STATISTICAL DEPENDENCY PARSING
2022-23

Greed is Good if Randomized: New Inference for Dependency Parsing

Yuan Zhang*, Tao Lei*, Regina Barzilay, and Tommi Jaakkola

Akshat Gupta, Tejaswi Choppa

Overview

Motivation

- High-order feature parsing is a hard decoding problem
- The combinatorial solution, finding the highest-scoring parse tree(McDonald and pereira)
- State of the art approach: Dual decomposition and MCMC sampling methods
- Use randomised greedy inference algorithm to achieve optimal results

Introduction

- The combinatorial solution, finding the highest-scoring parse tree(McDonald and pereira)
- Randomized greedy algorithms are good across a broad class of np-hard problems
- Utilise multiple random restarts to hill climbing algorithms
- Success of randomised greedy algorithm depends on the number of local maxima
- Hypothesis is that it is true for high order scoring functions as well
- Evaluation is done on CoNLL dependency benchmarks for 14 languages

Background Work

Finding optimal structure in parsing

- NP-hard for non-projective second-order parsing
- Some work includes easy-first parsing, inexact search, partial dynamic
- Programming, and dual decomposition

Greedy approximations for np-hard problems

- Travelling salesman problem
- Vertex cover

Method

Preliminaries

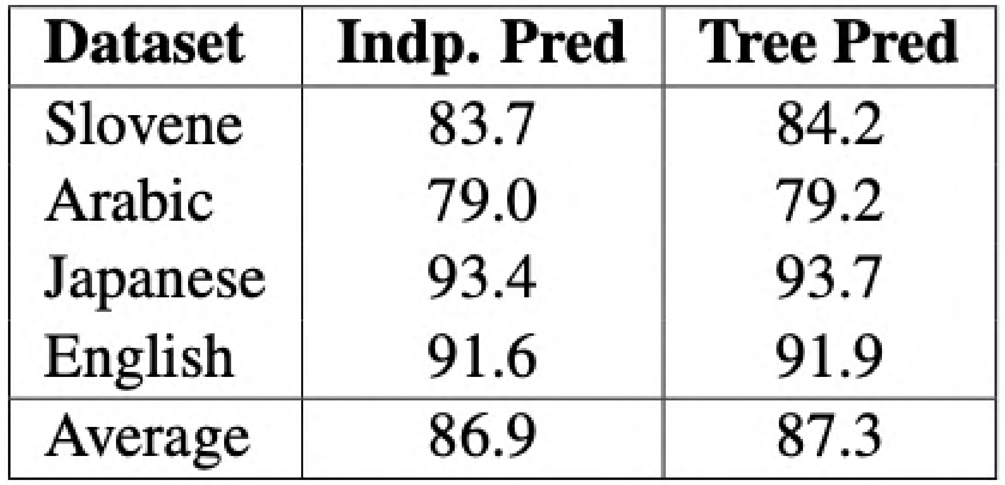
- If x is a sentence and T(x) is set of possible dependency trees for words in x.
- If $y \in T(x)$ is to denote the dependency tree for x
- y(m) is the head of the modifier word indexed by m in the tree y
- T(y,m) is a set of neighboring trees of y
- The scoring function is defined as $S(x,y) = \theta \cdot \varphi(x,y)$, where θ is a vector of parameters and $\varphi(x, y)$ is a sparse feature vector representation of tree y for sentence x
- $\varphi(x, y)$ will include the third-order features as global features
- The max-margin framework is used with a stochastic parameter learning scheme

Try Pitch

Method

Locality and Parsing

- The greedy algorithm performs well because the learned scoring function is local
- It means decisions i.e for the modification of head word is not based on surrounding structure i.e. context dependent
- By contrast the first order tree-based parser with a parser predicting head of each word independently, we obtain degree of locality



