Recent Advances in Dependency Parsing

Tutorial, EACL, April 27th, 2014

Ryan McDonald\textsuperscript{1} Joakim Nivre\textsuperscript{2}

\textsuperscript{1}Google Inc., USA/UK
E-mail: ryanmcd@google.com

\textsuperscript{2}Uppsala University, Sweden
E-mail: joakim.nivre@lingfil.uu.se
Introduction

- **Pre-2008**
  - What we will briefly cover in first quarter of tutorial
  - Textbook: *Dependency Parsing* [Kübler et al. 2009]
  - ESSLLI 2007:  
    http://www.ryanmcd.com/courses/esslli2007/
  - ACL 2006:  
    http://stp.lingfil.uu.se/~nivre/docs/ACLslides.pdf

- **Post-2008**
  - What we will mainly cover today
  - NAACL 2010:  
    http://naaclhlt2010.isi.edu/tutorials/t7.html
Overview of the Tutorial

- **Introduction to Dependency Parsing** (Joakim)
  - Formulation, definitions, evaluation, etc.
  - Graph-based and transition-based parsing
  - Contrastive error analysis
- Graph-based parsing post-2008 (Ryan)
- Transition-based parsing post-2008 (Joakim)
- Summary and final thoughts (Ryan)
Introduction: Outline

- Dependency syntax:
  - Basic concepts
  - Terminology and notation
  - Dependency graphs
  - Data-driven dependency parsing

- Paradigms:
  - Graph-based parsing
  - Transition-based parsing
  - Alternatives (brief)

- Contrastive error analysis [McDonald and Nivre 2007]
The basic idea:

- Syntactic structure consists of lexical items, linked by binary asymmetric relations called dependencies.

In the words of Lucien Tesnière [Tesnière 1959]:

- La phrase est un ensemble organisé dont les éléments constitutants sont les mots. [1.2] Tout mot qui fait partie d’une phrase cesse par lui-même d’être isolé comme dans le dictionnaire. Entre lui et ses voisins, l’esprit aperçoit des connexions, dont l’ensemble forme la charpente de la phrase. [1.3] Les connexions structurales établissent entre les mots des rapports de dépendance. Chaque connexion unit en principe un terme supérieur à un terme inférieur. [2.1] Le terme supérieur reçoit le nom de régissant. Le terme inférieur reçoit le nom de subordonné. Ainsi dans la phrase Alfred parle […], parle est le régissant et Alfred le subordonné. [2.2]
Dependency Syntax

- The basic idea:
  - Syntactic structure consists of lexical items, linked by binary asymmetric relations called dependencies.
- In the words of Lucien Tesnière [Tesnière 1959]:
  - The sentence is an organized whole, the constituent elements of which are words. [1.2] Every word that belongs to a sentence ceases by itself to be isolated as in the dictionary. Between the word and its neighbors, the mind perceives connections, the totality of which forms the structure of the sentence. [1.3] The structural connections establish dependency relations between the words. Each connection in principle unites a superior term and an inferior term. [2.1] The superior term receives the name governor. The inferior term receives the name subordinate. Thus, in the sentence Alfred parle [. . .], parle is the governor and Alfred the subordinate. [2.2]
Economic news had little effect on financial markets.
Economic news had little effect on financial markets.
Economic news had little effect on financial markets.
Economic news had little effect on financial markets.
Economic news had little effect on financial markets.
Economic news had little effect on financial markets.
Economic news had little effect on financial markets.
## Terminology

<table>
<thead>
<tr>
<th>Superior</th>
<th>Inferior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Head</td>
<td>Dependent</td>
</tr>
<tr>
<td>Governor</td>
<td>Modifier</td>
</tr>
<tr>
<td>Regent</td>
<td>Subordinate</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>
## Terminology

<table>
<thead>
<tr>
<th>Superior</th>
<th>Inferior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Head</td>
<td>Dependent</td>
</tr>
<tr>
<td>Governor</td>
<td>Modifier</td>
</tr>
<tr>
<td>Regent</td>
<td>Subordinate</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Economic news had little effect on financial markets.
Comparison

- Dependency structures explicitly represent
  - head-dependent relations (directed arcs),
  - functional categories (arc labels),
  - possibly some structural categories (parts-of-speech).

- Phrase structures explicitly represent
  - phrases (nonterminal nodes),
  - structural categories (nonterminal labels),
  - possibly some functional categories (grammatical functions).

