Vine Parsing and Minimum Risk Reranking for Speed and Precision*

Markus Dreyer, David A. Smith, and **Noah A. Smith**
Department of Computer Science / Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD 21218 USA
{markus,{d,n}asmith}@cs.jhu.edu

Abstract

We describe our entry in the CoNLL-X shared task. The system consists of three phases: a probabilistic vine parser (Eisner and N. Smith, 2005) that produces unlabeled dependency trees, a probabilistic relation-labeling model, and a discriminative minimum risk reranker (D. Smith and Eisner, 2006). The system is designed for fast training and decoding and for high precision. We describe sources of cross-lingual error and ways to ameliorate them. We then provide a detailed error analysis of parses produced for sentences in German (much training data) and Arabic (little training data).

1 Introduction

Standard state-of-the-art parsing systems (e.g., Charniak and Johnson, 2005) typically involve two passes. First, a **parser** produces a list of the most likely n parse trees under a generative, probabilistic model (usually some flavor of PCFG). A discriminative **reranker** then chooses among trees in this list by using an extended feature set (Collins, 2000). This paradigm has many advantages: PCFGs are fast to train, can be very robust, and perform better as more data is made available; and rerankers train quickly (compared to discriminative models), require few parameters, and permit arbitrary features.

We describe such a system for **dependency** parsing. Our shared task entry is a preliminary system developed in only 3 person-weeks, and its accuracy is typically one s.d. below the average across systems and 10–20 points below the best system. On the positive side, its decoding algorithms have guaranteed $O(n)$ runtime, and training takes only a couple of hours. Having designed primarily for **speed** and **robustness**, we sacrifice accuracy. Better **estimation**, reranking on larger datasets, and more fine-grained parsing constraints are expected to boost accuracy while maintaining speed.

2 Notation

Let a sentence $\mathbf{x} = \langle x_1, x_2, ..., x_n \rangle$, where each x_i is a tuple containing a part-of-speech tag t_i and a word w_i, and possibly more information.[1] x_0 is a special wall symbol, \$, on the left. A dependency tree $\mathbf{y}$ is defined by three functions: $\mathbf{y}_{left}$ and $\mathbf{y}_{right}$ (both $\{0, 1, 2, ..., n\} \rightarrow 2^{\{1,2,...,n\}}$) that map each word to its sets of left and right dependents, respectively, and $\mathbf{y}_{label} : \{1, 2, ..., n\} \rightarrow \mathcal{D}$, which labels the relationship between word i and its parent from label set $\mathcal{D}$.

In this work, the graph is constrained to be a *projective* tree rooted at \$: each word except \$ has a single parent, and there are no cycles or crossing dependencies. Using a simple dynamic program to find the minimum-error projective parse, we find that assuming projectivity need not harm accuracy very much (Tab. 1, col. 3).

3 Unlabeled Parsing

The first component of our system is an unlabeled parser that, given a sentence, finds the U best unlabeled trees under a probabilistic model using a bottom-up dynamic programming algorithm.[2] The model is a probabilistic head automaton grammar (Alshawi, 1996) that assumes conditional indepen-

*This work was supported by NSF ITR grant IIS-0313193, an NSF fellowship to the second author, and a Fannie and John Hertz Foundation fellowship to the third author. The views expressed are not necessarily endorsed by the sponsors. We thank Charles Schafer, Keith Hall, Jason Eisner, and Sanjeev Khudanpur for helpful conversations.

[1] We used words and fine tags in our parser and labeler, with coarse tags in one backoff model. Other features are used in reranking; we never used the given morphological features or the "projective" annotations offered in the training data.

[2] The execution model we use is best-first, exhaustive search, as described in Eisner et al. (2004). All of our dynamic programming algorithms are implemented concisely in the Dyna language.

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 201–205, New York City, June 2006.