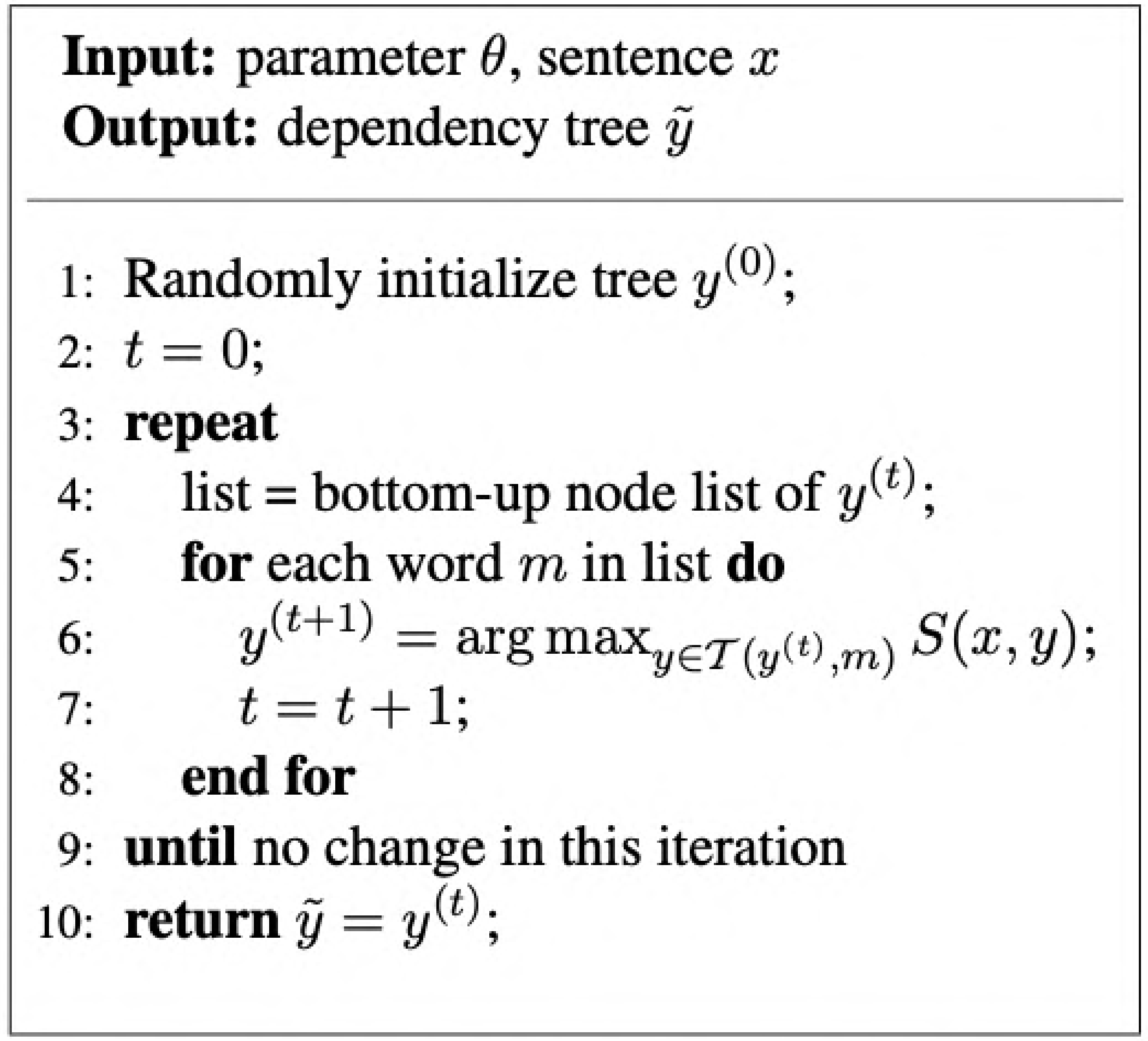
Dataset	Indp. Pred	Tree Pred
Slovene	83.7	84.2
Arabic	79.0	79.2
Japanese	93.4	93.7
English	91.6	91.9
Average	86.9	87.3

Table 1: Head attachment accuracy of a first-order local classifier (left) and a first-order structural prediction model (right). The two types of models are trained using the same set of features.