- Hybrid representations may combine all elements.
Some Theoretical Frameworks

- **Word Grammar (WG)** [Hudson 1984, Hudson 1990, Hudson 2007]
- **Functional Generative Description (FGD)** [Sgall et al. 1986]
- **Dependency Unification Grammar (DUG)** [Hellwig 1986, Hellwig 2003]
- **Meaning-Text Theory (MTT)** [Mel’čuk 1988, Milićević 2006]
- **Functional Dependency Grammar (FDG)** [Tapanainen and Järvinen 1997, Järvinen and Tapanainen 1998]
Some Theoretical Issues

- Dependency structure sufficient as well as necessary?
- Mono-stratal or multi-stratal syntactic representations?
- What is the nature of lexical elements (nodes)?
  - Morphemes?
  - Word forms?
  - Multi-word units?
- What is the nature of dependency types (arc labels)?
  - Grammatical functions?
  - Semantic roles?
- What are the criteria for identifying heads and dependents?
- What are the formal properties of dependency structures?
Some Theoretical Issues

- Dependency structure **sufficient** as well as necessary?
- **Mono-stratal** or multi-stratal syntactic representations?
- What is the nature of lexical elements (nodes)?
  - Morphemes?
  - **Word forms**?
  - Multi-word units?
- What is the nature of dependency types (arc labels)?
  - **Grammatical functions**?
  - Semantic roles?
- What are the criteria for identifying heads and dependents?
- What are the formal properties of dependency structures?

Dependency Graphs

- A dependency structure can be defined as a directed graph $G$, consisting of
  - a set $V$ of nodes (vertices),
  - a set $A$ of arcs (directed edges),
  - a linear precedence order $<$ on $V$ (word order).
- Labeled graphs:
  - Nodes in $V$ are labeled with word forms (and annotation).
  - Arcs in $A$ are labeled with dependency types:
    - $L = \{l_1, \ldots, l_{|L|}\}$ is the set of permissible arc labels.
    - Every arc in $A$ is a triple $(i, j, k)$, representing a dependency from $w_i$ to $w_j$ with label $l_k$. 
Dependency Graph Notation

- For a dependency graph $G = (V, A)$
- With label set $L = \{l_1, \ldots, l_{|L|}\}$
  - $i \rightarrow j \equiv \exists k: (i, j, k) \in A$
  - $i \leftrightarrow j \equiv i \rightarrow j \vee j \rightarrow i$
  - $i \rightarrow^* j \equiv i = j \vee \exists i': i \rightarrow i', i' \rightarrow^* j$
  - $i \leftrightarrow^* j \equiv i = j \vee \exists i': i \leftrightarrow i', i' \leftrightarrow^* j$
Formal Conditions on Dependency Graphs

- $G$ is (weakly) connected:
  - If $i, j \in V$, $i \leftrightarrow^* j$.
- $G$ is acyclic:
  - If $i \rightarrow j$, then not $j \rightarrow^* i$.
- $G$ obeys the single-head constraint:
  - If $i \rightarrow j$, then not $i' \rightarrow j$, for any $i' \neq i$.
- $G$ is projective:
  - If $i \rightarrow j$, then $i \rightarrow^* i'$, for any $i'$ such that $i < i' < j$ or $j < i' < i$. 
Connectedness, Acyclicity and Single-Head

▶ Intuitions:
  ▶ Syntactic structure is complete (Connectedness).
  ▶ Syntactic structure is hierarchical (Acyclicity).
  ▶ Every word has at most one syntactic head (Single-Head).

▶ Connectedness can be enforced by adding a special root node.
Connectedness, Acyclicity and Single-Head

- **Intuitions:**
  - Syntactic structure is complete (**Connectedness**).
  - Syntactic structure is hierarchical (**Acyclicity**).
  - Every word has at most one syntactic head (**Single-Head**).

- Connectedness can be enforced by adding a special root node.

```
ROOT Economic
  adj news
  noun had
  verb little
  adj effect
  noun on
  prep financial
  adj markets
  .
```
Most theoretical frameworks do not assume projectivity.

Non-projective structures are needed to account for
- long-distance dependencies,
- free word order.

