Density-Driven Cross-Lingual Transfer of Dependency Parsers

Mohammad Sadegh Rasooli Michael Collins

rasooli@cs.columbia.edu

Presented by
Owen Rambow

EMNLP 2015

Motivation

Availability of treebanks

- Accurate parsers use annotated treebanks.
- There are no gold-standard treebanks for many languages.
- Annotated treebanks are very expensive to create.

Common approach: using universal linguistic information

- Without parallel data; e.g [Zhang and Barzilay, 2015]
- With parallel data; e.g [Ma and Xia, 2014]
  - The best results but still lags behind supervised parsing

Motivation

Availability of treebanks

- Accurate parsers use annotated treebanks.
- There are no gold-standard treebanks for many languages.
- Annotated treebanks are very expensive to create.

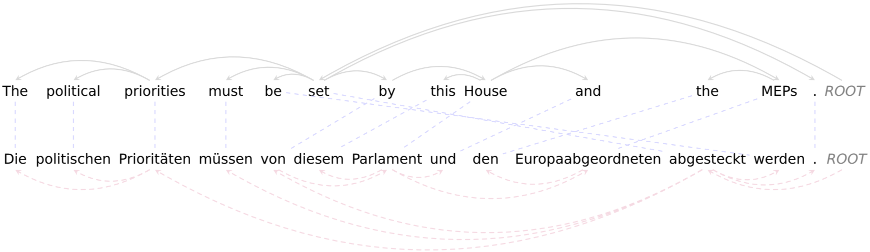
Common approach: using universal linguistic information

- Without parallel data; e.g [Zhang and Barzilay, 2015]
- With parallel data; e.g [Ma and Xia, 2014]
    - The best results but still lags behind supervised parsing

Projecting Dependencies from Parallel Data

Bitext

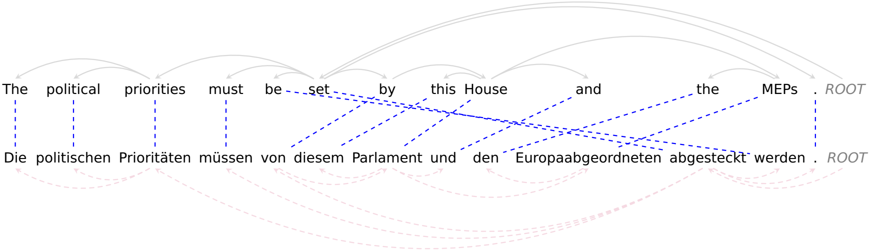
Prepare bitext



Projecting Dependencies from Parallel Data

Align

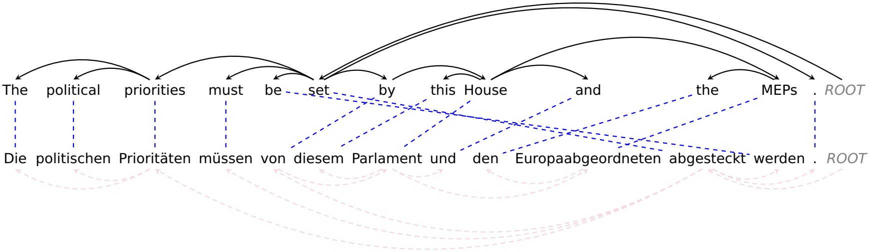
Align bitext (e.g. via Giza++)



Projecting Dependencies from Parallel Data

Parse

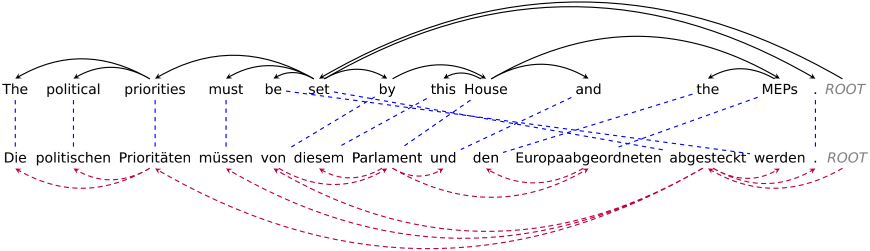
Parse source sentence with a supervised parser.