Method

Hill Climbing with Random Restarts

- Randomly a tree is intilaised
- Greedy algorithms break decoding problems into sequences of local steps
- Updated by changing the head of each modifier word
- Modification is done in a way that results in a bottom-up tree structure
- Randomized hill climbing is cheaper
- All the runs are independent and multiple runs possible parallelly
- The final result is of the optimal highest scoring tree amongst all the multiple runs



```
Input: parameter $\theta$, sentence $x$
Output: dependency tree $\tilde{y}$

1: Randomly initialize tree $y^{(0)}$;
2: $t = 0$;
3: repeat
4:     list = bottom-up node list of $y^{(t)}$;
5:     for each word $m$ in list do
6:         $y^{(t+1)} = \arg\max_{y \in \mathcal{T}(y^{(t)}, m)} S(x, y)$;
7:         $t = t + 1$;
8:     end for
9: until no change in this iteration
10: return $\tilde{y} = y^{(t)}$;
```

Figure 1: A randomized hill-climbing algorithm for dependency parsing.

Method

Algorithm

- The main focus is on decoding problem which means finding the highest scoring tree y ∈ T(x) for each sentence x
- It can be represented as follows:

$$\tilde{y} = \arg\max_{y \in \mathcal{T}(\hat{x}_i)} \{\theta \cdot \phi(\hat{x}_i, y) + \|y - \hat{y}_i\|_1\} \quad \textbf{(train)}$$

$$\tilde{y} = \arg\max_{y \in \mathcal{T}(x)} \{\theta \cdot \phi(x, y)\} \quad \textbf{(test)}$$

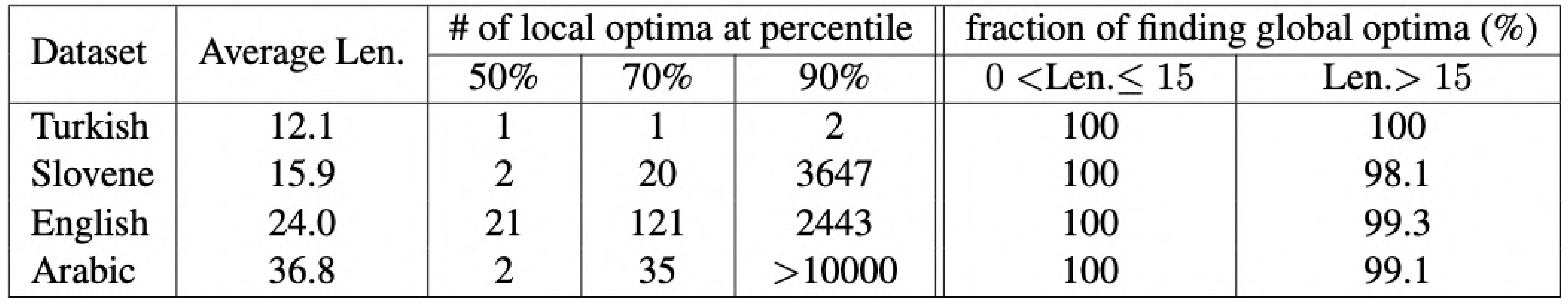
- The feature sets exhibit np-hard problem, the decoder still performs well

Analysis

First-Order Parsing

- A first-order scoring function doesn't guarantee that a greedy algorithm operates correctly
- A first-order arc factor scoring helps to check if the best scoring tree is found
- Count the number of locally optimal solutions for the greedy algorithm and relate this to success rate of algorithm

These reasons can be extended to the analysis of the performance of a high-level parser

