Outline

• Dependency Parsing
  • Formal definition
  • Dynamic programming
  • Supervised Classification

• Semantic Role Labeling
  • Propbank
  • Automatic SRL
  • FrameNet
Interpreting Language is Hard!

I saw a girl with a telescope

- “Parsing” resolves structural ambiguity in a formal way
Two Types of Parsing

- **Dependency**: focuses on relations between words

  ![Dependency Diagram](image)

  *I saw a girl with a telescope*

- **Phrase structure**: focuses on identifying phrases and their recursive structure

  ![Phrase Structure Diagram](image)

  *I saw a girl with a telescope*
Dependencies Also Resolve Ambiguity

I saw a girl with a telescope

I saw a girl with a telescope
Dependencies

- **Typed**: Label indicating relationship between words

  ![Diagram of typed dependencies]

  I saw a girl with a telescope

- **Untyped**: Only which words depend

  ![Diagram of untyped dependencies]

  I saw a girl with a telescope
Dependency Grammars

• In CFG-style phrase-structure grammars the main focus is on *constituents*.
• But it turns out you can get a lot done with just binary relations among the words in an utterance.
• In a *dependency grammar* framework, a parse is a tree where
  • the nodes stand for the words in an utterance
  • The links between the words represent dependency relations between pairs of words.
    • Relations may be typed (labeled), or not.
### Dependency Relations

<table>
<thead>
<tr>
<th>Argument Dependencies</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nsubj</td>
<td>nominal subject</td>
</tr>
<tr>
<td>csubj</td>
<td>clausal subject</td>
</tr>
<tr>
<td>dobj</td>
<td>direct object</td>
</tr>
<tr>
<td>iobj</td>
<td>indirect object</td>
</tr>
<tr>
<td>pobj</td>
<td>object of preposition</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Modifier Dependencies</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tmod</td>
<td>temporal modifier</td>
</tr>
<tr>
<td>appos</td>
<td>appositional modifier</td>
</tr>
<tr>
<td>det</td>
<td>determiner</td>
</tr>
<tr>
<td>prep</td>
<td>prepositional modifier</td>
</tr>
</tbody>
</table>
They hid the letter on the shelf
Dependency Parsing

- The dependency approach has a number of advantages over full phrase-structure parsing.
  - Deals well with free word order languages where the constituent structure is quite fluid
  - Parsing is much faster than CFG-bases parsers
  - Dependency structure often captures the syntactic relations needed by later applications
    - CFG-based approaches often extract this same information from trees anyway.
Dependency Parsing

• There are two modern approaches to dependency parsing
  • Optimization-based approaches that search a space of trees for the tree that *best* matches some criteria
  • Shift-reduce approaches that greedily take actions based on the current word and state.
Phrase Structure Tree

```
S
  /\  \
NP  VP
  /\  /\  \\\
NP  PP  NP
     /\  /\  \\
JJ  NNS IN DT NNS VBD VBG NNS
```

Red figures on the screens indicated falling stocks
Dependency Grammar

• Syntactic structure consists of **lexical items**, linked by binary asymmetric relations called **dependencies**

• Interested in grammatical relations between individual words (**governing** & **dependent** words)

• Does not propose a recursive structure
  • Rather a network of relations

• These relations can also have labels
Draw the dependency tree

• Red figures on the screens indicated falling stocks
Dependency Tree

Red figures on the screens indicated falling stocks
Dependency Tree Example

- Phrasal nodes are missing in the dependency structure when compared to constituency structure.
Dependency Tree with Labels

Red figures on the screens indicated falling stocks
Comparison

• Dependency structures explicitly represent
  • Head-dependent relations (directed arcs)
  • Functional categories (arc labels)
  • Possibly some structural categories (parts-of-speech)

• Phrase structure explicitly represent
  • Phrases (non-terminal nodes)
  • Structural categories (non-terminal labels)
  • Possibly some functional categories (grammatical functions)
Learning DG over PSG

- Dependency Parsing is more straightforward
  - Parsing can be reduced to labeling each token $w_i$ with $w_j$

- Direct encoding of predicate-argument structure
  - Fragments are directly interpretable

- Dependency structure independent of word order
  - Suitable for free word order languages (like Indian languages)
Dependency Tree

• Formal definition
  • An input word sequence $w_1 \ldots w_n$
  • Dependency graph $D = (W,E)$ where
    • $W$ is the set of nodes i.e. word tokens in the input seq.
    • $E$ is the set of unlabeled tree edges $(w_i, w_j)$ ($w_i, w_j \in W$).
    • $(w_i, w_j)$ indicates an edge from $w_i$ (parent) to $w_j$ (child).

