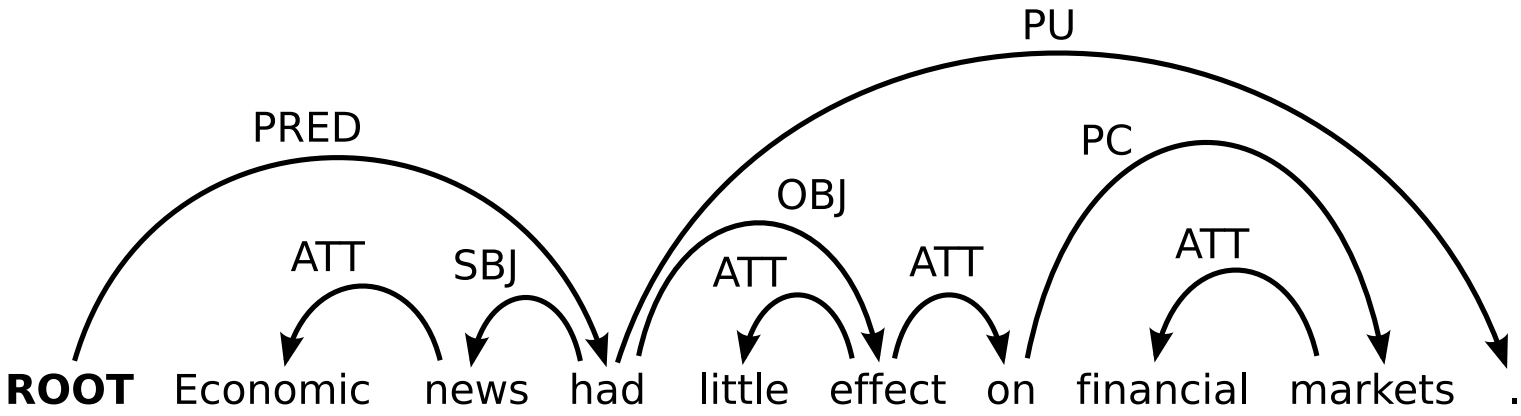
Inductive Dependency Parsing of Natural Language Text

Joakim Nivre

Uppsala University
Department of Linguistics and Philology
joakim.nivre@lingfil.uu.se

Introduction

- Syntactic parsing of natural language:
  - Who does what to whom?
- Dependency-based syntactic representations
  - give a transparent encoding of predicate-argument structure
  - can be learned using (simple) data-driven models
  - can be parsed very efficiently



Outline

- Text parsing
  - Parsing natural language text – an ill-defined problem?
  - Evaluation criteria for text parsing
- Dependency parsing
  - Theoretical assumptions
  - Formal representations
- Transition-based parsing
  - Transition systems for parsing
  - Machine learning from treebanks
- Empirical results:
  - Multilingual parser evaluation
  - Parsing the SynTagRus treebank of Russian

Text Parsing

Two Notions of Parsing

- Grammar parsing:
  - Given a grammar $\mathcal{G}$ and an input string $S \in \Sigma^*$, derive some or all of the analyses assigned to S by $\mathcal{G}$
  - Well-defined abstract problem – non-empirical
- Text parsing:
  - Given a text $\mathcal{T} = (S_1, \ldots, S_n)$ in the language L, derive the correct analysis for every sentence $S_i \in \mathcal{T}$
  - Empirical approximation problem – ill-defined?

Evaluation Criteria for Text Parsing

- Robustness:
  - At least one analysis per sentence
- Disambiguation:
  - At most one analysis per sentence
- Accuracy:
  - The correct analysis for every sentence
- Efficiency:
  - Parsing in linear time and space

