier
  - Nominal predicative subject extraction
  - ...

\[\text{Verb}\]

\[\text{Noun}\]

\[\text{Supertags}\]

\[\text{Alice opened her eyes and saw.}\]

\[\text{Verb}\]

\[\text{Noun}\]

\[\text{Supertags}\]

\[\text{Alice opened her eyes and saw.}\]
HOW TO SUPERTAG

“Alice opened her eyes and saw.”

● Supertags:
  ● Verb
    ○ Transitive verb
    ○ Intransitive verb
    ○ Infinitive verb
    ○ ...
  ● Noun
    ○ Noun phrase (subject)
    ○ Nominal predicative
    ○ Nominal modifier
    ○ Nominal predicative subject extraction
    ○ ...
HOW TO SUPERTAG

- A supertag can be ruled out for a given word in a given input string...
  - Left and/or right context is too long/short for the input
  - If the supertag contains other terminals not found in the input
HOW TO SUPERTAG

“Alice opened her eyes and saw.”

- Supertags:
  - Verb
    - Transitive verb
    - Intransitive verb
    - Infinitive verb
    - ...
  - Noun
    - Noun phrase (subject)
    - Nominal predicative
    - Nominal modifier
    - Nominal predicative subject extraction
    - ...

... to saw ...

...
**HOW TO SUPERTAG**

- This works fairly well
  - 50% average reduction in number of possible supertags
HOW TO SUPERTAG

...but there’s more to be done

- Good: average number of possible supertags per word reduced from 47 to 25
- Bad: average of 25 possible supertags per word
How to Supertag

- Disambiguation by unigrams?
  - Give each word its most frequent supertag after PoS tagging
    - ~75% accurate
      - Better results than one might expect given large number of possible supertags
      - Common words (determiners, etc.) usually correct
        - This helps accuracy
      - Back off to PoS for unknown words
        - Also usually correct
**How to Supertag**

- **Disambiguation by n-grams?**

  $$T = \arg \max_{T} \Pr(T_1, T_2, ..., T_N) \times \Pr(W_1, W_2, ..., W_N|T_1, T_2, ..., T_N)$$

  - We assume that subsequent words are independent

    $$\Pr(W_1, W_2, ..., W_N|T_1, T_2, ..., T_N) \approx \prod_{i=1}^{N} \Pr(W_i|T_i)$$

  - Trigrams plus Good-Turing smoothing
    - Accuracy around 90%
      - Versus 75% from unigrams
    - Contextual information more important than lexical
      - Reversal of trend for PoS tagging
However...

- Correctly supertagged text yields a 30X parsing speedup
  - But even one mistake can cause parsing to fail completely
    - This is rather likely

- Solution: n-best supertags?
  - When n=3, we get up to 96% accuracy...
    - Not bad at all for such a simple method
    - 425 lexical categories (PTB-CFG: ~50)
    - 12 combinatory rules (PTB-CFG: > 500,000)