Projecting Dependencies from Parallel Data

Project

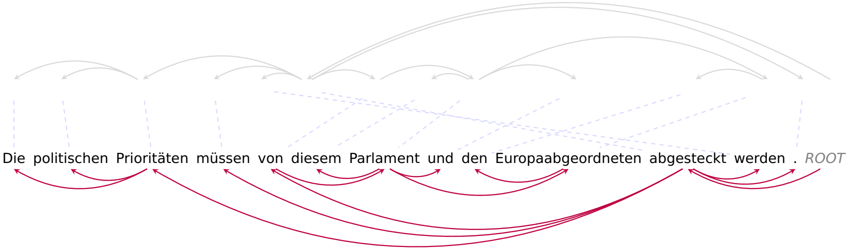
Project dependencies.



Projecting Dependencies from Parallel Data

Train

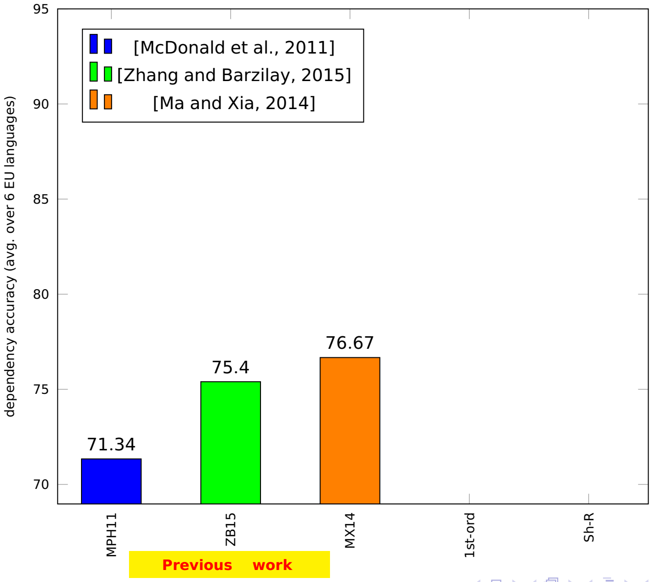
Train on the projected dependencies.



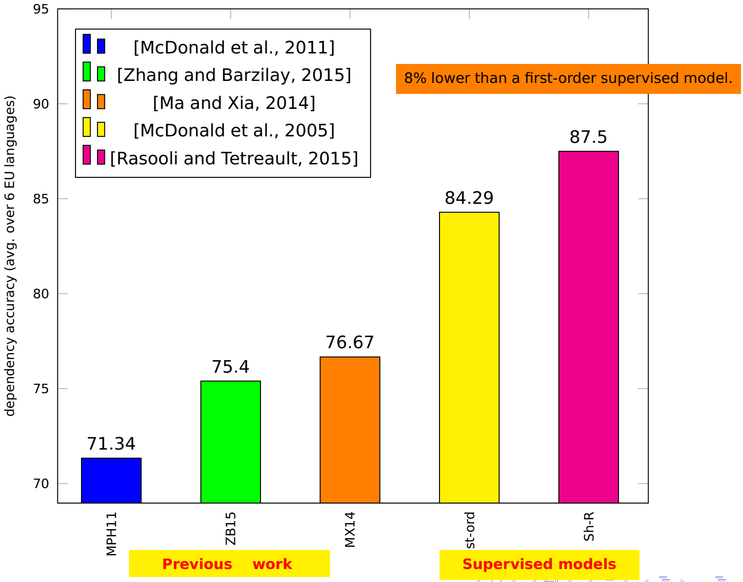
Practical Problems

- Most translations are not word-to-word.
- Alignment errors!
- Supervised parsers are not perfect.
- Difference in syntactic behavior across languages.