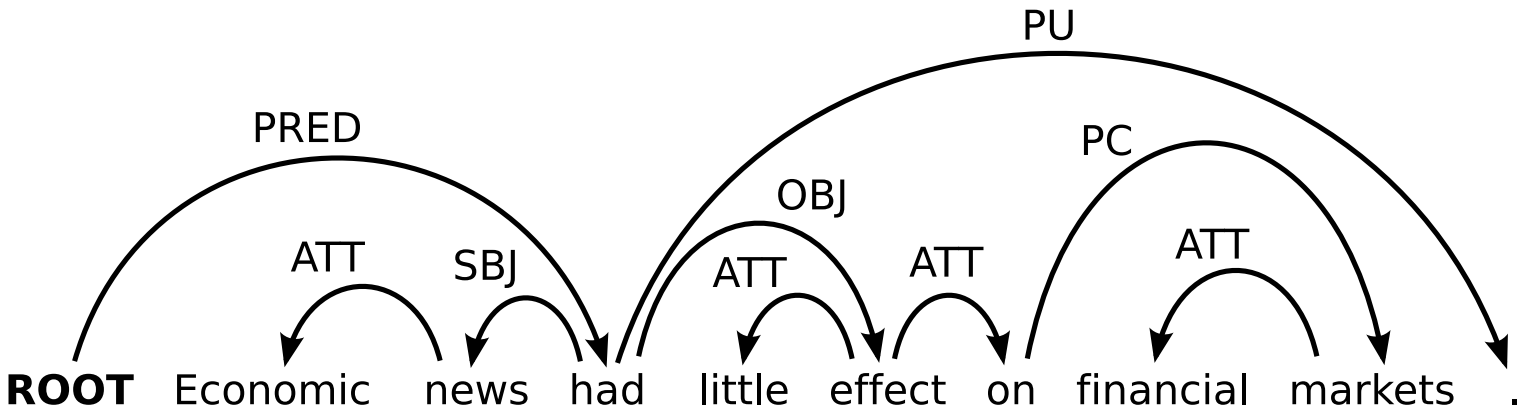
Two Strategies for Text Parsing

- Grammar-driven text parsing:
  - Grammar-based approximation: $L(\mathcal{G}) \approx L$
  - Traditional problems: robustness, disambiguation, efficiency?
- Data-driven text parsing:
  - Inductive approximation: $I(\mathcal{T}) \approx L$
  - Traditional problems: accuracy, efficiency?
- Orthogonal approaches – not incompatible

Dependency Parsing

Dependency Grammar

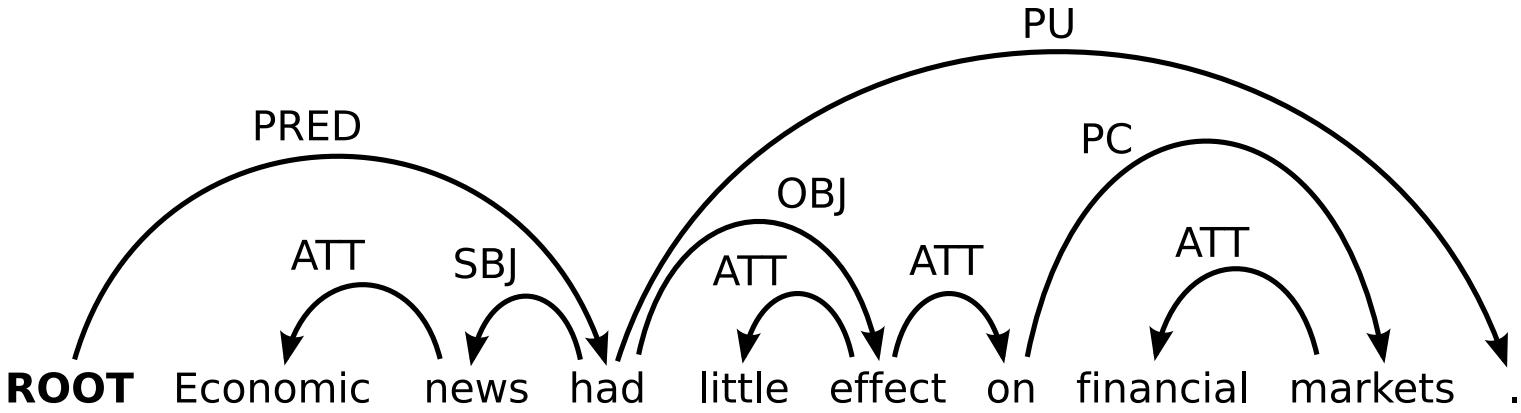
- The basic idea:
  - Syntactic structure consists of lexical items, linked by binary asymmetric relations called dependencies
- Many different theoretical frameworks