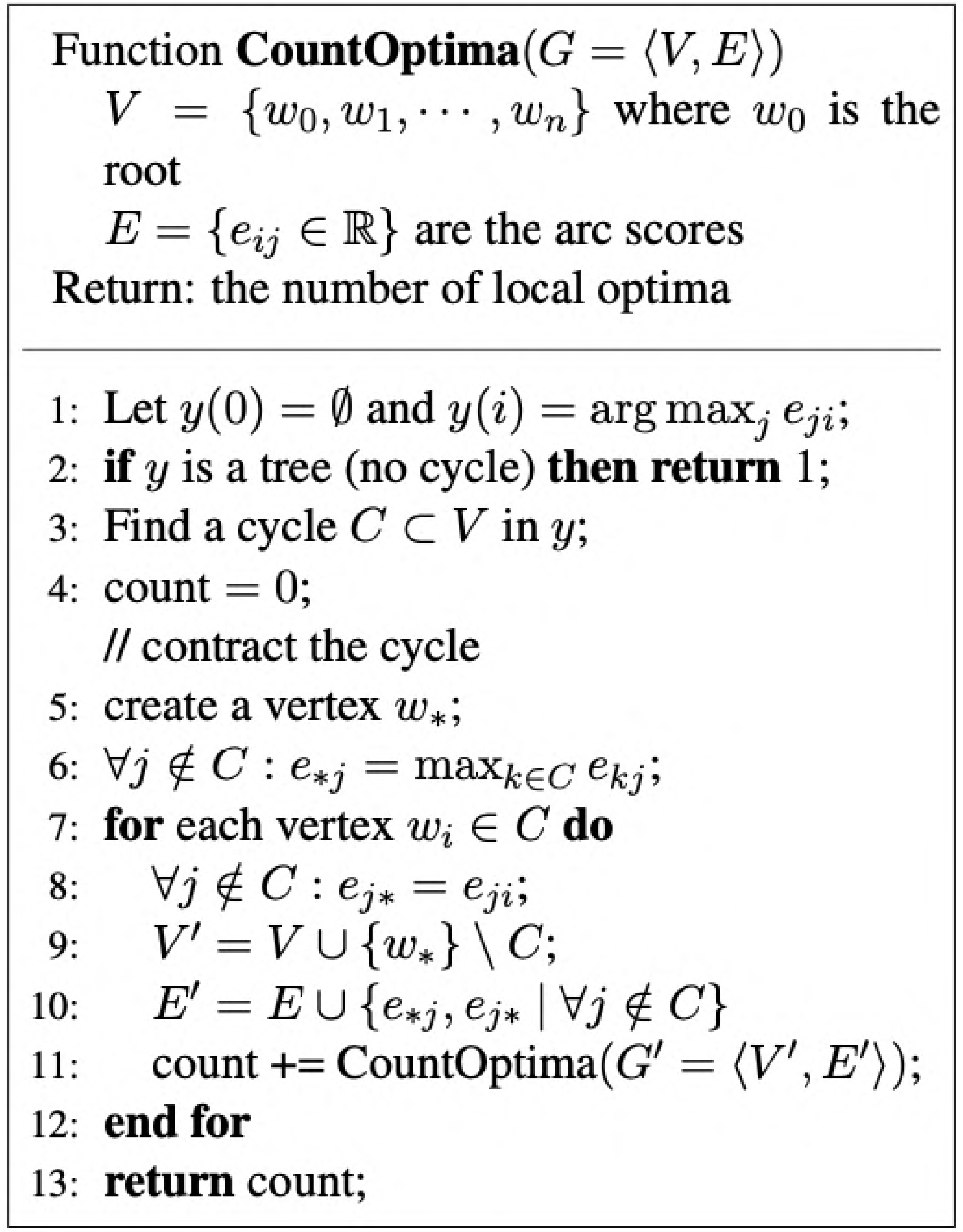


Dataset	Average Len.	# of local optima at percentile			fraction of finding global optima (%)	
		50%	70%	90%	$0 <$ Len. ≤ 15	Len. > 15
Turkish	12.1	1	1	2	100	100
Slovene	15.9	2	20	3647	100	98.1
English	24.0	21	121	2443	100	99.3
Arabic	36.8	2	35	>10000	100	99.1

Analysis

Rechability

- The target tree is reachable from any starting point using only single-arc changes
- Bottom-up order ensures that no cycle is introduced w.r.t terminating unmodified nodes
- Reachability only possible with 'k' changes, 'k'=No. of head differences
- Function CountOptima is a recursive algorithm for counting local optima for a sentence with words(first-order parsing).
- It is similar to Chu-Liu-Edmonds algorithm



```
Function **CountOptima**($G = \langle V, E \rangle$)
    $V = \{w_0, w_1, \cdots, w_n\}$ where $w_0$ is the root
    $E = \{e_{ij} \in \mathbb{R}\}$ are the arc scores
Return: the number of local optima

1:  Let $y(0) = \emptyset$ and $y(i) = \arg\max_j e_{ji}$;
2:  **if** $y$ is a tree (no cycle) **then return** 1;
3:  Find a cycle $C \subset V$ in $y$;
4:  count $= 0$;
    // contract the cycle
5:  create a vertex $w_*$;
6:  $\forall j \notin C : e_{*j} = \max_{k \in C} e_{kj}$;
7:  **for** each vertex $w_i \in C$ **do**
8:      $\forall j \notin C : e_{j*} = e_{ji}$;
9:      $V' = V \cup \{w_*\} \setminus C$;
10:     $E' = E \cup \{e_{*j}, e_{j*} \mid \forall j \notin C\}$
11:     count += CountOptima($G' = \langle V', E' \rangle$);
12: **end for**
13: **return** count;
```

Analysis

Locally Optimal Trees

- Greedy algorithms are prone to be stuck at locally optimal solutions.