	B_ℓ	B_r	projective oracle	(B_ℓ, B_r)-vine oracle	20-best unlabeled oracle	1-best unlabeled	20×50-best labeled oracle	1×1-best labeled	**reranked** (labeled)	(unlabeled)	(non-$ unl. recall)	(non-$ unl. precision)	
Arabic	10	4	**99.8**	**90.7**	**71.5**	68.1	68.7	59.7	52.0	53.4	68.5	63.4	76.0
Bulgarian	5	4	**99.6**	**90.7**	86.4	80.1	80.5	85.1	73.0	74.8	82.0	74.3	86.3
Chinese	4	4	**100.0**	**93.1**	89.9	79.4	*77.7*	88.6	72.6	*71.6*	77.6	61.4	80.8
Czech	6	4	**97.8**	**90.5**	79.2	70.3	71.5	72.8	58.1	60.5	70.7	64.8	75.7
Danish	5	4	**99.2**	**91.4**	**84.6**	77.7	78.6	79.3	65.5	66.6	77.5	71.4	83.4
Dutch	6	5	**94.6**	**88.3**	77.5	67.9	68.8	73.6	59.4	61.6	68.3	60.4	73.0
German	8	7	**98.8**	**90.9**	83.4	75.5	76.2	82.3	70.1	71.0	77.0	70.2	82.9
Japanese	4	1	**99.2**	**92.2**	90.7	86.3	*85.1*	89.4	81.6	82.9	86.0	68.5	91.5
Portuguese	5	5	**98.8**	**91.5**	85.9	81.4	82.5	83.7	73.4	75.3	82.4	76.2	87.0
Slovene	6	4	**98.5**	**91.7**	**80.5**	72.0	73.3	72.8	57.5	58.7	72.9	66.3	78.5
Spanish	5	6	**100.0**	**91.2**	77.3	71.5	72.6	74.9	66.2	67.6	72.9	69.3	80.7
Swedish	4	5	**99.7**	**94.0**	**87.5**	79.3	79.6	81.0	65.5	67.6	79.5	72.6	83.3
Turkish	6	1	**98.6**	**89.5**	**73.0**	61.0	61.8	64.4	44.9	46.1	60.5	48.5	61.6
						parser	reranker	labeler			reranker		
	1	2	3	4	5	6	7	8	9	10	11	12	13

Table 1: Parameters and performance on test data. B_ℓ and B_r were chosen to retain 90% of dependencies in training data. We show oracle, 1-best, and reranked performance on the test set at different stages of the system. Boldface marks oracle performance that, given perfect downstream modules, would supercede the best system. Italics mark the few cases where the reranker increased error rate. Columns 8–10 show labeled accuracy; column 10 gives the final shared task evaluation scores.

dence between the left yield and the right yield of a given head, given the head (Eisner, 1997).[3] The best known parsing algorithm for such a model is $O(n^3)$ (Eisner and Satta, 1999). The U-best list is generated using Algorithm 3 of Huang and Chiang (2005).

3.1 Vine parsing (dependency length bounds)

Following Eisner and N. Smith (2005), we also impose a bound on the string distance between every child and its parent, with the exception of nodes attaching to \$. Bounds of this kind are intended to improve precision of non-\$ attachments, perhaps sacrificing recall. Fixing bound B_ℓ, no left dependency may exist between child x_i and parent x_j such that $j - i > B_\ell$ (similarly for right dependencies and B_r). As a result, edge-factored parsing runtime is reduced from $O(n^3)$ to $O(n(B_\ell^2 + B_r^2))$. For each language, we choose B_ℓ (B_r) to be the minimum value that will allow recovery of 90% of the left (right) dependencies in the training corpus (Tab. 1, cols. 1, 2, and 4). In order to match the training data to the parsing model, we re-attach disallowed long dependencies to \$ during training.

3.2 Estimation

The probability model predicts, for each parent word x_j, $\{x_i\}_{i \in \mathbf{y}_{left}(j)}$ and $\{x_i\}_{i \in \mathbf{y}_{right}(j)}$. An advantage of head automaton grammars is that, for a given parent node x_j, the children on the same side, $\mathbf{y}_{left}(j)$,