Previous Results



Previous Results



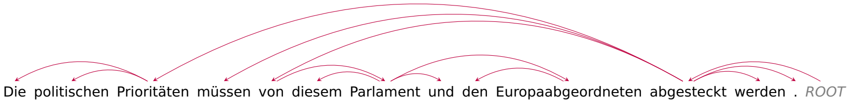
Our Approach

- We define different sets of dense structures
    - Full trees
    - Dense partial trees

Dense Structures

A projected **full tree** $t \in \mathcal{P}_{100}$ is:

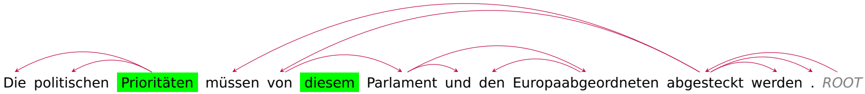
- A projective dependency tree
- All words have one parent



Dense Structures

A **partial tree** $t \in \mathcal{P}_{80}$ is:

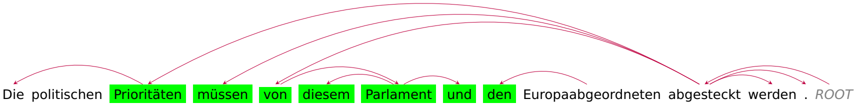
- A projective dependency tree (a collection of projective trees)
- At least 80% of words have one parent



Dense Structures

A **partial tree** $t \in \mathcal{P}_{\geq k}$ is:

- A projective dependency tree (a collection of projective trees)
- There is at least one span of length$\geq k$ where all words in that span have one parent

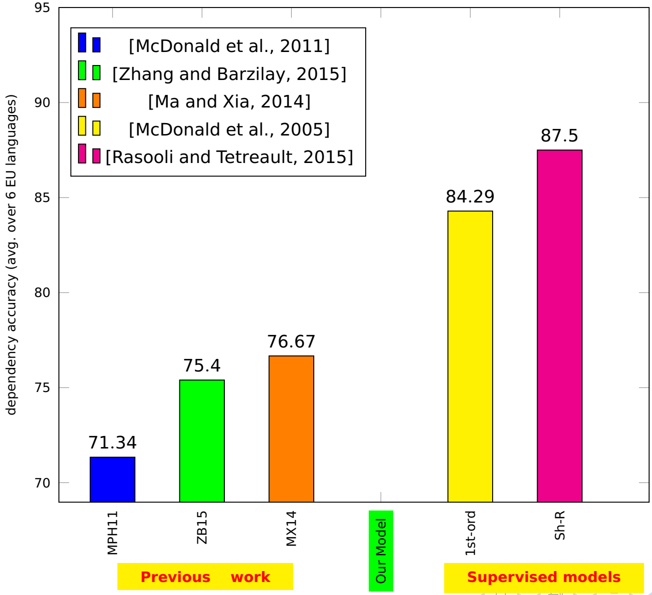


k=7 in the above tree

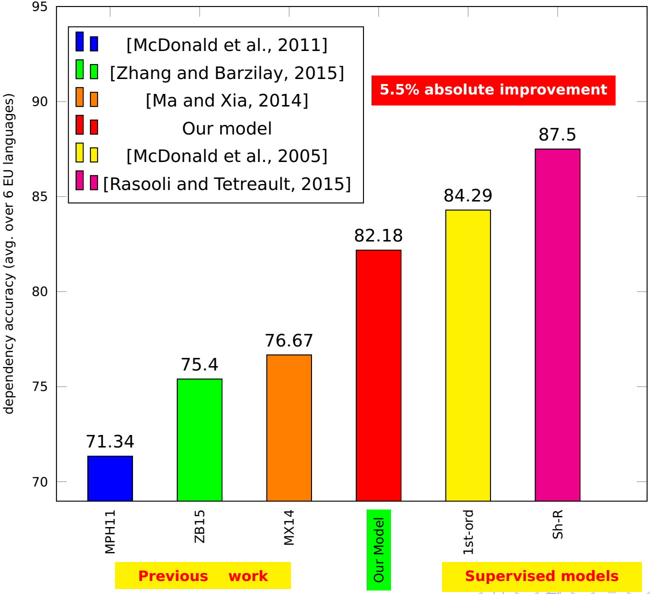
Our Contributions

- We demonstrate the utility of dense projected structures.
- We describe a training algorithm that builds on dense structures.