- Decoding with learned scoring functions has only a few local optima

- Find the highest scoring among all local optima

- 2 approaches:
  - Worst-Case Analysis
  - Average-Case Analysis

- For worst case: A tighter bound is setup on the number of optima according to Cayley's formula

- It can be almost 2^n-1 locally optimal trees for n words

Analysis

Average Case Analysis and Optimal Decoding

- Unlike the worst-case analysis, count the actual number of local optima per sentence
- In the average case, decoding using learned scoring functions is easier
- We perform optimal decoding to check if the greedy algorithm performs well in first-order parsing which means fewer number of restarts are needed.

Analysis

High Order Parsing

- Exact decoding of high-order features is hard.
- In 97.8% of sentences, Hill Climbing obtains the same score as Dual Decomposition
- In 1.3% Hill Climbing finds a higher scoring tree
- In 0.9% Dual Decomposition results in a better tree
- Average rate of certificates for Dual Decomposition was 92%
- In over 99% of sentences Hill Climbing reaches the same optimum.

Experiments

Dataset

- CoNLL dependency treebanks for 14 languages are used
- POS tags and the morphological information used from the corpus

Evaluation Metrics:

- UAS (Unlabeled Attachment Score)

Experiments

Baselines

- Compared with TurboParser(Martin et. al. 2013) and Sampling-based Parser(Zhang et al. 2014).

Additional Comparisons

- Martins et al. (2011), Koo et. al. (2010), Zhang et. al. (2013), and tensor-based parser(Lei et. al. 2014)

Features

- Same features as prior work in Zhang et. al. 2014, third-order templates, for Global features
- Right-branching, coordination, PP attachement, span length, neighbors, valency, and non-projective arcs features

