Parsing of natural language sentences to syntactic and semantic graph representations

Abschlussvortrag zum Forschungsprojekt

Pius Meinert

April 13, 2018

Overview

Graph Representations

Corpora

Parsing Techniques

Parser

Overview

Graph Representations

Corpora

Parsing Techniques

Parser

Semantic: AMR, UCCA, dependency graphs

Syntactic: Constituency tree derived, Use of syntactic information

Overview

Graph Representations

Corpora

Parsing Techniques

Parser

AMR, UCCA, SemEval-2014/-2015: dependency graphs, Penn Treebank, TIGER Corpus

Overview

Graph Representations

Corpora

Parsing Techniques

- Maximum Subgraph
- Transition-Based
- Synchronous HRG

Parser

Overview

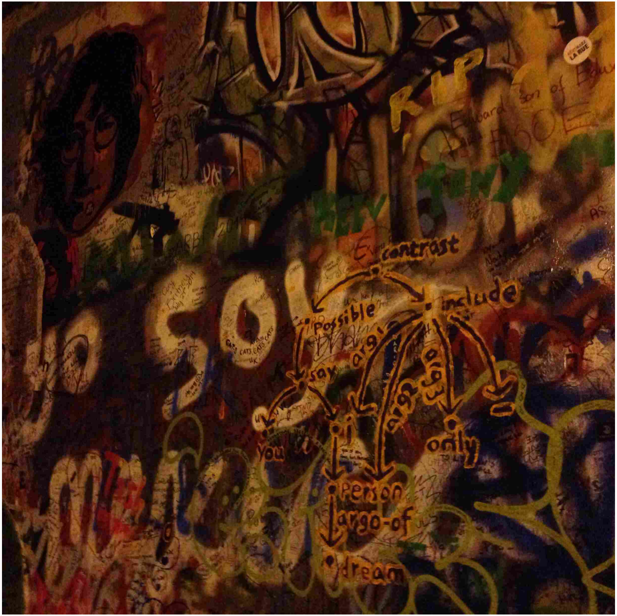
Graph Representations

Corpora

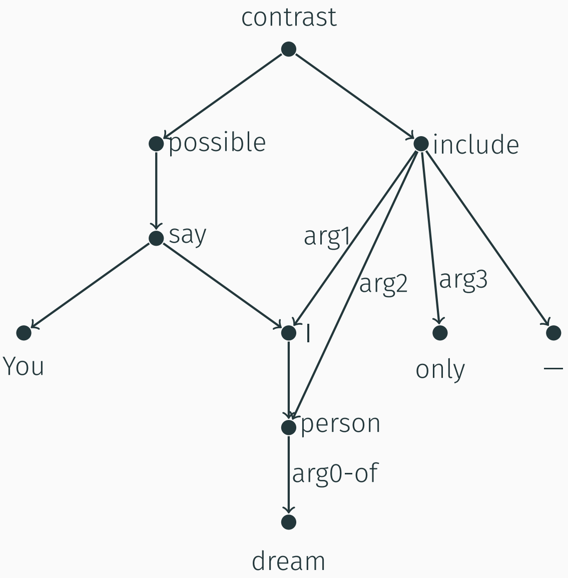
Parsing Techniques

Parser

Abstract Meaning Representation (AMR) [Ban+13]

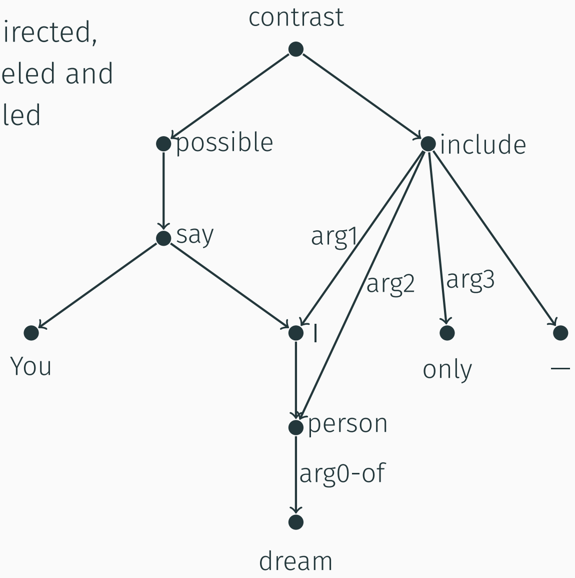


Abstract Meaning Representation (AMR) [Ban+13]