[3] To empirically test this assumption across languages, we measured the mutual information between different features of $\mathbf{y}_{left}(j)$ and $\mathbf{y}_{right}(j)$, given x_j. (Mutual information is a statistic that equals zero iff conditional independence holds.) A detailed discussion, while interesting, is omitted for space, but we highlight some of our findings. First, unsurprisingly, the split-head assumption appears to be less valid for languages with freer word order (Czech, Slovene, German) and more valid for more fixed-order languages (Chinese, Turkish, Arabic) or corpora (Japanese). The children of verbs and conjunctions are the most frequent violators. The mutual information between the sequence of dependency labels on the left and on the right, given the head's (coarse) tag, only once exceeded 1 bit (Slovene).

for example, can depend on each other (cf. McDonald et al., 2005). Child nodes in our model are generated outward, conditional on the parent and the most recent same-side sibling (MRSSS). This increases our parser's theoretical runtime to $O(n(B_\ell^3 + B_r^3))$, which we found was quite manageable.

Let $\text{par}_{\mathbf{y}} : \{1, 2, ..., n\} \rightarrow \{0, 1, ..., n\}$ map each node to its parent in $\mathbf{y}$. Let $\text{pred}_{\mathbf{y}} : \{1, 2, ..., n\} \rightarrow \{\emptyset, 1, 2, ..., n\}$ map each node to the MRSSS in $\mathbf{y}$ if it exists and $\emptyset$ otherwise. Let $\Delta_i = |i - j|$ if j is i's parent. Our (probability-deficient) model defines

$$p(\mathbf{y}) = \prod_{j=1}^{n} \left(\prod_{i \in \mathbf{y}_{left}(j)} p(x_i, \Delta_i \mid x_j, x_{\text{pred}_{\mathbf{y}}(i)}, \textit{left}) \right) \times p(\text{STOP} \mid x_j, x_{\min_{\mathbf{y}_{left}(j)} j}, \textit{left}) \times \left(\prod_{i \in \mathbf{y}_{right}(j)} p(x_i, \Delta_i \mid x_j, \text{pred}_{\mathbf{y}}(i), \textit{right}) \right) \times p(\text{STOP} \mid x_j, x_{\max_{\mathbf{y}_{right}(j)} j}, \textit{right}) \quad (1)$$

Due to the familiar sparse data problem, a maximum likelihood estimate for the ps in Eq. 1 performs very badly (2–23% unlabeled accuracy). Good statistical parsers smooth those distributions by making **conditional independence assumptions** among variables, including backoff and factorization. Arguably the choice of assumptions made (or interpolated among) is central to the success of many existing parsers.

Noting that (a) there are exponentially many such options, and (b) the best-performing independence assumptions will almost certainly vary by language, we use a mixture among 8 such models. The same mixture is used for all languages. The models were not chosen with particular care,[4] and the mixture is *not trained*—the coefficients are fixed at uniform, with a unigram coarse-tag model for backoff. In principle, this mixture should be trained (e.g., to maximize likelihood or minimize error on a development dataset).

The performance of our unlabeled model's top choice and the top-20 oracle are shown in Tab. 1, cols. 5–6. In 5 languages (boldface), perfect labeling and reranking at this stage would have resulted in performance superior to the language's best labeled system, although the oracle is never on par with the best *unlabeled* performance.

4 Labeling

The second component of our system is a labeling model that *independently* selects a label from $\mathcal{D}$ for each parent/child pair in a tree. Given the U best unlabeled trees for a sentence, the labeler produces the L best *labeled* trees for each unlabeled one. The computation involves an $O(|\mathcal{D}|n)$ dynamic programming algorithm, the output of which is passed to Huang and Chiang's (2005) algorithm to generate the L-best list.

We separate the labeler from the parser for two reasons: speed and candidate diversity. In principle the vine parser could jointly predict dependency labels along with structures, but parsing runtime would increase by at least a factor of $|\mathcal{D}|$. The two stage process also forces diversity in the candidate list (20 structures with 50 labelings each); the 1,000-best list of *jointly*-decoded parses often contained many (bad) relabelings of the same tree.

In retrospect, assuming independence among dependency labels damages performance substantially for some languages (Turkish, Czech, Swedish, Danish, Slovene, and Arabic); note the often large drop in oracle performance between Tab. 1, cols. 5 and 8. This assumption is necessary in our framework, because the $O(|\mathcal{D}|^{M+1}n)$ runtime of decoding with an Mth-order Markov model of labels[5] is in general prohibitive—in some cases $|\mathcal{D}| > 80$. Pruning and search heuristics might ameliorate runtime.