Our Contributions



Our Contributions



Overview

- **The learning algorithm**
- Results
- Analysis

Projecting Dependencies

- Languages from Google universal treebank:
  - English (only as source), German, Spanish, French, Italian, Portuguese, and Swedish.
  - English to German transfer data for developing our models.
- We use Giza++ intersected alignments on EuroParl data
- We use the Yara parser [Rasooli and Tetreault, 2015], a shift-reduce beam parser.

Functions Used in Our Algorithm

We use the following function definitions:

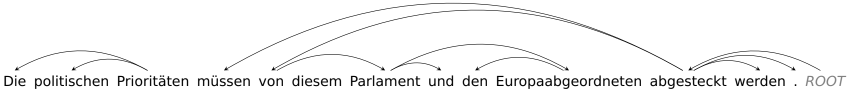
- Train(D)
- CDECODE(P, θ)
- TOP(D, θ)

Train(D)

- Input D
  - A set of dependency trees (full trees)
- Output θ
  - A parsing model

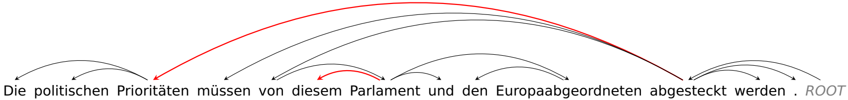
CDECODE(P, θ)

- Input P
  - A set of partial dependency structures
- Input θ
  - Parsing model
- Output D
  - A set of full trees that are completely consistent with the dependencies in P.
  - Filling in partial trees with dynamic oracles [Goldberg and Nivre, 2013].



CDECODE(P, θ)

- Input P
  - A set of partial dependency structures
- Input θ
  - Parsing model
- Output D
  - A set of full trees that are completely consistent with the dependencies in P.
  - Filling in partial trees with dynamic oracles [Goldberg and Nivre, 2013].



TOP(D, θ)

- Input D
  - A set of full dependency trees
- Input θ
  - Parsing model
- Output A
  - Top m highest scoring trees in D
    - We use m=200,000 in our experiments.
  - **Score**: Perceptron-based parse score normalized by sentence length

Definitions

- $A_0 = P_{100}$
- $A_1 = P_{\geq 7} \cup P_{80}$
- $A_2 = P_{\geq 5} \cup P_{80}$
- $A_3 = P_{\geq 1} \cup P_{80}$

Note $A_1 \subseteq A_2 \subseteq A_3$

Learning Algorithm

Train on full trees

θ_0 **= Train(**A_0**)**
for $i = 1 \ldots 3$ **do**
  D_i = CDECODE(A_i, θ_{i-1})
  A'_i = TOP(D_i, θ_{i-1})
  θ_i = Train($A_0 \cup A'_i$)
end for
Return θ_3

Given definitions:

- $A_0 = P_{100}$
- $A_1 = P_{\geq 7} \cup P_{80}$
- $A_2 = P_{\geq 5} \cup P_{80}$
- $A_3 = P_{\geq 1} \cup P_{80}$

Note $A_1 \subseteq A_2 \subseteq A_3$

Learning Algorithm

Gradually decrease density

$\theta_0 = \text{Train}(A_0)$

for **i = 1 ... 3** **do**

  $D_i = \text{CDECODE}(A_i, \theta_{i-1})$

  $A'_i = \text{TOP}(D_i, \theta_{i-1})$

  $\theta_i = \text{Train}(A_0 \cup A'_i)$

end for

Return θ_3

Given definitions:

- $A_0 = P_{100}$
- $A_1 = P_{\geq 7} \cup P_{80}$
- $A_2 = P_{\geq 5} \cup P_{80}$
- $A_3 = P_{\geq 1} \cup P_{80}$