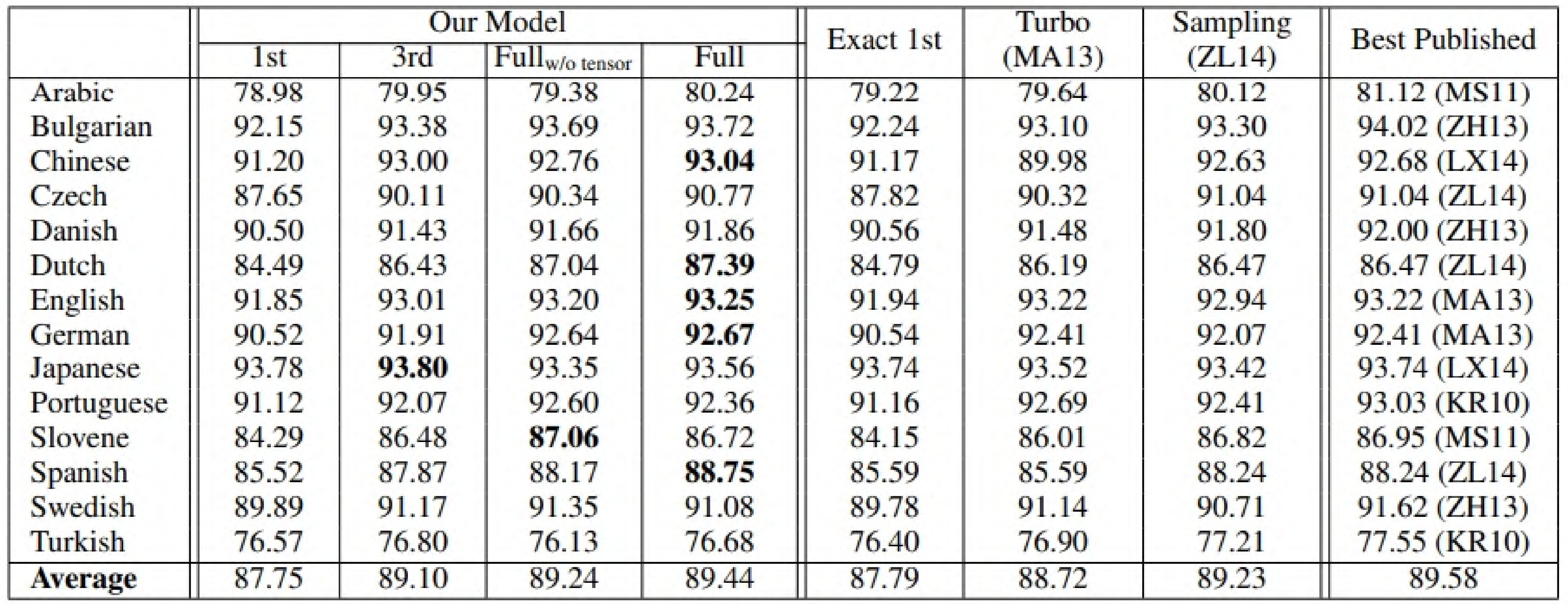
Experiments

Implementation Details

- Training using passive-aggressive online learning algorithm(MIRA) and parameter averaging(crammer et al. 2006)
- The adaptive strategy got a hill-climbing algorithm
- For a given sentence parallelly run for k=300 consecutive restarts
- Start sampling is done from first-order distribution
- First-order and third-order models trained for 10 and 20 epochs for all languages

Results

- **Comparison with the Baselines**
- **Impact of High Order Parsing**



	Our Model				Exact 1st	Turbo (MA13)	Sampling (ZL14)	Best Published
	1st	3rd	Full$_{\text{w/o tensor}}$	Full				
Arabic	78.98	79.95	79.38	80.24	79.22	79.64	80.12	81.12 (MS11)
Bulgarian	92.15	93.38	93.69	93.72	92.24	93.10	93.30	94.02 (ZH13)
Chinese	91.20	93.00	92.76	**93.04**	91.17	89.98	92.63	92.68 (LX14)
Czech	87.65	90.11	90.34	90.77	87.82	90.32	91.04	91.04 (ZL14)
Danish	90.50	91.43	91.66	91.86	90.56	91.48	91.80	92.00 (ZH13)
Dutch	84.49	86.43	87.04	**87.39**	84.79	86.19	86.47	86.47 (ZL14)
English	91.85	93.01	93.20	**93.25**	91.94	93.22	92.94	93.22 (MA13)
German	90.52	91.91	92.64	**92.67**	90.54	92.41	92.07	92.41 (MA13)
Japanese	93.78	**93.80**	93.35	93.56	93.74	93.52	93.42	93.74 (LX14)
Portuguese	91.12	92.07	92.60	92.36	91.16	92.69	92.41	93.03 (KR10)
Slovene	84.29	86.48	**87.06**	86.72	84.15	86.01	86.82	86.95 (MS11)
Spanish	85.52	87.87	88.17	**88.75**	85.59	85.59	88.24	88.24 (ZL14)
Swedish	89.89	91.17	91.35	91.08	89.78	91.14	90.71	91.62 (ZH13)
Turkish	76.57	76.80	76.13	76.68	76.40	76.90	77.21	77.55 (KR10)
Average	87.75	89.10	89.24	89.44	87.79	88.72	89.23	89.58

Table 4: Results of our model and several state-of-the-art systems. "Best Published UAS" includes the most accurate parsers among Martins et al. (2011), Martins et al. (2013), Koo et al. (2010), Zhang et al. (2013), Lei et al. (2014) and Zhang et al. (2014). For the third-order model, we use the feature set of TurboParser (Martins et al., 2013). The full model combines features of our sampling-based parser (Zhang et al., 2014) and tensor features (Lei et al., 2014).