• Task of mapping an input string to a dependency graph satisfying certain conditions is called dependency parsing
Well-formedness

- A dependency graph is well-formed iff
  - **Single head**: Each word has only one head.
  - **Acyclic**: The graph should be acyclic.
  - **Connected**: The graph should be a single tree with all the words in the sentence.
  - **Projective**: If word A depends on word B, then all words between A and B are also subordinate to B (i.e. dominated by B).
Dependency Parsing

• Dependency based parsers can be broadly categorized into
  • Grammar driven approaches
    • Parsing done using grammars.
  • Data driven approaches
    • Parsing by training on annotated/un-annotated data.
Dependency Parsing

- Dependency based parsers can be broadly categorized into
  - Grammar driven approaches
    - Parsing done using grammars.
  - Data driven approaches
    - Parsing by training on annotated/un-annotated data.

- These approaches are not mutually exclusive
Covington’s Incremental Algorithm

- Incremental parsing in $O(n^2)$ time by trying to link each new word to each preceding one [Covington 2001]:

\[
\text{PARSE}(x = (w_1, \ldots, w_n))
\]

1. \text{for } i = 1 \text{ up to } n
2. \text{for } j = i - 1 \text{ down to } 1
3. \text{LINK}(w_i, w_j)
Covington’s Incremental Algorithm

- Incremental parsing in $O(n^2)$ time by trying to link each new word to each preceding one [Covington 2001]:

  \[
  \text{PARSE}(x = (w_1, \ldots , w_n))
  \]

  1. \textbf{for} $i = 1$ \textbf{up to} $n$
  2. \textbf{for} $j = i - 1$ \textbf{down to} 1
  3. \textbf{LINK}(w_i, w_j)

- Different conditions, such as Single-Head and Projectivity, can be incorporated into the LINK operation.
Dynamic Programming

- Basic Idea: Treat dependencies as constituents.
- Use, e.g., CYK parser (with minor modifications)
Dynamic Programming Approaches

- Original version [Hays 1964] (grammar driven)
- Link grammar [Sleator and Temperley 1991] (grammar driven)
- Bilexical grammar [Eisner 1996] (data driven)
- Maximum spanning tree [McDonald 2006] (data driven)
Eisner 1996

- Two novel aspects:
  - Modified parsing algorithm
  - Probabilistic dependency parsing
- Time requirement: $O(n^3)$
- Modification: Instead of storing subtrees, store spans
- Span: Substring such that no interior word links to any word outside the span.
- Idea: In a span, only the boundary words are active, i.e. still need a head or a child
- One or both of the boundary words can be active
Example

(ROOT) Red figures on the screen indicated falling stocks
Example

Red figures on the screen indicated falling stocks

Spans:

Red figures
indicated falling stocks
Assembly of correct parse

Start by combining adjacent words to minimal spans
Assembly of correct parse

Combine spans which overlap in one word; this word must be governed by a word in the left or right span.
Assembly of correct parse

Combine spans which overlap in one word; this word must be governed by a word in the left or right span.
Assembly of correct parse

Combine spans which overlap in one word; this word must be governed by a word in the left or right span.

Invalid span
Assembly of correct parse

Combine spans which overlap in one word; this word must be governed by a word in the left or right span.

\[
\{\text{indicated}, \text{falling}\} + \{\text{falling}, \text{stocks}\} \rightarrow \{\text{indicated}, \text{falling}, \text{stocks}\}
\]
Classifier-Based Parsing

- Data-driven deterministic parsing:
  - Deterministic parsing requires an oracle.
  - An oracle can be approximated by a classifier.
  - A classifier can be trained using treebank data.