Abstract Meaning Representation (AMR) [Ban+13]

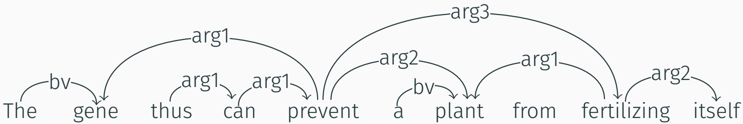
rooted, directed, edge-labeled and leaf-labeled



Tree Definition

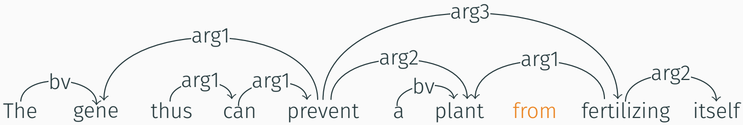
A tree is a directed graph $G = (V, A)$ that has a vertex r, named root, such that every vertex $v \in V$ is reachable from r via a unique directed path. [KJ15; KO16]

Dependency Graph [KJ15]



Dependency Graph [KJ15]

unconnected

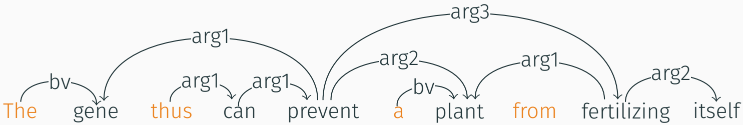


Connectedness:

There exists an undirected path between every two pairs of vertices. Nodes with in- and out-degree zero are called singletons. [KO16]

Dependency Graph [KJ15]

unconnected, multi-rooted

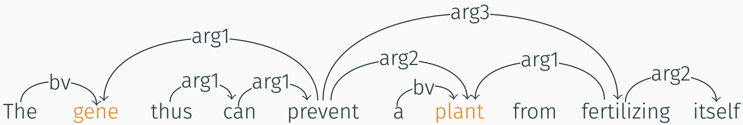


Top nodes:

Nodes of in-degree zero, a graph’s equivalent to the unique root in a tree. [KO16]

Dependency Graph [KJ15]

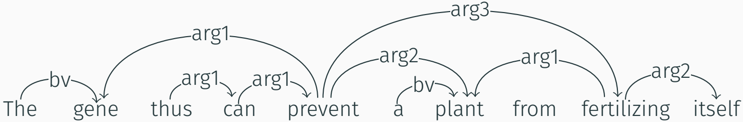
unconnected, multi-rooted, reentrancy



Reentrant nodes:

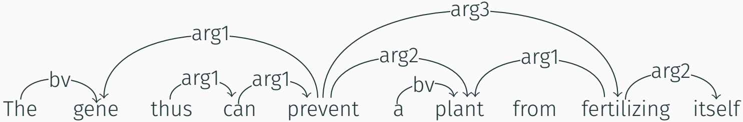
Nodes with in-degree greater than one. [WXP15; DCS17; BB17]

Dependency Graph - Noncrossing [KJ15]

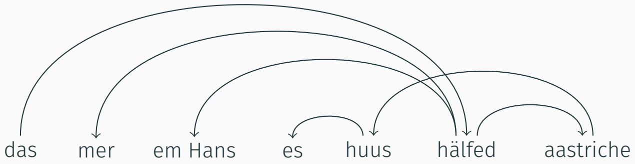


Dependency Graph - Noncrossing [KJ15]

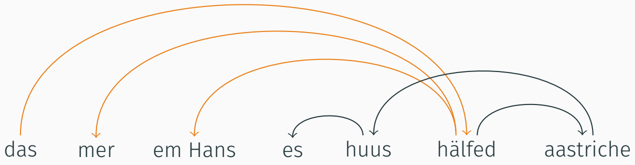
Coverage ranges from 48% to 78% for various graph banks (CCGbank, Prage Semantic Dependencies, etc.). [KJ15; SCW17]



Dependency Graph - 1-Endpoint-Crossing [PKM13]