Note $A_1 \subseteq A_2 \subseteq A_3$

Learning Algorithm

Fill in partial trees

θ_0 = Train(A_0)
for $i = 1 \ldots 3$ **do**
    D_i **= CDECODE(**A_i**,** θ_{i-1}**)**
    A'_i **= TOP(**D_i**,** θ_{i-1}**)**
    θ_i **= Train(**$A_0 \cup A'_i$**)**
end for
Return θ_3

Given definitions:

- $A_0 = P_{100}$
- $A_1 = P_{\geq 7} \cup P_{80}$
- $A_2 = P_{\geq 5} \cup P_{80}$
- $A_3 = P_{\geq 1} \cup P_{80}$

Note $A_1 \subseteq A_2 \subseteq A_3$

Learning Algorithm

Select high-scoring trees

θ_0 = Train(A_0)
for $i = 1...3$ **do**
  D_i = CDECODE(A_i, θ_{i-1})
  A'_i = **TOP(D_i, θ_{i-1})**
  θ_i = Train($A_0 \cup A'_i$)
end for
Return θ_3

Given definitions:

- $A_0 = P_{100}$
- $A_1 = P_{\geq 7} \cup P_{80}$
- $A_2 = P_{\geq 5} \cup P_{80}$
- $A_3 = P_{\geq 1} \cup P_{80}$

Note $A_1 \subseteq A_2 \subseteq A_3$

Learning Algorithm

Train on the new set

θ_0 = Train(A_0)
for $i = 1 \ldots 3$ **do**
    D_i = CDECODE(A_i, θ_{i-1})
    A'_i = TOP(D_i, θ_{i-1})
    θ_i = Train($A_0 \cup A'_i$)
end for
Return θ_3

Given definitions:

- $A_0 = P_{100}$
- $A_1 = P_{\geq 7} \cup P_{80}$
- $A_2 = P_{\geq 5} \cup P_{80}$
- $A_3 = P_{\geq 1} \cup P_{80}$

Note $A_1 \subseteq A_2 \subseteq A_3$

Learning Algorithm

Return the final model

$\theta_0 = \text{Train}(A_0)$
for $i = 1...3$ **do**
    $D_i = \text{CDECODE}(A_i, \theta_{i-1})$
    $A'_i = \text{TOP}(D_i, \theta_{i-1})$
    $\theta_i = \text{Train}(A_0 \cup A'_i)$
end for
Return θ_3

Given definitions:

- $A_0 = P_{100}$
- $A_1 = P_{\geq 7} \cup P_{80}$
- $A_2 = P_{\geq 5} \cup P_{80}$
- $A_3 = P_{\geq 1} \cup P_{80}$

Note $A_1 \subseteq A_2 \subseteq A_3$

Overview

- The learning algorithm
- **Results**
- Analysis

Two Settings

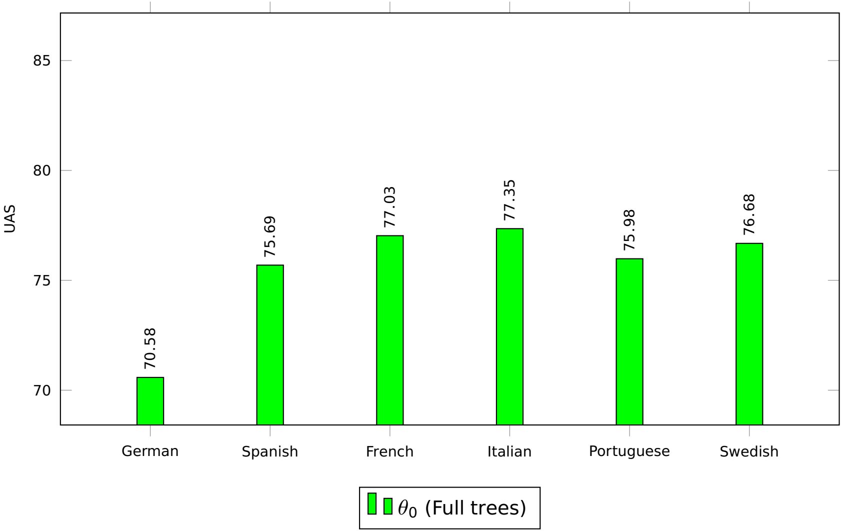
Scenario 1

- Transfer from English.