- Learning algorithms:
  - Memory-based learning (MBL) [Nivre et al. 2004, Nivre and Scholz 2004]
  - Maximum entropy modeling (MaxEnt) [Cheng et al. 2005]
Feature Models

• Learning problem:
  • Approximate a function from parser states, represented by feature vectors to parser actions, given a training set of gold standard derivations.

• Typical features:
  • Tokens and POS tags of:
    • Target words
    • Linear context (neighbors in S and Q)
    • Structural context (parents, children, siblings in G)
    • Can not be used in dynamic programming algorithms.
Feature Models

Maximum Spanning Tree

- Each dependency is an edge in a directed graph
- Assign each edge a score (with machine learning)
- Keep the tree with the highest score

Graph

```
I  a  girl
```

Scored Graph

```
I  a  girl
6  6
-1  2
1    4
-2  7
1
5
```

Dependency Tree

```
I  girl
```

(Chu-Liu-Edmonds Algorithm)
Dependency Parsers for download

- MST parser by Ryan McDonald
- Malt parser by Joakim Nivre
- Stanford parser
Outline

• Dependency Parsing
  • Formal definition
  • Dynamic programming
  • Supervised Classification

• Semantic Role Labeling
  • Propbank
  • Automatic SRL
  • FrameNet
What is PropBank:
From Sentences to Propositions

Powell met Zhu Rongji

Powell and Zhu Rongji met

Powell met with Zhu Rongji

Powell and Zhu Rongji had a meeting

When Powell met Zhu Rongji on Thursday they discussed the return of the spy plane.

Proposition: meet(Powell, Zhu Rongji)

meet(Somebody1, Somebody2)

When Powell met Zhu Rongji on Thursday they discussed the return of the spy plane.

meet(Powell, Zhu) discuss([Powell, Zhu], return(X, plane))
Capturing semantic roles

- Faisal broke [ARG1 Tom’s chair].
- [ARG1 Monica’s chair] was broken by Lucian.
- [ARG1 Nitin’s chair] broke into pieces when it fell down.
A TreeBanked Sentence

The Supreme Court states "Non Terminal" (130969 ARGs)

"Terminal" (4246 ARGs)
The Same Sentence, PropBanked

The Supreme Court states working Leeway/ARG0 gave Leeway/ARG0.
Core Arguments

- Arg0 = agent
- Arg1 = direct object / theme / patient
- Arg2 = indirect object / benefactive / instrument / attribute / end state
- Arg3 = start point / benefactive / instrument / attribute
- Arg4 = end point
Secondary ArgMs

- LOC - where at?
- DIR - where to?
- MNR - how?
- PRP -why?
- REC - himself, themselves, each other
- PRD -this argument refers to or modifies another
- ADV –others
- TMP - when?
- TPC – topic
- ADV –others
Distributions of Argument Types
How to Use PropBank: Train a Semantic Role Labeling System

• (CONLL 04, CONLL 05 and) Our Goal: Given a list of (3073) target verbs the system should be able to tag the possible nodes with semantic role labels (Ji et al., 2005)
Predicate Features: Lexical

- Head Word, Head Pos of (-2,-1,0,1,2) window of Predicate

- Predicate is a Transitive verb or not

- Predicate Voice (Passive or not)
  - Verb itself: must be in its past particle form
  - Passive Context
    -- Immediately following the verb "be"
    -- Postmodifying a noun in a reduced relative clause, "The building damaged by fire".

