

Using a maximum entropy-based tagger to improve a very fast vine parser

Anders Søgaard and Jonas Kuhn

The 11th International Conference on Parsing Technology (IWPT 2009)

October 9 2009

Why vine parsing?

Our baseline system

Experiments

Results

Does vine parsing scale?

Why vine parsing?

- Vine parsers are *very* fast, i.e. low linear and finite-state.
 - A local classifier (Galley and Manning, 2009) runs in $\mathcal{O}(|G|n^2)$ consider $n + 1$ alternative heads for n dependents.
 - A local *vine* classifier with dependency lengths at most k runs in $\mathcal{O}(|G|(2k + 1)n)$.
- While vine parsers such as Eisner and Smith (2005) and Dreyer et al. (2006) have been non-competitive, constraints on dependency length improve accuracy under *some* conditions:

Why vine parsing?

- Vine parsers are *very* fast, i.e. low linear and finite-state.
 - A local classifier (Galley and Manning, 2009) runs in $\mathcal{O}(|G|n^2)$ consider $n + 1$ alternative heads for n dependents.
 - A local *vine* classifier with dependency lengths at most k runs in $\mathcal{O}(|G|(2k + 1)n)$.
- While vine parsers such as Eisner and Smith (2005) and Dreyer et al. (2006) have been non-competitive, constraints on dependency length improve accuracy under *some* conditions:
 - Projective, MLE (Eisner and Smith, 2005): ✓ (\sim 0–10 UAS)

Why vine parsing?

- Vine parsers are *very* fast, i.e. low linear and finite-state.
 - A local classifier (Galley and Manning, 2009) runs in $\mathcal{O}(|G|n^2)$ consider $n + 1$ alternative heads for n dependents.
 - A local *vine* classifier with dependency lengths at most k runs in $\mathcal{O}(|G|(2k + 1)n)$.
- While vine parsers such as Eisner and Smith (2005) and Dreyer et al. (2006) have been non-competitive, constraints on dependency length improve accuracy under *some* conditions:
 - Projective, MLE (Eisner and Smith, 2005): ✓ (\sim 0–10 UAS)
 - Non-projective, MLE (this work): ✓ (\sim 5-15 UAS)

Why vine parsing?

- Vine parsers are *very* fast, i.e. low linear and finite-state.
 - A local classifier (Galley and Manning, 2009) runs in $\mathcal{O}(|G|n^2)$ consider $n + 1$ alternative heads for n dependents.
 - A local *vine* classifier with dependency lengths at most k runs in $\mathcal{O}(|G|(2k + 1)n)$.
- While vine parsers such as Eisner and Smith (2005) and Dreyer et al. (2006) have been non-competitive, constraints on dependency length improve accuracy under *some* conditions:
 - Projective, MLE (Eisner and Smith, 2005): ✓ (\sim 0–10 UAS)
 - Non-projective, MLE (this work): ✓ (\sim 5-15 UAS)
 - Projective, MIRA (other work): (\sim -0.5–0.5 UAS)
 - Non-projective, MIRA (other work): (\sim -0.5–0.5 UAS)

Why vine parsing?

- Vine parsers are *very* fast, i.e. low linear and finite-state.
 - A local classifier (Galley and Manning, 2009) runs in $\mathcal{O}(|G|n^2)$ consider $n + 1$ alternative heads for n dependents.
 - A local *vine* classifier with dependency lengths at most k runs in $\mathcal{O}(|G|(2k + 1)n)$.
- While vine parsers such as Eisner and Smith (2005) and Dreyer et al. (2006) have been non-competitive, constraints on dependency length improve accuracy under *some* conditions:
 - Projective, MLE (Eisner and Smith, 2005): ✓ (\sim 0–10 UAS)
 - Non-projective, MLE (this work): ✓ (\sim 5-15 UAS)
 - Projective, MIRA (other work): (\sim -0.5–0.5 UAS)
 - Non-projective, MIRA (other work): (\sim -0.5–0.5 UAS)
- **Goal:** To improve a non-projective MLE-based vine parser without sacrificing speed.

Our baseline system

- MLE-informed Chu-Liu-Edmonds without cycle contraction, i.e. a local classifier (Galley and Manning, 2009).
- Hard and soft constraints on dependency length optimized on 7.5k tuning sections. Soft constraints:
 - POS/CPOS as head, and
 - POS/CPOS as dependent.
- The parser only does *unlabeled* dependency parsing of *POS sequences*, i.e. like Eisner and Smith (2005).

Experiments

Languages:

Arabic	Semitic	CONLL-X
Czech	Slavic	CONLL-X
Dutch	Germanic	CONLL-X
Italian	Romance	EVALITA 2007
Japanese	Japonic-Ryukyuan	CONLL-X
Turkish	Uralic	CONLL-X

- A maximum-entropy-based tagger (Ratnaparkhi, 1998) is used to find ROOT, LEFT and RIGHT dependencies.
- **Idea:** Use high-precision tags as *hard* constraints.

Results

	Arabic	Czech	Dutch	Italian	Japanese	Turkish
MaltParser	66.22	67.78	65.03	75.48	89.13	68.94
MLE	56.41	55.30	59.21	69.93	78.11	57.32
Vine	67.99	66.70	65.98	75.50	83.15	68.53
Vine+Ro	68.68	66.65	66.21	78.06	83.82	68.45
Vine+Ro+L	69.68	68.14	68.05	77.14	84.64	68.37
Vine+Ri	68.50	67.38	68.18	78.55	84.17	69.87
Vine+Ro+Ri	69.20	67.32	68.40	78.29	84.78	69.79
Vine+Ro+L+Ri	70.28	68.70	70.06	77.26	85.45	69.74

Does vine parsing scale?

- Experiment 1: using a (non-competitive) tagger to predict ROOTs, LEFTs and RIGHTs *obviously* hurt the performance of state-of-the-art parsers.
- Experiment 2: hard constraints on “CPOS as dependent” on a MIRA-informed (first-order) projective parser (McDonald et al., 2005). Results:

	BL	T=1	T=5	G+T=5	Δ	Off
Arabic	78.26	77.41	77.98	77.88	-0.28	79.34
Slovene	81.83	81.14	81.85	80.72	0.02	83.17
Turkish	74.83	74.11	74.89	74.75	0.06	74.67