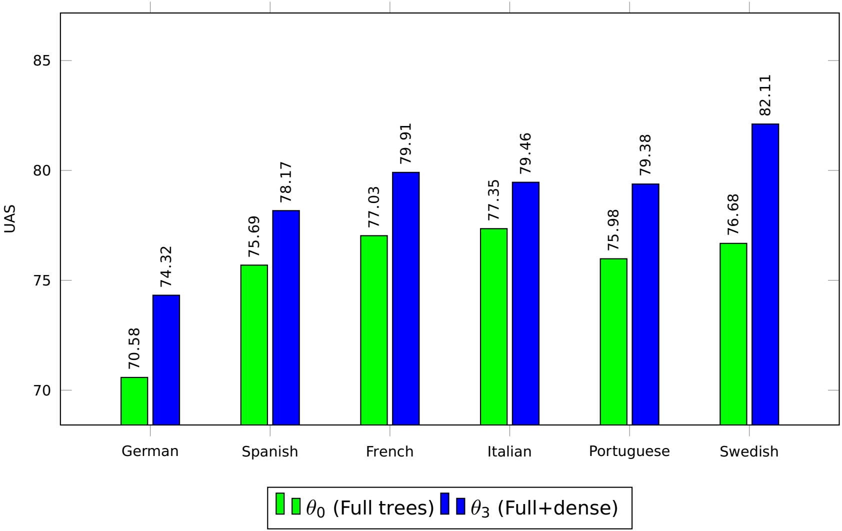
Scenario 2 (voting)

- The different languages vote on dependencies.
    - This scenario is true for cases such as Europarl.