- Encoding
  Conjunction feature of Predicate POS_Passive Context
Predicate Sub-Categorization Feature

The Phrase structure rule expanding the predicate’s grandparent

VP->VBD-NP-NP
Predicate Pivot Features

Consider the predicate as a “Pivot”, and its grandparent’s children are defined in relation to it.
Argument Features: Lexical/Syntactic

- Head Word, Head Pos, Phrase Type of (-2,-1,0,1,2) window words, Begin Word, Last Word, Left Sister, Right Sister, Parent of Argument
  (Head of PP replaced by head word of NP inside it)
- Head Pos, Phrase Type of GrandParent
- Suffix1, Suffix2, Suffix3
- Preceding, Succeeding Node’s Label
- Length (Span)
- Level from Leaves
- Beginning Letter of Phrase Type (for generalization)
- Punctuation before/after
- If it includes a Preposition or not, and Prep POS
Intervene Feature: Path

Path: NP \uparrow S \downarrow VP \downarrow VBD \downarrow gave
PartialPath: NP \uparrow S \downarrow
InterPath: S \downarrow VP \downarrow VBD
PredPathArgPathLens: 3_1
CollapsedPath (delete nodes between clauses): NP \uparrow S \downarrow VP \downarrow VBD \downarrow gave
PathSONly (Replace all the non-clause nodes with "*"), NP \uparrow S \downarrow * \downarrow * \downarrow gave
PathSRemain (only keep clause nodes): NP \uparrow S \downarrow gave
PathNGrmas: NP|S|VP, S|VP|VBD, VP|VBD|gave, VBD|gave|*, gave|*|*
PathLen: 4
ArgPhraseType, PathLen: NP, 4
PredHeadWord, PathSRemain: gave, NP \uparrow S \downarrow gave

The Supreme Court gave
Intervene Feature: Position

[Diagram of a syntactic tree]

Directionality: Left    SameClause: true    Dominate Phrase Type: S
Adjacency (Adjacent/not-Adjacent): Adjacent
ArgPhraseType,Adjacency: NP,Adjacent
PositionInClause (Begin, Inside, End): Begin
RelPosition (Distance between Spans): 0
RelPosition,Directionality: 0, Left    RelPosition,Transitive: 0, True
RelPosition,PredHeadPos,PassiveContext: 0, VBD, False
RelPosition,ArgPhraseType: 0, NP
RelPosition,PredPivotv: 0, v_NNS_VBG_NN

The Supreme Court gave
Intervene Features: Pivot

np_v_NP_NP_PP, CUR_v_NP_NP_PP, CUR_gave_NP_NP_PP

- Consider the predicate and candidate argument as “Pivots”, and other constituents are defined in relation to them
Other Features

- PredHeadWord, ArgHeadWord
- PredHeadWord, ArgPhraseType
- ArgPreposition, Transitive
- Frequency of VP, NP, SBAR, CC, ",", ",:", """" in the sentence
- …
- …
Welcome to FrameNet

The Berkeley FrameNet project is creating an on-line lexical resource for English, based on frame semantics and supported by corpus evidence. The aim is to document the range of semantic and syntactic combinatory possibilities (valences) of each word in each of its senses, through computer-assisted annotation of example sentences and automatic tabulation and display of the annotation results. The major product of this work, the FrameNet lexical database, currently contains more than 10,000 lexical units (defined below). more than 6,100 of which

The "Book" v1.3 Available for Download

Our main project document, "FrameNet II: Extended Theory and Practice" gives a basic introduction to frame semantics and offers guidelines for frame semantic annotation, discussing in detail the inventories of grammatical functions and phrase types that we use. The document includes a discussion of the frame development process and a consolidated list of frame-general, extra-thematic frame elements. Our latest version contains some general updates plus a new section
Hmm. Haven’t I heard that word “frame” before?

Yes, it’s intended to be seen as a variation on the word as it’s been used in various branches of the cognitive sciences in recent decades.
“Frames” Traditions

Let’s locate “our” notion of frame within the various traditions in the cognitive sciences that use the words frame or schema (joined, sometimes, with stereotype, script, scenario or idealized cognitive model) which appear to be dealing with essentially the same concept.

These terms are used to denote structured sets of expectations that play a central role in how human beings create or interpret their experiences.
customer

The noun **customer** is typically defined as ‘someone who buys something in a shop or business.’ That includes everyone I know over the age of 5.