Dependency Trees

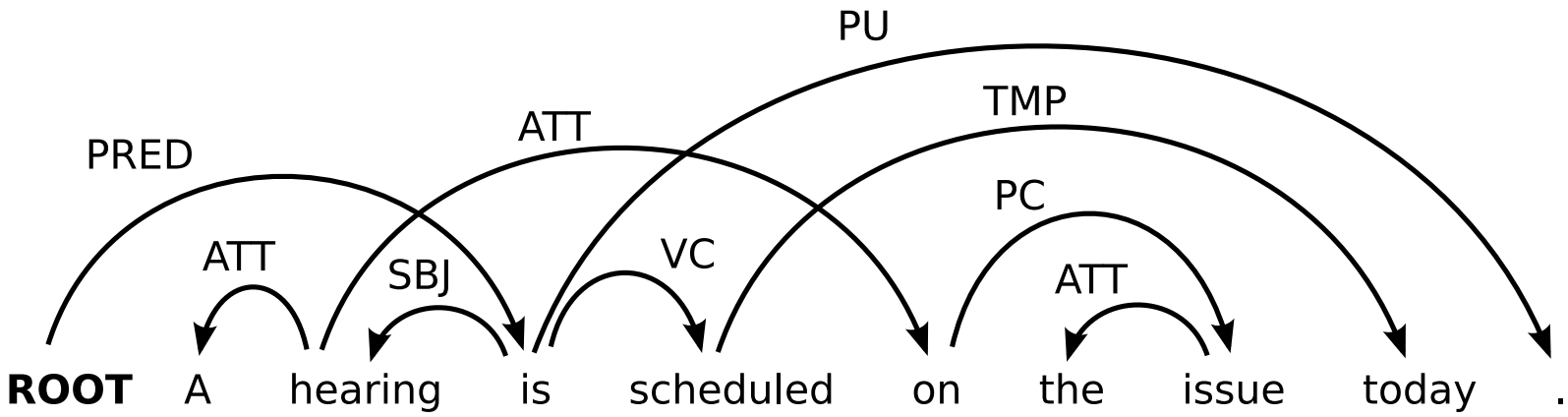
- A dependency structure for a sentence $S = w_1, \ldots, w_n$ is a labeled directed tree $T = (V, A)$, consisting of
  - a set V of nodes, labeled with words w_i (including ROOT)
  - a set A of arcs (w_i, w_j, l), labeled with dependency types $l \in L$

  with the node labeled ROOT as the unique root



Projectivity

- A dependency tree T is projective iff every word has a contiguous projection in S
  - Projection of w = the set of words dominated by w
- Most theoretical frameworks do not assume projectivity
- Non-projective structures needed to account for
  - long-distance dependencies
  - free word order



Inductive Dependency Parsing

- Dependency parsing:
  - Input: Sentence $S = w_1, \ldots, w_n$
  - Output: Dependency tree T
- Inductive dependency parsing:
  - Parsing model learned from data (treebank)
  - May or may not involve a formal grammar
- Two main approaches:
  - Transition-based [Yamada and Matsumoto 2003, Nivre et al. 2004]
  - Graph-based [Eisner 1996, McDonald et al. 2005]

Inductive Dependency Parsing

- Dependency parsing:
  - Input: Sentence $S = w_1, \ldots, w_n$
  - Output: Dependency tree T
- Inductive dependency parsing:
  - Parsing model learned from data (treebank)
  - May or may not involve a formal grammar
- Two main approaches:
  - Transition-based [Yamada and Matsumoto 2003, Nivre et al. 2004]
  - Graph-based [Eisner 1996, McDonald et al. 2005]

Transition-Based Parsing

Overview of the Approach

- The basic idea:
  - Define a transition system for dependency parsing
  - Train a classifier for predicting the next transition
  - Use the classifier to do parsing as greedy, deterministic search
- Advantages:
  - Efficient parsing (linear time complexity)
  - Robust disambiguation (discriminative classifiers)

Transition System: Configurations

- A parser configuration is a triple $c = (S, Q, A)$, where
  - S = a stack $[\ldots, w_i]_S$ of partially processed nodes,
  - Q = a queue $[w_j, \ldots]_Q$ of remaining input nodes,
  - A = a set of labeled arcs (w_i, w_j, l).
- Initialization:

  $([w_0]_S, [w_1, \ldots, w_n]_Q, \{\})$

  NB: w_0 = ROOT
- Termination:

  $([w_0]_S, []_Q, A)$

Transition System: Transitions

- Left-Arc(l)

$$\frac{([\ldots, w_i, w_j]_S, \quad Q, \quad A)}{([\ldots, w_j]_S, \quad Q, \quad A \cup \{(w_j, w_i, l)\})} \quad [i \neq 0]$$

- Right-Arc(l)

$$\frac{([\ldots, w_i, w_j]_S, \quad Q, \quad A)}{([\ldots, w_i]_S, \quad Q, \quad A \cup \{(w_i, w_j, l)\})}$$

- Shift

$$\frac{([\ldots]_S, \quad [w_i, \ldots]_Q, \quad A)}{([\ldots, w_i]_S, \quad [\ldots]_Q, \quad A)}$$

Deterministic Parsing

- Given an oracle o that correctly predicts the next transition $o(c)$, parsing is deterministic:

Parse$(w_1, \ldots, w_n)$
1 $c \leftarrow ([w_0]_S, [w_1, \ldots, w_n]_Q, \{\})$
2 **while** $Q_c \neq [\,]$ **or** $|S_c| > 1$
3 $t \leftarrow o(c)$
4 $c \leftarrow t(c)$
5 **return** $G = (\{w_0, w_1, \ldots, w_n\}, A_c)$