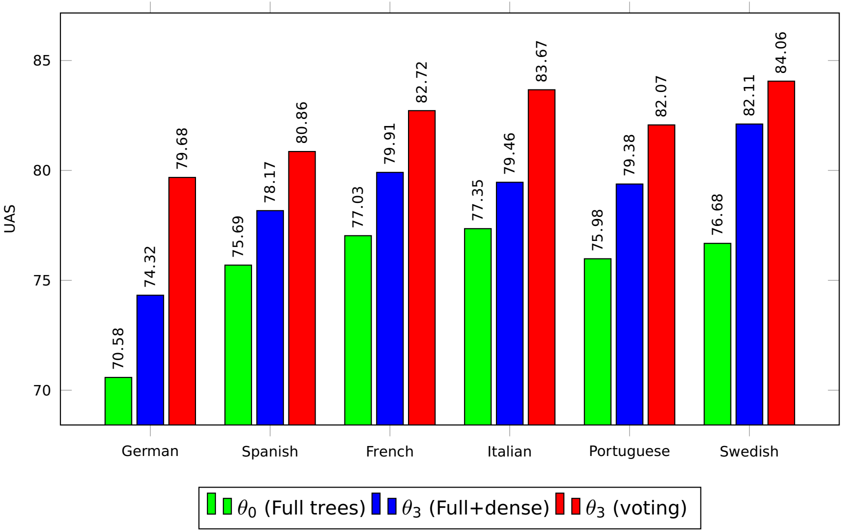
Results on European Languages



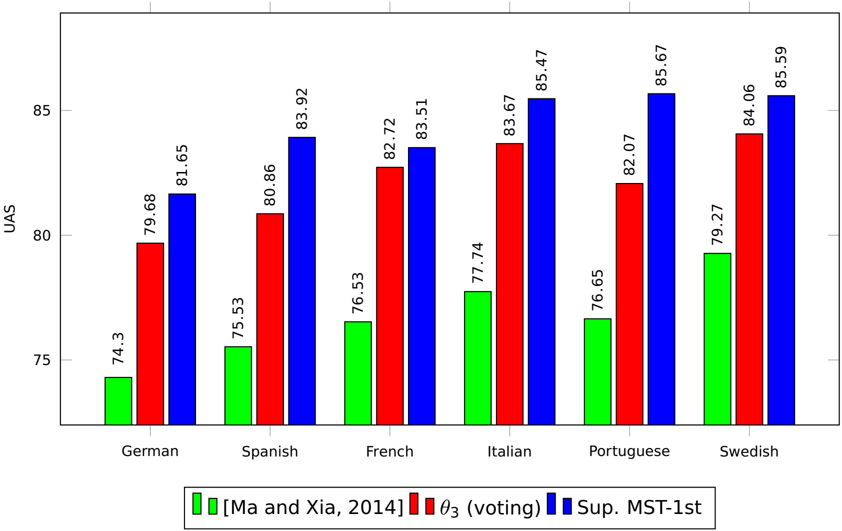
Results on European Languages



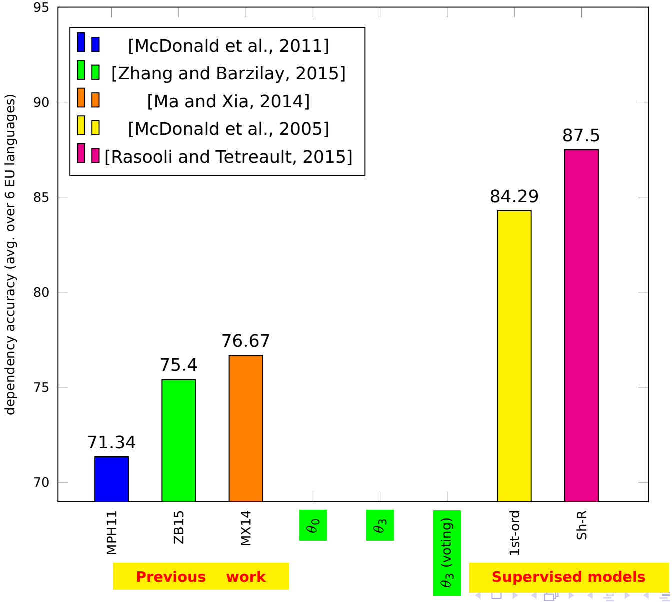
Results on European Languages



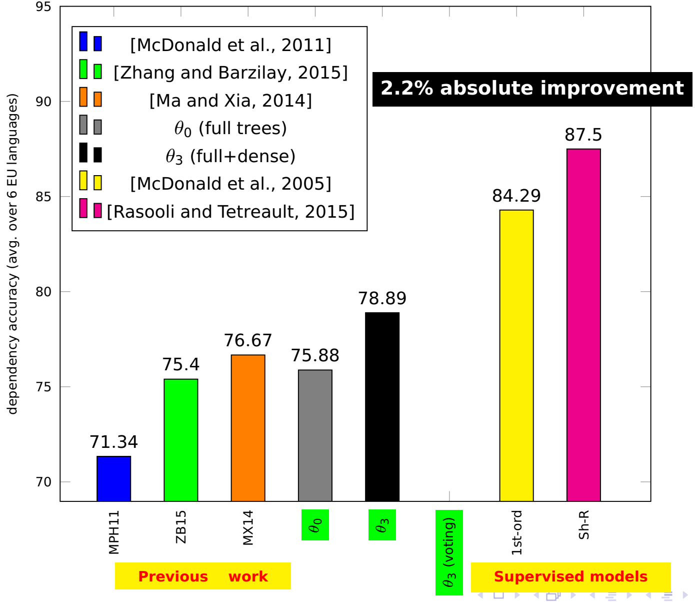
Results on European Languages (Comparison)