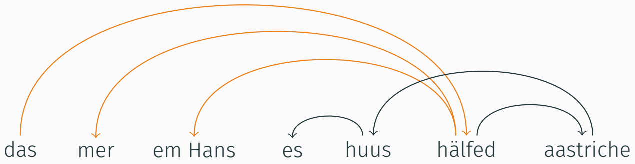


Dependency Graph - 1-Endpoint-Crossing [PKM13]

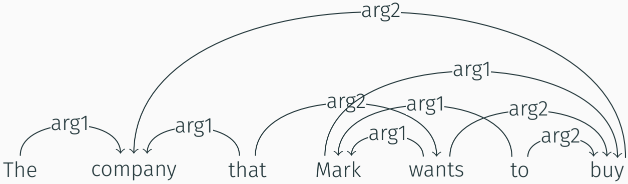


Dependency Graph - 1-Endpoint-Crossing [PKM13]

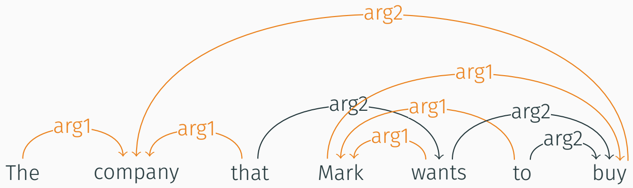
Account for 95.7 – 97.7% of the dependency structures that are used in [Cao+17].



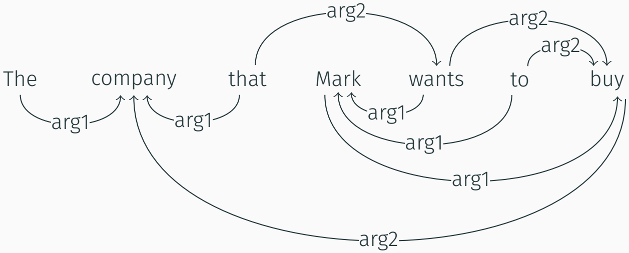
Dependency Graph - Book Embedding [SCW17]



Dependency Graph - Book Embedding [SCW17]



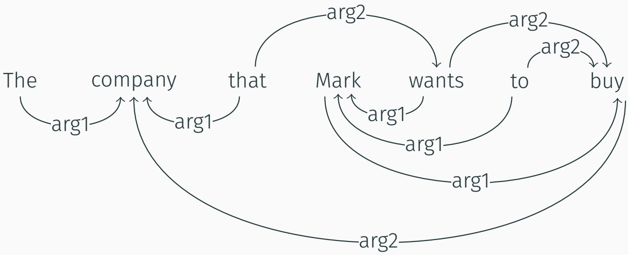
Dependency Graph - Book Embedding [SCW17]



Dependency Graph - Book Embedding [SCW17]

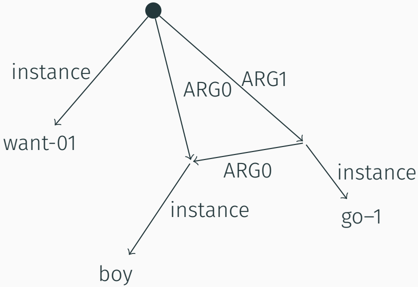
Coverage with respect to different page numbers:

PN	coverage
1	48 – 78%
2	20 – 49%
3	0.3 – 1.7%



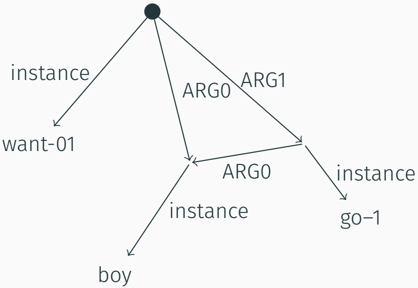
Evaluation Metric - AMR Representations

AMR graph



Evaluation Metric - AMR Representations

AMR graph

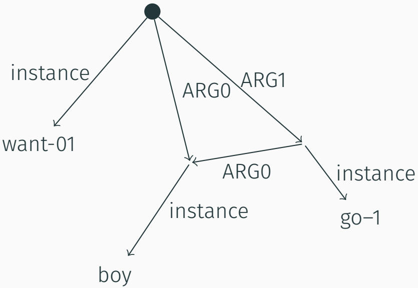


PENMAN notation

```
(w / want-01
  :arg0 (b / boy)
  :arg1 (g / go-01)
          :arg0 b)
```