Parsing Example

$[\text{ROOT}]_S$ $[\text{Economic}, \text{news}, \text{had}, \text{little}, \text{effect}, \text{on}, \text{financial}, \text{markets}, .]_B$

Parsing Example

$[\text{ROOT}, \text{Economic}]_S \quad [\text{news}, \text{had}, \text{little}, \text{effect}, \text{on}, \text{financial}, \text{markets}, .]_B$

Parsing Example

$[\text{ROOT}, \text{Economic}, \text{news}]_S \quad [\text{had}, \text{little}, \text{effect}, \text{on}, \text{financial}, \text{markets}, .]_B$

Parsing Example

$[\text{ROOT}, \text{news}]_S \quad [\text{had}, \text{little}, \text{effect}, \text{on}, \text{financial}, \text{markets}, .]_B$

$\text{Economic} \xleftarrow{\text{ATT}} \text{news}$

Parsing Example

$[\text{ROOT}, \text{news}, \text{had}]_S \quad [\text{little}, \text{effect}, \text{on}, \text{financial}, \text{markets}, .]_B$

$\text{Economic} \xleftarrow{\text{ATT}} \text{news}$

Parsing Example

$[\text{ROOT}, \text{had}]_S \quad [\text{little}, \text{effect}, \text{on}, \text{financial}, \text{markets}, .]_B$

$$\text{Economic} \xleftarrow{\text{ATT}} \text{news}$$
$$\text{news} \xleftarrow{\text{SBJ}} \text{had}$$

Parsing Example

$[\text{ROOT}, \text{had}, \text{little}]_S \quad [\text{effect}, \text{on}, \text{financial}, \text{markets}, .]_B$

$$\text{Economic} \xleftarrow{\text{ATT}} \text{news}$$
$$\text{news} \xleftarrow{\text{SBJ}} \text{had}$$

Parsing Example

$[\text{ROOT}, \text{had}, \text{little}, \text{effect}]_S \quad [\text{on}, \text{financial}, \text{markets}, .]_B$

$$\text{Economic} \xleftarrow{\text{ATT}} \text{news}$$
$$\text{news} \xleftarrow{\text{SBJ}} \text{had}$$

Parsing Example

$[\text{ROOT}, \text{had}, \text{effect}]_S \quad [\text{on}, \text{financial}, \text{markets}, .]_B$

$$\begin{aligned} \text{Economic} &\xleftarrow{\text{ATT}} \text{news} \\ \text{news} &\xleftarrow{\text{SBJ}} \text{had} \\ \text{little} &\xleftarrow{\text{ATT}} \text{effect} \end{aligned}$$

Parsing Example

$[\text{ROOT}, \text{had}, \text{effect}, \text{on}]_S \quad [\text{financial}, \text{markets}, .]_B$

$$\text{Economic} \xleftarrow{\text{ATT}} \text{news}$$
$$\text{news} \xleftarrow{\text{SBJ}} \text{had}$$
$$\text{little} \xleftarrow{\text{ATT}} \text{effect}$$

Parsing Example

$[\text{ROOT}, \text{had}, \text{effect}, \text{on}, \text{financial}]_S \quad [\text{markets}, .]_B$

$$\text{Economic} \xleftarrow{\text{ATT}} \text{news}$$
$$\text{news} \xleftarrow{\text{SBJ}} \text{had}$$
$$\text{little} \xleftarrow{\text{ATT}} \text{effect}$$

Parsing Example

$[\text{ROOT}, \text{had}, \text{effect}, \text{on}, \text{financial}, \text{markets}]_S \quad [.]_B$

$$\text{Economic} \xleftarrow{\text{ATT}} \text{news}$$
$$\text{news} \xleftarrow{\text{SBJ}} \text{had}$$
$$\text{little} \xleftarrow{\text{ATT}} \text{effect}$$

Parsing Example

$[\text{ROOT}, \text{had}, \text{effect}, \text{on}, \text{markets}]_S \quad [.]_B$

$$\begin{aligned}
\text{Economic} &\xleftarrow{\text{ATT}} \text{news} \\
\text{news} &\xleftarrow{\text{SBJ}} \text{had} \\
\text{little} &\xleftarrow{\text{ATT}} \text{effect} \\
\text{financial} &\xleftarrow{\text{ATT}} \text{markets}
\end{aligned}$$

Parsing Example

$[\text{ROOT}, \text{had}, \text{effect}, \text{on}]_S \quad [.]_B$

$$\begin{aligned}
\text{Economic} &\xleftarrow{\text{ATT}} \text{news} \\
\text{news} &\xleftarrow{\text{SBJ}} \text{had} \\
\text{little} &\xleftarrow{\text{ATT}} \text{effect} \\
\text{financial} &\xleftarrow{\text{ATT}} \text{markets} \\
\text{on} &\xrightarrow{\text{PC}} \text{markets}
\end{aligned}$$