If x_i is a child of x_j in direction D, and x_{pred} is the MRSSS (possibly $\emptyset$), where $\Delta_i = |i - j|$, we estimate $p(\ell, x_i, x_j, x_{pred}, \Delta_i \mid D)$ by a mixture (untrained, as in the parser) of four backed-off, factored estimates.

After parsing and labeling, we have for each sentence a list of $U \times L$ candidates. Both the oracle performance of the best candidate in the (20×50)-best list and the performance of the top candidate are shown in Tab. 1, cols. 8–9. It should be clear from the drop in both oracle and 1-best accuracy that our labeling model is a major source of error.

[4] Our infrastructure provides a concise, interpreted language for expressing the models to be mixed, so large-scale combination and comparison are possible.

[5] We tested first-order Markov models that conditioned on parent or MRSSS dependency labels.

5 Reranking

We train a log-linear model combining many feature scores (see below), including the log-probabilities from the parser and labeler. Training minimizes the expected error under the model; we use deterministic annealing to smooth the error surface and avoid local minima (Rose, 1998; D. Smith and Eisner, 2006).

We reserved 200 sentences in each language for training the reranker, plus 200 for choosing among rerankers trained on different feature sets and different $(U \times L)$-best lists.[6]

Features Our reranking features predict tags, labels, lemmata, suffixes and other information given all or some of the following non-local conditioning context: bigrams and trigrams of tags or dependency labels; parent and grandparent dependency labels; subcategorization frames (in terms of tags or dependency labels); the occurrence of certain tags between head and child; surface features like the lemma[7] and the 3-character suffix. In some cases the children of a node are considered all together, and in other cases left and right are separated.

The highest-ranked features during training, for all languages, are the parser and labeler probabilities, followed by $p(\Delta_i \mid t_{parent})$, $p(direction \mid t_{parent})$, $p(label \mid label_{pred}, label_{succ}, subcat)$, and $p(\mathrm{coarse}(t) \mid D, \mathrm{coarse}(t_{parent}), \mathrm{Betw})$, where Betw is TRUE iff an instance of the coarse tag type with the highest mutual information between its left and right children (usually verb) is between the child and its head.

Feature and Model Selection For training speed and to avoid overfitting, only a subset of the above features are used in reranking. Subsets of different sizes (10, 20, and 40, plus "all") are identified for each language using two naïve feature-selection heuristics based on independent performance of features. The feature subset with the highest accuracy on the 200 heldout sentences is selected.

[6] In training our system, we made a serious mistake in training the reranker on only 200 sentences. As a result, our pretesting estimates of performance (on data reserved for model selection) were very bad. The reranker, depending on condition, had only 2–20 times as many examples as it had parameters to estimate, with overfitting as the result.

[7] The first 4 characters of a word are used where the lemma is not available.

Performance Accuracy of the top parses after reranking is shown in Tab. 1, cols. 10–11. Reranking almost always gave some improvement over 1-best parsing.[8] Because of the vine assumption and the preprocessing step that re-attaches all distant children to $, our parser learns to over-attach to $, treating $-attachment as a default/agnostic choice. For many applications a local, incomplete parse may be sufficiently useful, so we also measured non-$ unlabeled precision and recall (Tab. 1, cols. 12–13); our parser has $>$ 80% precision on 8 of the languages. We also applied reranking (with unlabeled features) to the 20-best unlabeled parse lists (col. 7).

6 Error Analysis: German

The plurality of errors (38%) in German were erroneous $ attachments. For ROOT dependency labels, we have a high recall (92.7%), but low precision (72.4%), due most likely to the dependency length bounds. Among the most frequent tags, our system has most trouble finding the correct heads of prepositions (APPR), adverbs (ADV), finite auxiliary verbs (VAFIN), and conjunctions (KON), and finding the correct dependency labels for prepositions, nouns, and finite auxiliary verbs.