What did economic news have little effect on?
Dependency Parsing

- **The problem:**
  - Input: Sentence $x = w_0, w_1, \ldots, w_n$ with $w_0 = \text{ROOT}$
  - Output: Dependency graph $G = (V, A)$ for $x$ where:
    - $V = \{0, 1, \ldots, n\}$ is the vertex set,
    - $A$ is the arc set, i.e., $(i, j, k) \in A$ represents a dependency from $w_i$ to $w_j$ with label $l_k \in L$

- **Two main approaches:**
  - Grammar-based parsing
    - Context-free dependency grammar
    - Lexicalized context-free grammars
    - Constraint dependency grammar
  - Data-driven parsing
    - Graph-based models
    - Transition-based models
    - Easy-first parsing
    - Hybrids: grammar + data-driven, ensembles, etc.
Data-Driven Dependency Parsing

- Need to define a function $f : \mathcal{X} \rightarrow \mathcal{G}$
  - From sentences $x \in \mathcal{X}$ to valid dependency graphs $G \in \mathcal{G}$
- Most common approach is to learn from training data $\mathcal{T}$,
  - where $\mathcal{T} = \{(x_1, G_1), (x_2, G_2), \ldots, (x_n, G_n)\}$,
  - and $(x_i, G_i)$ are labeled sentence and dependency graph pairs that make up the treebank.
- **Supervised learning**: Fully annotated training examples
- Semi-supervised learning: Annotated data plus constraints and features drawn from unlabeled resources
- Weakly-supervised learning: Constraints drawn from ontologies, structural and lexical resources
- Unsupervised learning: Learning only from unlabeled data
Evaluation Metrics

- **Standard setup:**
  - Test set $\mathcal{E} = \{(x_1, G_1), (x_2, G_2), \ldots, (x_n, G_n)\}$
  - Parser predictions $\mathcal{P} = \{(x_1, G'_1), (x_2, G'_2), \ldots, (x_n, G'_n)\}$

- **Evaluation on the word (arc) level:**
  - Labeled attachment score (LAS) = head and label
  - Unlabeled attachment score (UAS) = head
  - Label accuracy (LA) = label

- **Evaluation on the sentence (graph) level:**
  - Exact match (labeled or unlabeled) = complete graph

- **NB:** Evaluation metrics may or may not include punctuation
Graph-Based Parsing (Pre-2008)

- **Basic idea:**
  - Define a space of candidate dependency graphs for a sentence.
  - **Learning:** Induce a model for scoring an entire dependency graph for a sentence.
  - **Parsing:** Find the highest-scoring dependency graph, given the induced model.

- **Characteristics:**
  - Global training of a model for optimal dependency graphs
  - Exhaustive search/inference
Graph-Based Parsing

For input sentence $x$, define a graph $G_x = (V_x, A_x)$, where

$V_x = \{0, 1, \ldots, n\}$

$A_x = \{(i, j, k) \mid i, j \in V \text{ and } l_k \in L\}$

Key observation:

- Valid dependency trees for $x = \text{directed spanning trees of } G_x$

Score of dependency tree $T$ factors by subgraphs $G_1, \ldots, G_m$:

$s(T) = \sum_{i=1}^{m} s(G_i)$

Learning:

- Scoring function $s(G_i)$ for subgraphs $G_i \in G$

Inference:

- Search for maximum spanning tree $T^*$ of $G_x$ given $s(G_i)$
Parameterizing Graph-Based Parsing

- First-order (arc-factored) model:
  - \[ s(T = (V, A)) = \sum_{(i,j,k) \in A} s(i, j, k) \]
  - Exact inference in \( O(n^2) \) time for non-projective trees using the Chu-Liu-Edmonds algorithm [McDonald et al. 2005b]
Parameterizing Graph-Based Parsing

- Higher-order models [McDonald and Pereira 2006, Carreras 2007]:
  - Subgraphs $G_i$ involving a (small) number of arcs
  - Intractable in non-projective case [McDonald and Satta 2007]
  - Exact inference in $O(n^3)$ time for projective trees with second-order model using Eisner’s algorithm [Eisner 1996]

- Efficient parsing requires that scores factor by small subgraphs
Learning Graph-Based Models

Typical scoring function:

\[ s(G_i) = w \cdot f(G_i) \]

where

\[ f(G_i) = \text{high-dimensional feature vector over subgraphs} \]
\[ w = \text{weight vector} \quad [w_j = \text{weight of feature } f_j(G_i)] \]

Structured learning [McDonald et al. 2005a]:

Learn weights that maximize the score of the correct dependency tree for every sentence in the training set

Learning is **global** (trees), but features are **local** (subgraphs)
Transition-Based Parsing (Pre-2008)

- Basic idea:
  - Define a transition system (state machine) for mapping a sentence to its dependency graph.
  - Learning: Induce a model for predicting the next state transition, given the transition history.
  - Parsing: Construct the optimal transition sequence, given the induced model.