Parsing Example

$[\text{ROOT}, \text{had}, \text{effect}]_S \quad [.]_B$

$$\begin{aligned}
\text{Economic} &\xleftarrow{\text{ATT}} \text{news} \\
\text{news} &\xleftarrow{\text{SBJ}} \text{had} \\
\text{little} &\xleftarrow{\text{ATT}} \text{effect} \\
\text{financial} &\xleftarrow{\text{ATT}} \text{markets} \\
\text{on} &\xrightarrow{\text{PC}} \text{markets} \\
\text{effect} &\xrightarrow{\text{ATT}} \text{on}
\end{aligned}$$

Parsing Example

$[\text{ROOT}, \text{had}]_S \quad [.]_B$

$$\begin{aligned}
\text{Economic} &\xleftarrow{\text{ATT}} \text{news} \\
\text{news} &\xleftarrow{\text{SBJ}} \text{had} \\
\text{little} &\xleftarrow{\text{ATT}} \text{effect} \\
\text{financial} &\xleftarrow{\text{ATT}} \text{markets} \\
\text{on} &\xrightarrow{\text{PC}} \text{markets} \\
\text{effect} &\xrightarrow{\text{ATT}} \text{on} \\
\text{had} &\xrightarrow{\text{OBJ}} \text{effect}
\end{aligned}$$

Parsing Example

$[\text{ROOT}, \text{had}, .]_S \quad [\ ]_B$

$$\begin{aligned}
\text{Economic} &\xleftarrow{\text{ATT}} \text{news} \\
\text{news} &\xleftarrow{\text{SBJ}} \text{had} \\
\text{little} &\xleftarrow{\text{ATT}} \text{effect} \\
\text{financial} &\xleftarrow{\text{ATT}} \text{markets} \\
\text{on} &\xrightarrow{\text{PC}} \text{markets} \\
\text{effect} &\xrightarrow{\text{ATT}} \text{on} \\
\text{had} &\xrightarrow{\text{OBJ}} \text{effect}
\end{aligned}$$

Parsing Example

$[\text{ROOT}, \text{had}]_S \quad [\ ]_B$

$$\begin{aligned}
\text{Economic} &\xleftarrow{\text{ATT}} \text{news} \\
\text{news} &\xleftarrow{\text{SBJ}} \text{had} \\
\text{little} &\xleftarrow{\text{ATT}} \text{effect} \\
\text{financial} &\xleftarrow{\text{ATT}} \text{markets} \\
\text{on} &\xrightarrow{\text{PC}} \text{markets} \\
\text{effect} &\xrightarrow{\text{ATT}} \text{on} \\
\text{had} &\xrightarrow{\text{OBJ}} \text{effect} \\
\text{had} &\xrightarrow{\text{PU}} .
\end{aligned}$$

Parsing Example

$[\text{ROOT}]_S \quad [\,]_B$

$$\begin{aligned}
\text{Economic} &\xleftarrow{\text{ATT}} \text{news} \\
\text{news} &\xleftarrow{\text{SBJ}} \text{had} \\
\text{little} &\xleftarrow{\text{ATT}} \text{effect} \\
\text{financial} &\xleftarrow{\text{ATT}} \text{markets} \\
\text{on} &\xrightarrow{\text{PC}} \text{markets} \\
\text{effect} &\xrightarrow{\text{ATT}} \text{on} \\
\text{had} &\xrightarrow{\text{OBJ}} \text{effect} \\
\text{had} &\xrightarrow{\text{PU}} . \\
\text{ROOT} &\xrightarrow{\text{PRED}} \text{had}
\end{aligned}$$

Algorithm Analysis

- Given an input sentence of length n, the parser terminates after exactly $2n$ transitions.
- The algorithm is sound and complete for projective dependency trees.
- The algorithm is arguably optimal with respect to
  - robustness (at least one analysis),
  - disambiguation (at most one analysis),
  - efficiency (linear time).
- Accuracy depends on how well we can approximate oracles using machine learning.