Suppose you overhear somebody say

*Sue tends to be rude to customers.*

What situation do you imagine?
chicken (mass noun)

- The noun chicken, as a count noun, is the name of a well-known domestic bird. As a mass noun it is defined as ‘the meat of a chicken or chickens’.
- What’s wrong with the following sentence?
  The fox that lives near our farm likes chicken. (compare: likes chickens)
- The image you might get is of a fox eating fried chicken, holding a knife and a fork, and a napkin, in its paws.
• The products of the lexical construction that yields mass noun uses of **chicken, lamb, duck, turkey**, etc., refer to *meats prepared as part of human cuisine.*

*The wolf that lives near our ranch prefers lamb.*
Invoking and Evoking Frames

- **People invoke** (summon up from their memory) frames, to make sense of their experience, linguistic or otherwise.
  - a cognitive act

- **Words evoke** categories and knowledge structures that shape interpreters’ understanding of a text.
  - a cognitive experience

*Warning: this not a standard use of these words.*
So,

We need to describe words in terms of the “framal” background.
If we don’t understand the frame, we don’t understand
the word,
or why the language needs this word,
or why the speaker chose to use it.
The ideal dictionary should let you

1. Look up a word
2. Get a link to a description of the relevant frame, for each of its meanings, and see the names of the frame’s components
3. See a display of its combinatory affordances, its **valence** possibilities, both semantic and syntactic
4. Find a collection of example sentences illustrating all of its main combinatory patterns
5. Find a list of other words that evoke the same frame
6. Link to other semantically related frames
Frame examples: Risk

Taking_a_risk:
   Protagonist, Action, Harm, Asset
   1. I’m going to risk a swim in the sea.
   2. You’ll risk bankruptcy if you make that investment.
   3. You’re risking your reputation by doing that.
   4. You’re taking a big risk.

Being_at_risk:
   Protagonist, Harm, Asset
   2. Newborns in this hospital run the risk of hypothermia.
   3. We risk our lives every day.
   4. I am at risk of stroke.
Frame examples: Explanation

Communication.explanation:
   Speaker, Addressee, Mystery, Account
1. The coach explained the problem to the team.
2. The coach explained that they hadn't learned the maneuvers.
3. What's your explanation of these facts?
4. The defense lawyer gave an inadequate explanation.

Cognition.explanation:
   Mystery, Account
1. What can explain these facts?
2. A history of unrestricted logging explains the erosion pattern we see here.
Compare: *explain* & *reveal*

In their cognitive senses, as opposed to their meanings as verbs of speaking, the verbs *explain* and *reveal* are near-inverses of each other, where the *Mystery* and the *Account* in the former correspond to *Evidence* and *Conclusion* in the latter.

1. A history of unrestrained logging *explains* the erosion pattern we see here. *(explains, accounts for)*
2. The erosion pattern we see here *reveals* a history of unrestrained logging. *(reveals, shows, suggests)*
## Reading the Pictures

- The boxes refer to five-part scenarios consisting of an initial state, a transition, an intermediary state, another transition, and a final state.
- The writing under the pictures abbreviates particular role names and gives verbs that evoke instances of the scenario.
- The bold borders indicate a profiling of some portion of the event.

<table>
<thead>
<tr>
<th>state</th>
<th>transition</th>
<th>state</th>
<th>transition</th>
<th>state</th>
</tr>
</thead>
</table>

He returned to Hong Kong. He returned Tuesday evening after a week’s trip to Australia. He returned to his home for a few days.

*The verb RETURN profiles the time of arrival, but it evokes the entire frame; other information in the sentences can fill in some of the details of the larger scenario.*
I returned your books this morning.

I returned to your desk the books that I had borrowed last week.

After the earthquake we replaced all the books on the shelf.
frame elements

• Participants and Sub-events.
  • Avenger: the one who enacts revenge
  • Offender: the original offender
  • Injured_party: the offender’s victim
  • Injury: the offender’s act
  • Punishment: the avenger’s act
Components of linguistic form for expressing the FEs (defining *valence*).

- **Subject**
- **Direct Object**
- **Prepositional marking** *(by, for, with, on, at, against)*
- **Subordinate clause marking** *(for DOING, by DOING)*