Evaluation Metric - AMR Representations

AMR graph



logic format

```
instance(a, want-01) ∧
instance(b, boy) ∧
instance(c, go-01) ∧
ARG0(a, b) ∧
ARG1(a, c) ∧
ARG0(c, b)
```

Evaluation Metric - Smatch [CK13]

The boy wants the football

```
instance(x, want-01) ∧
instance(y, boy) ∧
instance(z, football) ∧
ARG0(x, y) ∧
ARG1(x, z)
```

The boy wants to go

```
instance(a, want-01) ∧
instance(b, boy) ∧
instance(c, go-01) ∧
ARG0(a, b) ∧
ARG1(a, c) ∧
ARG0(c, b)
```

Evaluation Metric - Smatch [CK13]

The boy wants the football

```
instance(x, want-01) ∧
instance(y, boy) ∧
instance(z, football) ∧
ARG0(x, y) ∧
ARG1(x, z)
```

The boy wants to go

```
instance(a, want-01) ∧
instance(b, boy) ∧
instance(c, go-01) ∧
ARG0(a, b) ∧
ARG1(a, c) ∧
ARG0(c, b)
```

inter-annotator agreement study:

Smatch score ranges from 0.79 to 0.83.

Graph Parsing Techniques

Maximum Subgraph

“all pairs” approach [BM06] - Consider all possible (weighted) arcs and find the maximum spanning connected subgraph.

Transition-based

“stepwise” approach [BM06] - Build the graph step by step by applying transitions to the current configuration.

Synchronous Hyperedge Replacement Grammar (SHRG)

HRGs as “an intuitive generalization of context free grammars (CFGs) from strings to hypergraphs.” [Jon+12; Hab92]

Maximum Subgraph - Problem Definition[SCW17]

Input directed, weighted graph $G = (V, A)$ (complete)

Implicit sentence s, class of graphs $\mathcal{G}$

Output subgraph $G' = (V, A' \subseteq A)$ with maximum total weight such that G' belongs to $\mathcal{G}$

$$G'(s) = \arg\max_{H \in \mathcal{G}(s,G)} \sum_{p \in H} \textsc{ScorePart}(s, p)$$

Example if class of graphs $\mathcal{G}$ is the class of all trees, Maximum Subgraph = Maximum Spanning Tree

Maximum Subgraph - Learning and Features [MN07]

$$G'(s) = \arg\max_{H \in \mathcal{G}(s,G)} \sum_{p \in H} \textsc{ScorePart}(s, p)$$

Maximum Subgraph - Learning and Features [MN07]

Global learning: Optimize entire graph score, not only single arc attachments.

$$G'(s) = \arg\max_{H \in \mathcal{G}(s,G)} \sum_{p \in H} \textsc{ScorePart}(s, p)$$

Maximum Subgraph - Learning and Features [MN07]

Global learning Optimize entire graph score, not only single arc attachments.

$$G'(s) = \arg\max_{H \in \mathcal{G}(s,G)} \sum_{p \in H} \textsc{ScorePart}(s, p)$$

Local features Restricted to a limited number of arcs (to keep inference and learning tractable).

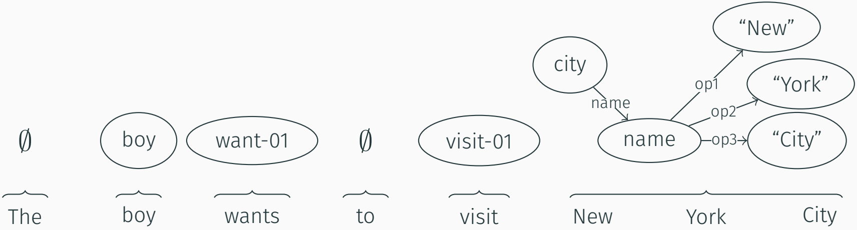
JAMR [Fla+14]

First published AMR parser. It solves the task by means of two phases:

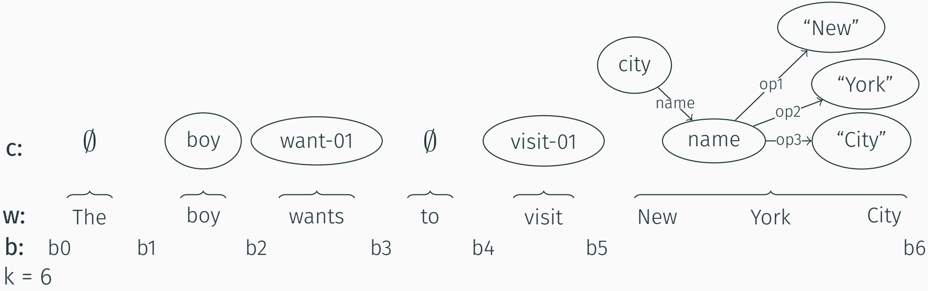
Concept identification: Match spans of words to concept graph fragments.

Relation identification: Find the maximum spanning connected subgraph over those graph fragments.

JAMR - Concept Identification Phase [Fla+14]



JAMR - Concept Identification Phase [Fla+14]



$$score(\mathbf{b}, \mathbf{c}; \theta) = \sum_{i=1}^{k} \theta^\top \mathbf{f}(\mathbf{w}_{b_{i-1}:b_i}, b_{i-1}, b_i, c_i)$$

Solve by dynamic programming: $\mathcal{O}(n^2)$.

JAMR - Relation Identification Phase [Fla+14]

1. Initialization:
   Include all edges and vertices given by the concept identification phase.
2. Pre-processing:
   Reduce the set of edges considered to one edge per pair of nodes: Either the edge given by the first phase or the highest scoring one.
3. Core algorithm:
   First, add all positive edges and then greedily add the least negative edge that connects two components until the graph is connected.

Transition-Based - Transition System [WXP15]

A transition system for parsing is a tuple $S = (S, T, s_0, S_t)$ where

- S is a set of parsing states (configurations).
- T is a set of parsing actions (transitions), each of which is a function $t : S \rightarrow S$.
- s_0 is an initialization function, mapping each input sentence w to an initial state.
- $S_t \subseteq S$ is a set of terminal states.

Transition-Based - Parsing Algorithm [WXP15]

```
**Input:** sentence $w = w_0 ... w_n$
**Output:** parsed graph $G$
$s \leftarrow s_0(w)$
while $s \notin S_t$ do
    $\mathcal{T} \leftarrow$ all possible actions according to $s$
    $bestT \leftarrow \arg\max_{t \in \mathcal{T}} score(t, s)$
    $s \leftarrow$ apply $bestT$ to $s$
end while
return $G$
```

Transition-Based - Learning and Features [MN07]

$$bestT \leftarrow \arg\max_{t \in \mathcal{T}} score(t, s)$$

Transition-Based - Learning and Features [MN07]

Local learning — Optimization only for single transitions, not transition sequences.

$$bestT \leftarrow \arg\max_{t \in \mathcal{T}} score(t, s)$$

Transition-Based - Learning and Features [MN07]

Local learning — Optimization only for single transitions, not transition sequences.

$$bestT \leftarrow \operatorname{arg\,max}_{t \in \mathcal{T}} score(t, s)$$

Global features — Features may be based on whole graph built so far/entire transition history.

Transition-Based AMR Parser [WXP15]

Idea: Use similarities between an AMR and the dependency structure of a sentence.

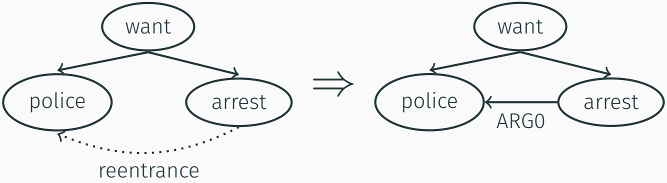
Two-stage framework:

1) dependency parser to generate dependency tree for the sentence
2) transition-based algorithm to transform dependency tree to an AMR graph

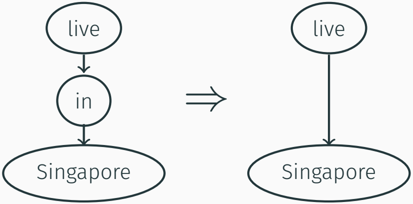
The dependency parser can be trained on a much larger data set.