A Theme with Variations

- Alternative transition systems:
  - Arc-standard shift-reduce [Yamada and Matsumoto 2003]
  - Arc-eager shift-reduce [Nivre 2003]
  - Limited non-projective [Attardi 2006]
  - Unrestricted non-projective [Nivre 2007, Nivre 2009]
- Alternative search strategies:
  - Deterministic single-pass [Nivre et al. 2004]
  - Deterministic multi-pass [Yamada and Matsumoto 2003]
  - Beam search [Johansson and Nugues 2006, Titov and Henderson 2007]
- Pseudo-projective parsing [Nivre and Nilsson 2005]:
  - Non-projective parsing through pre- and post-processing

Machine Learning – Oracles as Classifiers

- Learning problem in transition-based dependency parsing:
  - Approximate oracle $o(c)$ by classifier $g(c)$
- History-based feature models:
  - Parse history $c = (S, Q, A)$ represented by feature vector $\mathbf{x}(c)$
  - Individual features $\mathbf{x}_i(c)$ defined by properties of words in c, for example:
    - Lexical properties (FORM or LEMMA)
    - Part-of-speech tags (POSTAG)
    - Morphosyntactic features (FEATS)
    - Labels in the partially built dependency tree (DEPREL)

Typical Features

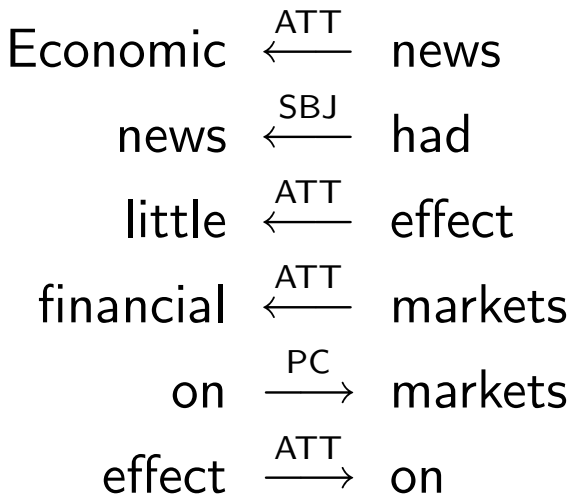
$[\text{ROOT}, \text{had}, \text{effect}]_S \quad [.]_B$

$$\begin{aligned}
\text{Economic} &\xleftarrow{\text{ATT}} \text{news} \\
\text{news} &\xleftarrow{\text{SBJ}} \text{had} \\
\text{little} &\xleftarrow{\text{ATT}} \text{effect} \\
\text{financial} &\xleftarrow{\text{ATT}} \text{markets} \\
\text{on} &\xrightarrow{\text{PC}} \text{markets} \\
\text{effect} &\xrightarrow{\text{ATT}} \text{on}
\end{aligned}$$

$$\begin{aligned}
&\text{POSTAG}(S[1]) = \text{VBD} \\
&\text{POSTAG}(S[0]) = \text{NN} \\
&\text{POSTAG}(B[0]) = \text{PU} \\
&\text{FORM}(S[1]) = \text{had} \\
&\text{FORM}(S[0]) = \text{effect} \\
&\text{FORM}(B[0]) = . \\
&\text{DEPREL}(\text{ldep}(S[1])) = \text{SBJ} \\
&\text{DEPREL}(\text{rdep}(S[1])) = \text{null} \\
&\text{DEPREL}(\text{ldep}(S[0])) = \text{ATT} \\
&\text{DEPREL}(\text{rdep}(S[0])) = \text{ATT}
\end{aligned}$$

Typical Features

$[\text{ROOT}, \text{had}, \text{effect}]_S \quad [.]_B$



$\text{POSTAG}(S[1]) = \text{VBD}$
$\text{POSTAG}(S[0]) = \text{NN}$
$\text{POSTAG}(B[0]) = \text{PU}$
$\text{FORM}(S[1]) = \text{had}$
$\text{FORM}(S[0]) = \text{effect}$
$\text{FORM}(B[0]) = .$
$\text{DEPREL}(\text{ldep}(S[1])) = \text{SBJ}$
$\text{DEPREL}(\text{rdep}(S[1])) = \text{null}$
$\text{DEPREL}(\text{ldep}(S[0])) = \text{ATT}$
$\text{DEPREL}(\text{rdep}(S[0])) = \text{ATT}$

Typical Features

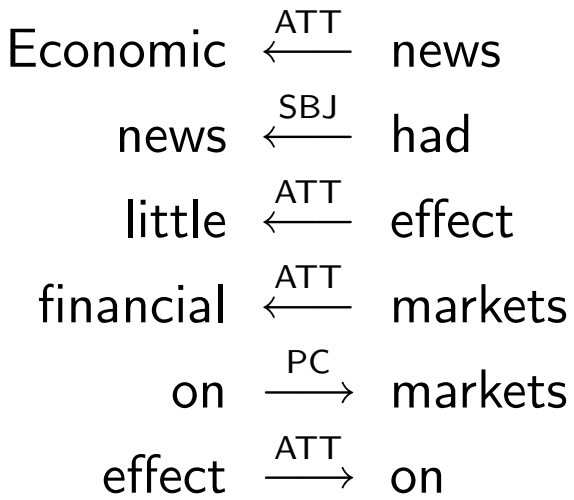
$[\text{ROOT}, \text{had}, \text{effect}]_S \quad [.]_B$