Results

Comparison with the Baselines

- On average across 14 languages full model with tensor component outperforms both turbo parser and sampling-based parser
- 89.10 vs 88.72 for turbo parser

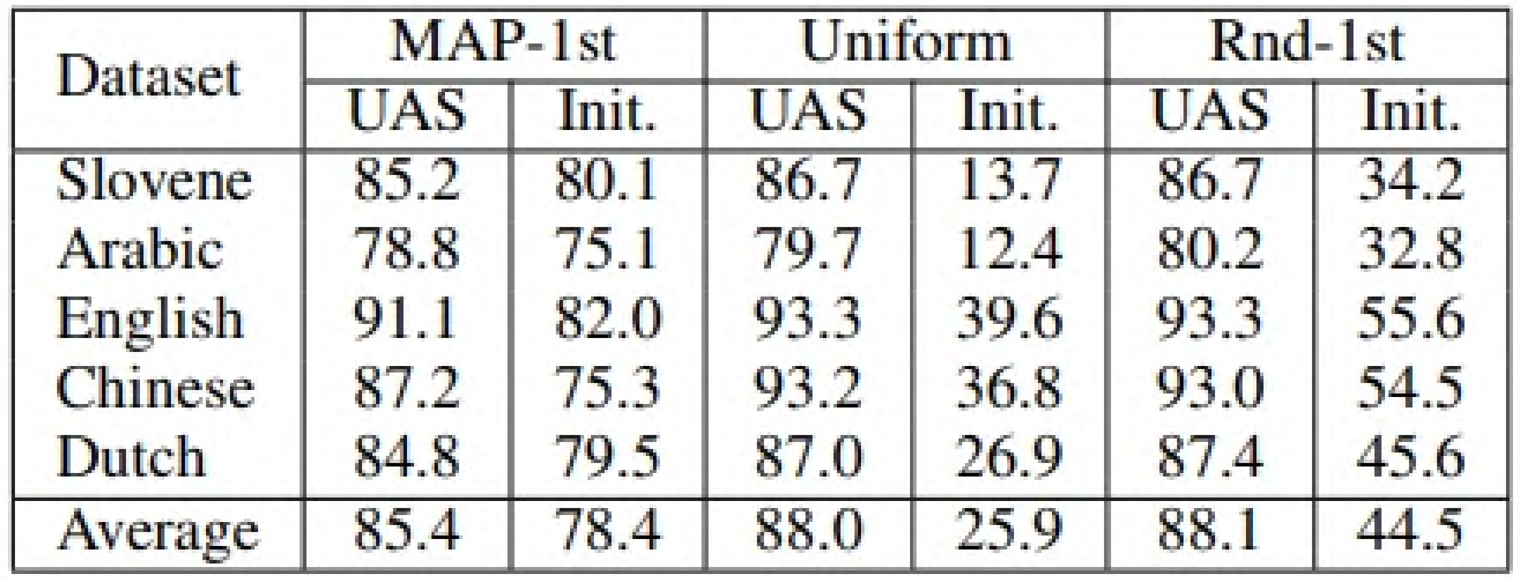
Impact of High Order Parsing

- Accuracy improves when high-order features are added
- Based on sentence length, high-order features are useful when parsing longer sentences

Results

Impact of Initialization and Restarts

- MAP estimate of the first-order score from the model
- Random trees sampled using uniform distribution
- Rnd-1st: random trees sampled from first order distribution
- The accuracy of initial trees varies greatly ranging from 78.4% for the MAP estimate to 25.9% and 44.5% from the latter randomized strategies.
- While the first-order MAP estimate gives the best initial guess this demonstrates the importance of restarts in contrast to randomized strategies.
- MAP performs only a single run of hill-climbing



Dataset	MAP-1st		Uniform		Rnd-1st	
	UAS	Init.	UAS	Init.	UAS	Init.
Slovene	85.2	80.1	86.7	13.7	86.7	34.2
Arabic	78.8	75.1	79.7	12.4	80.2	32.8
English	91.1	82.0	93.3	39.6	93.3	55.6
Chinese	87.2	75.3	93.2	36.8	93.0	54.5
Dutch	84.8	79.5	87.0	26.9	87.4	45.6
Average	85.4	78.4	88.0	25.9	88.1	44.5

Try Pitch

Results

Convergence Properties

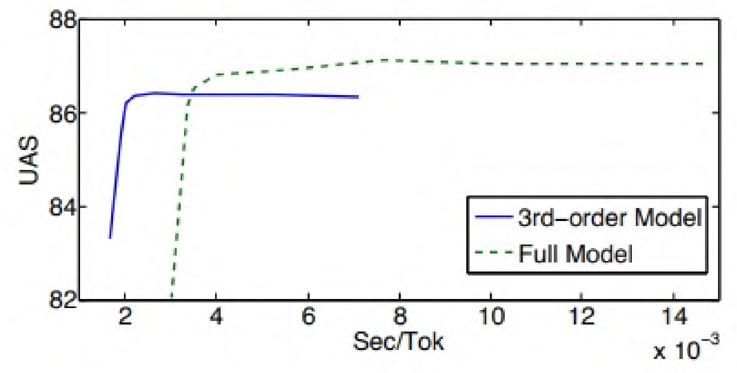
- It shows a score of trees retrieved by our full model concerning the number of restarts for short and long sentences in English and Slovene