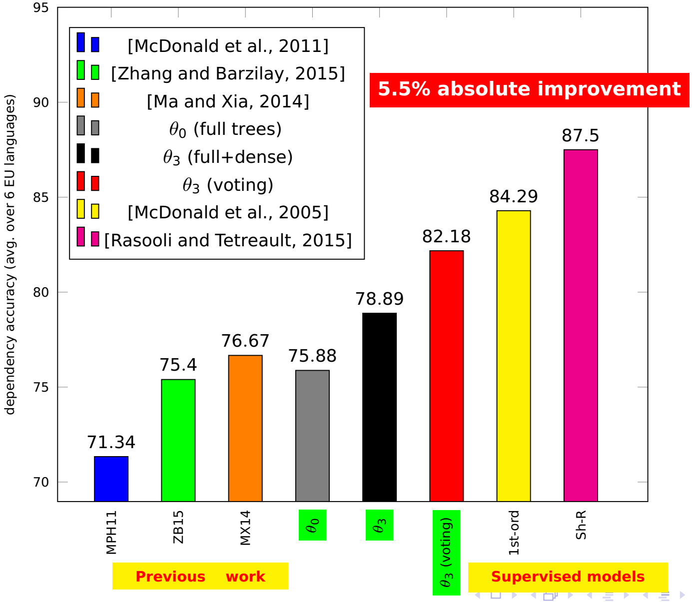
Comparison to Previous Work



Comparison to Previous Work



Comparison to Previous Work



Overview

- The learning algorithm
- Results
- **Analysis**

Accuracy of Full Trees

- The accuracy of full trees is high.
- Voting increases the number of words per sentence, number of sentences and accuracy of full trees.

Setting	English→target	Voting
Sen#	17K	77K
Word/sen	6.8	10.4
Prec. vs supervised	84.7	89.0

Density of Partial Trees in Voting

- The length and number of sentences are increased in partial dense trees.
- The accuracy of partial trees are lower than full trees.

Setting	$\mathcal{P}_{100}$	$\mathcal{P}_{80} \cup \mathcal{P}_{\geq 7}$
Sen#	77K	243K
Deps#	10.4	13.7
Words/sen	10.4	27.6
Density	100%	50%
Prec. vs supervised	89.0	84.7

Accuracy across Different Languages

Language	$\mathcal{P}_{100}$				$\mathcal{P}_{80} \cup \mathcal{P}_{\geq 7}$				Sup.
	#sen	words/sen	#dep	Prec.	#sen	words/sen	#dep	Prec.	
German	47K	8.2	8.2	91.4	75K	23.5	10.8	84.5	85.34
Spanish	109K	12.1	12.1	89.2	346K	28.5	17.0	86.1	86.69
French	78K	11.7	11.7	91.2	303K	29.9	14.9	87.4	86.24
Italian	101K	12.4	12.4	87.9	301K	28.5	15.2	84.5	88.83
Portuguese	39K	8.8	8.8	85.8	222K	30.3	12.4	81.3	89.44
Swedish	86K	9.5	9.5	88.8	211K	25.2	12.2	84.2	88.06
Average	77K	10.4	10.4	89.0	243K	27.6	13.7	84.7	87.50

Conclusion

- We showed the utility of dense structures in projected dependencies.
- We showed a simple and effective learning method to utilize dense structures.
- Our performance is very close to a supervised parser.
- Future work:
  - Applying to a broader set of languages.
  - Using this model to improve machine translation.

Thanks

Bloomberg

References I

Goldberg, Y. and Nivre, J. (2013).
Training deterministic parsers with non-deterministic oracles.
TACL, 1:403–414.

Ma, X. and Xia, F. (2014).
Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization.
In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1337–1348, Baltimore, Maryland. Association for Computational Linguistics.

McDonald, R., Pereira, F., Ribarov, K., and Hajič, J. (2005).
Non-projective dependency parsing using spanning tree algorithms.
In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 523–530, Stroudsburg, PA, USA. Association for Computational Linguistics.

McDonald, R., Petrov, S., and Hall, K. (2011).
Multi-source transfer of delexicalized dependency parsers.
In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72, Edinburgh, Scotland, UK. Association for Computational Linguistics.

References II

Rasooli, M. S. and Tetreault, J. (2015).
Yara parser: A fast and accurate dependency parser.
arXiv preprint arXiv:1503.06733.

Zhang, Y. and Barzilay, R. (2015).
Hierarchical low-rank tensors for multilingual transfer parsing.
In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Lisbon, Portugal.