$$\begin{aligned}
\text{Economic} &\xleftarrow{\text{ATT}} \text{news} \\
\text{news} &\xleftarrow{\text{SBJ}} \text{had} \\
\text{little} &\xleftarrow{\text{ATT}} \text{effect} \\
\text{financial} &\xleftarrow{\text{ATT}} \text{markets} \\
\text{on} &\xrightarrow{\text{PC}} \text{markets} \\
\text{effect} &\xrightarrow{\text{ATT}} \text{on}
\end{aligned}$$

$$\begin{aligned}
&\text{POSTAG}(S[1]) = \text{VBD} \\
&\text{POSTAG}(S[0]) = \text{NN} \\
&\text{POSTAG}(B[0]) = \text{PU} \\
&\text{FORM}(S[1]) = \text{had} \\
&\text{FORM}(S[0]) = \text{effect} \\
&\text{FORM}(B[0]) = . \\
&\text{DEPREL}(\text{ldep}(S[1])) = \text{SBJ} \\
&\text{DEPREL}(\text{rdep}(S[1])) = \text{null} \\
&\text{DEPREL}(\text{ldep}(S[0])) = \text{ATT} \\
&\text{DEPREL}(\text{rdep}(S[0])) = \text{ATT}
\end{aligned}$$

Typical Features

$[\text{ROOT}, \text{had}, \text{effect}]_S \quad [.]_B$



$\text{POSTAG}(S[1]) = \text{VBD}$
$\text{POSTAG}(S[0]) = \text{NN}$
$\text{POSTAG}(B[0]) = \text{PU}$
$\text{FORM}(S[1]) = \text{had}$
$\text{FORM}(S[0]) = \text{effect}$
$\text{FORM}(B[0]) = .$
$\text{DEPREL}(\text{ldep}(S[1])) = \text{SBJ}$
$\text{DEPREL}(\text{rdep}(S[1])) = \text{null}$
$\text{DEPREL}(\text{ldep}(S[0])) = \text{ATT}$
$\text{DEPREL}(\text{rdep}(S[0])) = \text{ATT}$

Training Data

- Training instances have the form $(\mathbf{x}(c), t)$, where
  1. $\mathbf{x}(c)$ is a feature vector representation of a configuration c,
  2. t is the correct transition out of c (i.e., $o(c) = t$).
- Given a dependency treebank, we can sample the oracle function o as follows:
  - For each sentence we reconstruct the transition sequence $C_{0,m} = (c_0, c_1, \ldots, c_m)$ for the gold standard dependency tree.
  - For each configuration $c_i (i < m)$, we construct a training instance $(\mathbf{x}(c_i), t_i)$, where $t_i(c_i) = c_{i+1}$.

Learning Algorithms

- Discriminative models for classification:
  - Support vector machines [Kudo and Matsumoto 2002]
  - Memory-based learning [Nivre et al. 2004]
  - Logistic regression [Cheng et al. 2005, Attardi 2006]
  - Perceptron [Ciaramita and Attardi 2007, Zhang and Clark 2008]
- Learning strategies:
  - Local optimization [Yamada and Matsumoto 2003, Nivre et al. 2004]
  - Global optimization [Zhang and Clark 2008]

Empirical Results

MaltParser

- MaltParser [Nivre et al. 2006]:
  - Language-independent system for dependency parsing
  - Open source: http://maltparser.org
- Evaluated on over 25 languages:
  - CoNLL shared task 2006 [Buchholz and Marsi 2006]: 13 languages
  - CoNLL shared task 2007 [Nivre et al. 2007]: 10 languages
  - ICON NLP tools contest 2009 [Husain 2009]: 3 languages
  - State-of-the-art accuracy for most languages

Experimental Setup

- Data preparation:
  - Split available data into training, development, and test sets.
  - Preprocess data using available taggers, lemmatizers, etc.
- Evaluation metrics:
  - Attachment score (AS):
    - Percentage of words with correct head (w/wo labels)
  - Exact match (EM):
    - Percentage of sentences with correct trees (w/wo labels)

Some Recent Results

- English:
  - Data from the Penn Treebank converted to dependency trees
  - LAS = 88.3 (UAS = 91.6)
- French:
  - Data from the French Treebank converted to dependency trees
  - LAS = 87.3 (UAS = 89.7)
- German:
  - Data from the TiGer Treebank converted to dependency trees
  - LAS = 84.1 (UAS = 87.0)
- Czech:
  - Data from the Prague Dependency Treebank
  - LAS = 80.7 (UAS = 86.3)