Transition-Based AMR Parser - Transition Actions [WXP15]

REENTRANCE action



REPLACE-HEAD action



Towards a Catalogue of Linguistic Graph Banks [KO16]

- graph structures are of growing relevance to much NLP research
- provide common terminology and transparent statistics for different (collections of) graphs
- propose to establish shared community resource: `https://aclweb.org/aclwiki/Graph_Parsing_(State_of_the_Art)`

References i

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. “Abstract Meaning Representation for Sembanking”. In: *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, LAW-ID@ACL 2013, August 8-9, 2013, Sofia, Bulgaria*. 2013, pp. 178–186. URL: `http://aclweb.org/anthology/W/W13/W13-2322.pdf`.

References ii

Jan Buys and Phil Blunsom. “Robust Incremental Neural Semantic Graph Parsing”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. 2017, pp. 1215–1226. DOI: `10.18653/v1/P17-1112`. URL: `https://doi.org/10.18653/v1/P17-1112`.

References iii

Sabine Buchholz and Erwin Marsi. “CoNLL-X Shared Task on Multilingual Dependency Parsing”. In: *Proceedings of the Tenth Conference on Computational Natural Language Learning, CoNLL 2006, New York City, USA, June 8-9, 2006*. 2006, pp. 149–164. URL: http://aclweb.org/anthology/W/W06/W06-2920.pdf.

References iv

Junjie Cao, Sheng Huang, Weiwei Sun, and Xiaojun Wan. “Parsing to 1-Endpoint-Crossing, Pagenumber-2 Graphs”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. 2017, pp. 2110–2120. DOI: `10.18653/v1/P17-1193`. URL: `https://doi.org/10.18653/v1/P17-1193`.

References v

Shu Cai and Kevin Knight. “Smatch: an Evaluation Metric for Semantic Feature Structures”. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers*. 2013, pp. 748–752. URL: `http://aclweb.org/anthology/P/P13/P13-2131.pdf`.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. “An Incremental Parser for Abstract Meaning Representation”. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*. 2017, pp. 536–546. URL: `https://aclanthology.info/papers/E17-1051/e17-1051`.

Jeffrey Flanigan, Sam Thomson, Jaime G. Carbonell, Chris Dyer, and Noah A. Smith. “A Discriminative Graph-Based Parser for the Abstract Meaning Representation”. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*. 2014, pp. 1426–1436. URL: http://aclweb.org/anthology/P/P14/P14-1134.pdf.

Annegret Habel. *Hyperedge Replacement: Grammars and Languages*. Vol. 643. Lecture Notes in Computer Science. Springer, 1992.

References viii

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. “Semantics-Based Machine Translation with Hyperedge Replacement Grammars”. In: *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*. 2012, pp. 1359–1376. URL: `http://aclweb.org/anthology/C/C12/C12-1083.pdf`.

Marco Kuhlmann and Peter Jonsson. “Parsing to Noncrossing Dependency Graphs”. In: *TACL* 3 (2015), pp. 559–570. URL: `https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/709`.

Marco Kuhlmann and Stephan Oepen. “Towards a Catalogue of Linguistic Graph Banks”. In: *Computational Linguistics* 42.4 (2016), pp. 819–827. DOI: `10.1162/COLI_a_00268`. URL: `https://doi.org/10.1162/COLI_a_00268`.

References x

Ryan T. McDonald and Joakim Nivre. “Characterizing the Errors of Data-Driven Dependency Parsing Models”. In: *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*. 2007, pp. 122–131. URL: `http://www.aclweb.org/anthology/D07-1013`.

Emily Pitler, Sampath Kannan, and Mitchell Marcus. “Finding Optimal 1-Endpoint-Crossing Trees”. In: *TACL* 1 (2013), pp. 13–24. URL: `https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/23`.

References xi

Weiwei Sun, Junjie Cao, and Xiaojun Wan. “Semantic Dependency Parsing via Book Embedding”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. 2017, pp. 828–838. DOI: `10.18653/v1/P17-1077`. URL: `https://doi.org/10.18653/v1/P17-1077`.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. "A Transition-based Algorithm for AMR Parsing". In: *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*. 2015, pp. 366–375. URL: `http://aclweb.org/anthology/N/N15/N15-1040.pdf`.