- Characteristics:
  - Local training of a model for optimal transitions
  - Greedy search/inference
Transition-Based Parsing (Pre-2008)

- A transition system for dependency parsing defines
  - a set $C$ of parser configurations
  - a set $T$ of transitions, each a function $t : C \rightarrow C$
  - initial configuration and terminal configurations for sentence $x$

- Key idea:
  - Valid dependency trees for $S$ defined by terminating transition sequences $C_{0,m} = t_1(c_0), \ldots, t_m(c_{m-1})$

- Score of $C_{0,m}$ factors by config-transition pairs $(c_{i-1}, t_i)$:
  - $s(C_{0,m}) = \sum_{i=1}^{m} s(c_{i-1}, t_i)$

- Learning:
  - Scoring function $s(c_{i-1}, t_i)$ for $t_i(c_{i-1}) \in C_{0,m}$

- Inference:
  - Search for highest scoring sequence $C_{0,m}^*$ given $s(c_{i-1}, t_i)$
Example: Arc-Eager Projective Parsing

Configuration: \((S, B, A)\) \([S = \text{Stack}, B = \text{Buffer}, A = \text{Arcs}]\)

Initial: \(([ ], [0, 1, \ldots, n], \{ \})\)

Terminal: \((S, [ ], A)\)

Shift: \((S, i|B, A)\) \(\Rightarrow (S|i, B, A)\)

Reduce: \((S|i, B, A)\) \(\Rightarrow (S, B, A)\)

\h(i, A)

Right-Arc\(k\): \((S|i, j|B, A)\) \(\Rightarrow (S|i|j, B, A \cup \{(i, j, k)\})\)

Left-Arc\(k\): \((S|i, j|B, A)\) \(\Rightarrow (S, j|B, A \cup \{(j, i, k)\})\)

\neg h(i, A) \land i \neq 0
Inference for Transition-Based Parsing

- Exact inference intractable for standard transition systems
- Standard approach:
  - Greedy (pseudo-deterministic) inference
    [Yamada and Matsumoto 2003, Nivre et al. 2004]
  - Complexity given by upper bound on transition sequence length
- Transition systems:
  - Projective $O(n)$ [Yamada and Matsumoto 2003, Nivre 2003]
  - Limited non-projective $O(n)$ [Attardi 2006, Nivre 2007]
  - Unrestricted non-projective $O(n^2)$ [Covington 2001, Nivre 2008]
- Efficient parsing requires approximate inference
Learning for Transition-Based Parsing

- Typical scoring function:
  - \( s(c, t) = \mathbf{w} \cdot \mathbf{f}(c, t) \)
  
  where
  - \( \mathbf{f}(c, t) = \) feature vector over configuration \( c \) and transition \( t \)
  - \( \mathbf{w} = \) weight vector \( [w_j = \) weight of feature \( \mathbf{f}_j(c, t) ] \)

- Simple classification problem:
  - Learn weights that maximize the score of the correct transition out of every configuration in the training set
  - Configurations represent derivation history, including partially built dependency tree

- Learning is local, but features are global
CoNLL 2006

- CoNLL 2006: Shared Task on Dependency Parsing
  - Evaluation of 13 different languages
- Top 2 systems statistically identical: One graph-based (MSTParser) and the other transition-based (MaltParser)
- Question: do the systems learn the same things?
## MSTParser and MaltParser