Parsing Russian

- A study in treebank parsing [Nivre et al. 2008]:
  - Joint work with Igor Boguslavsky and Leonid Iomdin
  - Data from the SynTagRus treebank [Boguslavsky et al. 2000]
  - Gold standard pos-tags, lemmas and morphological features
- Features explored:
  - POS: Basic parts of speech
  - DEP: Dependency labels in partially built tree
  - MOR: Morphological features
  - LEM: Lemmas
  - LEX: Raw word forms

Experimental Results

Features	**LAS**	**UAS**
Baseline (POS + DEP)	60.2	70.6

Experimental Results

Features	**LAS**	**UAS**
Baseline (POS + DEP)	60.2	70.6
Baseline + MOR	73.0	84.5
Baseline + LEX	74.5	84.6
Baseline + LEM	75.5	84.6

Experimental Results

Features	LAS	UAS
Baseline (POS + DEP)	60.2	70.6
Baseline + MOR	73.0	84.5
Baseline + LEX	74.5	84.6
Baseline + LEM	75.5	84.6
Baseline + MOR + LEX	81.0	89.0
Baseline + MOR + LEM	82.3	89.0

Experimental Results

Features	LAS	UAS
Baseline (POS + DEP)	60.2	70.6
Baseline + MOR	73.0	84.5
Baseline + LEX	74.5	84.6
Baseline + LEM	75.5	84.6
Baseline + MOR + LEX	81.0	89.0
Baseline + MOR + LEM	82.3	89.0
Baseline + MOR + LEX + LEM	82.3	89.1

Conclusion

Conclusion

- Inductive dependency parsing is a robust and efficient methodology for dependency-based text parsing.
- Multilingual experimental evaluation shows that parsing accuracy is state of the art – but varies across languages.
- Future research directions:
  - Efficient non-projective dependency parsing
  - Better parsing models for richly inflected languages
  - Integration of grammar constraints
  - Psycholinguistic modeling

- Giuseppe Attardi. 2006.
  Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 166–170.
- Igor Boguslavsky, Svetlana Grigorieva, Nikolai Grigoriev, Leonid Kreidlin, and Nadezhda Frid. 2000.
  Dependency treebank for Russian: Concept, tools, types of information. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*, pages 987–991.
- Sabine Buchholz and Erwin Marsi. 2006.
  CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 149–164.
- Yuchang Cheng, Masayuki Asahara, and Yuji Matsumoto. 2005.
  Machine learning-based dependency analyzer for Chinese. In *Proceedings of International Conference on Chinese Computing (ICCC)*, pages 66–73.
- Massimiliano Ciaramita and Giuseppe Attardi. 2007.
  Dependency parsing with second-order feature maps and annotated semantic information. In *Proceedings of the Tenth International Conference on Parsing Technologies*, pages 133–143, June.
- Jason M. Eisner. 1996.
  Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages 340–345.
- Samar Husain, editor. 2009.
  Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing.
- Richard Johansson and Pierre Nugues. 2006.
  Investigating multilingual dependency parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 206–210.
- Taku Kudo and Yuji Matsumoto. 2002.
  Japanese dependency analysis using cascaded chunking. In *Proceedings of the Sixth Workshop on Computational Language Learning (CoNLL)*, pages 63–69.

- Ryan McDonald and Joakim Nivre. 2007.
  Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131.
- Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005.
  Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 91–98.
- Joakim Nivre and Jens Nilsson. 2005.
  Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 99–106.
- Joakim Nivre, Johan Hall, and Jens Nilsson. 2004.
  Memory-based dependency parsing. In *Proceedings of the 8th Conference on Computational Natural Language Learning*, pages 49–56.
- Joakim Nivre, Johan Hall, and Jens Nilsson. 2006.
  Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, pages 2216–2219.
- Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007.
  The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*, pages 915–932.
- Joakim Nivre, Igor M. Boguslavsky, and Leonid L. Iomdin. 2008.
  Parsing the SynTagRus treebank of russian. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 641–648.
- Joakim Nivre. 2003.
  An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160.

- Joakim Nivre. 2007.
  Incremental non-projective dependency parsing. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT)*, pages 396–403.
- Joakim Nivre. 2009.
  Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP)*, pages 351–359.
- Ivan Titov and James Henderson. 2007.
  A latent variable model for generative dependency parsing. In *Proceedings of the 10th International Conference on Parsing Technologies (IWPT)*, pages 144–155.
- Hiroyasu Yamada and Yuji Matsumoto. 2003.
  Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 195–206.
- Yue Zhang and Stephen Clark. 2008.
  A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 562–571.