- To compare, normalized the score for this sentence after 3000 restarts, most sentences converge in 300 restarts more than 98% sentence converge within 300 restarts

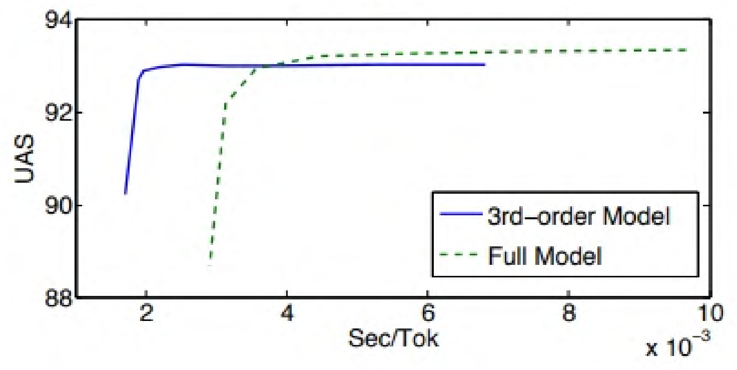
Decoding Speed

- Several restarts impact the parsing accuracy, and performance for speed can be traded.

- The model delivers parsing time comparable to other SOTA graph-based systems and sampling based parsing



(a) Slovene

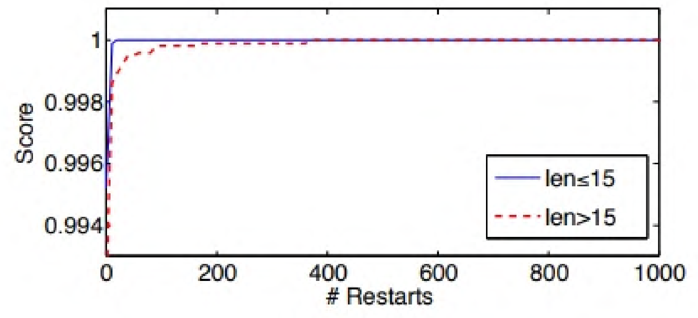


(b) English

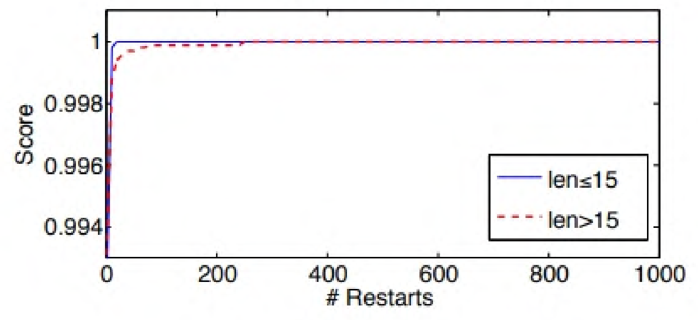
Figure 6: Trade-off between performance and speed on Slovene and English datasets. The graph shows the accuracy as a function of decoding speed measured in second per token. Variations in decoding speed is achieved by changing the number of restarts.

	Length $\leq$ 15	Length $>$ 15
Slovene	100	98.11
English	100	99.12

Table 6: Fractions (%) of the sentences that find the best solution among 3,000 restarts within the first 300 restarts.



(a) Slovene



(b) English

Conclusion

- A randomized greedy algorithm for inference suffices to deliver State of the Art Performance
- Effectiveness is contingent on gaining a small number of local optima in the scoring function
- By algorithmically counting the number of locally optimal solutions in the context of first-order parsing we show that this number is indeed quite small.
- Decoding algorithm of the greedy method surpasses the dual decomposition in second-order parsing
- With third order and global features outperforms the state-of-the-art parsers when evaluated on 14 languages of nonprojective coNLL datasets.