<table>
<thead>
<tr>
<th>Language</th>
<th>MSTParser</th>
<th>MaltParser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>66.91</td>
<td>66.71</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>87.57</td>
<td>87.41</td>
</tr>
<tr>
<td>Chinese</td>
<td>85.90</td>
<td>86.92</td>
</tr>
<tr>
<td>Czech</td>
<td>80.18</td>
<td>78.42</td>
</tr>
<tr>
<td>Danish</td>
<td>84.79</td>
<td>84.77</td>
</tr>
<tr>
<td>Dutch</td>
<td>79.19</td>
<td>78.59</td>
</tr>
<tr>
<td>German</td>
<td>87.34</td>
<td>85.82</td>
</tr>
<tr>
<td>Japanese</td>
<td>90.71</td>
<td>91.65</td>
</tr>
<tr>
<td>Portuguese</td>
<td>86.82</td>
<td>87.60</td>
</tr>
<tr>
<td>Slovene</td>
<td>73.44</td>
<td>70.30</td>
</tr>
<tr>
<td>Spanish</td>
<td>82.25</td>
<td>81.29</td>
</tr>
<tr>
<td>Swedish</td>
<td>82.55</td>
<td>84.58</td>
</tr>
<tr>
<td>Turkish</td>
<td>63.19</td>
<td>65.68</td>
</tr>
<tr>
<td>Overall</td>
<td>80.83</td>
<td>80.75</td>
</tr>
</tbody>
</table>
Comparing the Models

- **Inference**:
  - Exhaustive (**MSTParser**)
  - Greedy (**MaltParser**)

- **Training**:
  - Global structure learning (**MSTParser**)
  - Local decision learning (**MaltParser**)

- **Features**:
  - Local features (**MSTParser**)
  - Rich decision history (**MaltParser**)

- **Fundamental trade-off**:
  - Global learning and inference vs. rich feature space
Error Analysis [McDonald and Nivre 2007]

- **Aim:**
  - Relate parsing errors to linguistic and structural properties of the input and predicted/gold standard dependency graphs

- **Three types of factors:**
  - Length factors: sentence length, dependency length
  - Graph factors: tree depth, branching factor, non-projectivity
  - Linguistic factors: part of speech, dependency type

- **Statistics:**
  - Labeled accuracy, precision and recall
  - Computed over the test sets for all 13 languages
MaltParser is more accurate than MSTParser for short sentences (1–10 words) but its performance degrades more with increasing sentence length.
MaltParser is more precise than MSTParser for short dependencies (1–3 words) but its performance degrades drastically with increasing dependency length (> 10 words).

MSTParser has more or less constant precision for dependencies longer than 3 words.

Recall is very similar across systems.
MSTParser is much more precise than MaltParser for dependents of the root and has roughly constant precision for depth > 1, while MaltParser’s precision improves with increasing depth (up to 7 arcs).

Recall is very similar across systems.
Degrees of Non-Projectivity

- Degree of a dependency arc \((i, j, k)\) = The number of words in the span \(\min(i, j), \ldots, \max(i, j)\) that are not descendants of \(i\) and have their head outside the span.

- MaltParser has slightly higher precision, and MSTParser slightly higher recall, for non-projective arcs (degree > 0).

- No system predicts arcs with a higher degree than 2.
Part of Speech

- MSTParser is more accurate for verbs, adjectives, adverbs, adpositions, and conjunctions.
- MaltParser is more accurate for nouns and pronouns.
Dependency Type: Root, Subject, Object

- MSTParser has higher precision (and recall) for roots.
- MSTParser has higher recall (and precision) for subjects.
Discussion

- Many of the results are indicative of the fundamental trade-off: global learning/inference versus rich features.
- Global inference improves decisions for long sentences and those near the top of graphs.
- Rich features improve decisions for short sentences and those near the leaves of the graphs.
- Dependency parsing post-2008:
  - How do we use this to improve parser performance?
Voting and Stacking

- Early improvements were based on system combination
- Voting:
  - Let parsers vote for heads [Zeman and Žabokrtský 2005]
  - Use MST algorithm for tree constraint [Sagae and Lavie 2006]
- Stacking:
  - Use the output of one parser as features for the other [Nivre and McDonald 2008, Torres Martins et al. 2008]
- Focus today:
  - Recent work evolving the approaches themselves
  - Richer feature representations in graph-based parsing
  - Improved learning and inference in transition-based parsing
Summary

- Dependency syntax – basic concepts
- Dependency parsing – graph-based and transition-based
- Empirical trade-offs present way forward
References and Further Reading


Methods in Natural Language Processing and the Conference on Computational Natural Language Learning (EMNLP-CoNLL).


- Wolfgang Menzel and Ingo Schröder. 1998. Decision procedures for dependency parsing using graded constraints. In Sylvain Kahane and Alain Polguère, editors,
References and Further Reading