The German conjunction *und* is the single word with the most frequent head attachment errors. In many of these cases, our system does not learn the subtle difference between enumerations that are headed by *A* in *A und B*, with two children *und* and *B* on the right, and those headed by *B*, with *und* and *A* as children on its left.

Unlike in some languages, our labeled oracle accuracy is nearly as good as our unlabeled oracle accuracy (Tab. 1, cols. 8, 5). Among the ten most frequent dependency labels, our system has the most difficulty with accusative objects (OA), genitive attributes (AG), and postnominal modifiers (MNR). Accusative objects are often mistagged as subject (SB), noun kernel modifiers (NK), or AG. About 32% of the postnominal modifier relations (*ein Platz in der Geschichte*, 'a place in history') are labeled as modifiers (*in die Stadt fliegen*, 'fly into the city'). Genitive attributes are often tagged as NK since both are frequently realized as nouns.

[8] The exception is Chinese, where the training set for reranking is especially small (see fn. 6).

7 Error Analysis: Arabic

As with German, the greatest portion of Arabic errors (40%) involved attachments to $. Prepositions are consistently attached too low and accounted for 26% of errors. For example, if a form in construct (*idafa*) governed both a following noun phrase and a prepositional phrase, the preposition usually attaches to the lower noun phrase. Similarly, prepositions usually attach to nearby noun phrases when they should attach to verbs farther to the left.

We see a more serious casualty of the dependency length bounds with conjunctions. In ground truth test data, 23 conjunctions are attached to $ and 141 to non-$ to using the COORD relation, whereas 100 conjunctions are attached to $ and 67 to non-$ using the AUXY relation. Our system overgeneralizes and attaches 84% of COORD and 71% of AUXY relations to $. Overall, conjunctions account for 15% of our errors. The AUXY relation is defined as "auxiliary (in compound expressions of various kinds)"; in the data, it seems to be often used for *waw*-consecutive or paratactic chaining of narrative clauses. If the conjunction *wa* ('and') begins a sentence, then that conjunction is tagged in ground truth as attaching to $; if the conjunction appears in the middle of the sentence, it may or may not be attached to $.

Noun attachments exhibit a more subtle problem. The direction of system attachments is biased more strongly to the left than is the case for the true data. In canonical order, Arabic nouns do generally attach on the right: subjects and objects follow the verb; in construct, the governed noun follows its governor. When the data deviate from this canonical order—when, e.g, a subject precedes its verb—the system prefers to find some other attachment point to the left. Similarly, a noun to the left of a conjunction often erroneously attaches to its left. Such ATR relations account for 35% of noun-attachment errors.

8 Conclusion

The tradeoff between speed and accuracy is familiar to any parsing researcher. Rather than starting with an accurate system and then applying corpus-specific speedups, we start by imposing carefully-chosen constraints (projectivity and length bounds) for speed, leaving accuracy to the parsing and reranking models. As it stands, our system performs poorly, largely because the estimation is not state-of-the-art, but also in part due to dependency length bounds, which are rather coarse at present. Better results are achievable by picking different bounds for different head tags (Eisner and N. Smith, 2005). Accuracy should not be difficult to improve using better learning methods, especially given our models' linear-time inference and decoding.

References

H. Alshawi. 1996. Head automata and bilingual tiling: Translation with minimal representations. In *Proc. of ACL*.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. of ACL*.

M. Collins. 2000. Discriminative reranking for natural language parsing. In *Proc. of ICML*.

J. Eisner and G. Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proc. of ACL*.

J. Eisner and N. A. Smith. 2005. Parsing with soft and hard constraints on dependency length. In *Proc. of IWPT*.

J. Eisner, E. Goldlust, and N. A. Smith. 2004. Dyna: A declarative language for implementing dynamic programs. In *Proc. of ACL* (companion volume).

J. Eisner. 1997. Bilexical grammars and a cubic-time probabilistic parser. In *Proc. of IWPT*.

L. Huang and D. Chiang. 2005. Better k-best parsing. In *Proc. of IWPT*.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT-EMNLP*.

K. Rose. 1998. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proc. of the IEEE*, 86(11):2210–2239.

D. A. Smith and J. Eisner. 2006. Minimum risk annealing for training log-linear models. To appear in *Proc. of COLING-ACL*.