A Comparison of Loopy Belief Propagation and Dual Decomposition for Integrated CCG Supertagging and Parsing

Michael Auli
School of Informatics
University of Edinburgh
m.auli@sms.ed.ac.uk

Adam Lopez
HLTCOE
Johns Hopkins University
alopez@cs.jhu.edu

Abstract

Via an oracle experiment, we show that the upper bound on accuracy of a CCG parser is significantly lowered when its search space is pruned using a supertagger, though the supertagger also prunes many bad parses. Inspired by this analysis, we design a single model with both supertagging and parsing features, rather than separating them into distinct models chained together in a pipeline. To overcome the resulting increase in complexity, we experiment with both belief propagation and dual decomposition approaches to inference, the first empirical comparison of these algorithms that we are aware of on a structured natural language processing problem. On CCGbank we achieve a labelled dependency F-measure of 88.8% on gold POS tags, and 86.7% on automatic part-of-speeoch tags, the best reported results for this task.

1 Introduction

Accurate and efficient parsing of Combinatorial Categorial Grammar (CCG; Steedman, 2000) is a longstanding problem in computational linguistics, due to the complexities associated its mild context sensitivity. Even for practical CCG that are strongly context-free (Fowler and Penn, 2010), parsing is much harder than with Penn Treebank-style context-free grammars, with vast numbers of nonterminal categories leading to increased grammar constants. Where a typical Penn Treebank grammar may have fewer than 100 nonterminals (Hockenmaier and Steedman, 2002), we found that a CCG grammar derived from CCGbank contained over 1500. The same grammar assigns an average of 22 lexical categories per word (Clark and Curran, 2004a), resulting in an enormous space of possible derivations.

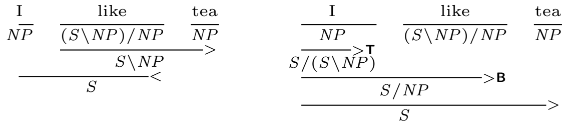
The most successful approach to CCG parsing is based on a pipeline strategy (§2). First, we tag (or multitag) each word of the sentence with a lexical category using a *supertagger*, a sequence model over these categories (Bangalore and Joshi, 1999; Clark, 2002). Second, we parse the sentence under the requirement that the lexical categories are fixed to those preferred by the supertagger. Variations on this approach drive the widely-used, broad coverage C&C parser (Clark and Curran, 2004a; Clark and Curran, 2007; Kummerfeld et al., 2010). However, it fails when the supertagger makes errors. We show experimentally that this pipeline significantly lowers the upper bound on parsing accuracy (§3).

The same experiment shows that the supertagger prunes many bad parses. So, while we want to avoid the error propagation inherent to a pipeline, ideally we still want to benefit from the key insight of supertagging: that a sequence model over lexical categories can be quite accurate. Our solution is to combine the features of both the supertagger and the parser into a single, less aggressively pruned model. The challenge with this model is its prohibitive complexity, which we address with *approximate* methods: dual decomposition and belief propagation (§4). We present the first side-by-side comparison of these algorithms on an NLP task of this complexity, measuring accuracy, convergence behavior, and runtime. In both cases our model significantly outperforms the pipeline approach, leading to the best published results in CCG parsing (§5).

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 470–480,
Portland, Oregon, June 19-24, 2011. ©2011 Association for Computational Linguistics

2 CCG and Supertagging

CCG is a lexicalized grammar formalism encoding for each word lexical categories that are either basic (eg. NN, JJ) or complex. Complex lexical categories specify the number and directionality of arguments. For example, one lexical category for the verb *like* is $(S\backslash NP)/NP$, specifying the first argument as an NP to the right and the second as an NP to the left; there are over 100 lexical categories for *like* in our lexicon. In parsing, adjacent spans are combined using a small number of binary combinatory rules like forward application or composition (Steedman, 2000; Fowler and Penn, 2010). In the first derivation below, $(S\backslash NP)/NP$ and NP combine to form the spanning category $S\backslash NP$, which only requires an NP to its left to form a complete sentence-spanning S. The second derivation uses type-raising to change the category type of *I*.



As can be inferred from even this small example, a key difficulty in parsing CCG is that the number of categories quickly becomes extremely large, and there are typically many ways to analyze every span of a sentence.

Supertagging (Bangalore and Joshi, 1999; Clark, 2002) treats the assignment of lexical categories (or *supertags*) as a sequence tagging problem. Because they do this with high accuracy, they are often exploited to prune the parser's search space: the parser only considers lexical categories with high posterior probability (or other figure of merit) under the supertagging model (Clark and Curran, 2004a). The posterior probabilities are then discarded; it is the extensive pruning of lexical categories that leads to substantially faster parsing times.

Pruning the categories in advance this way has a specific failure mode: sometimes it is not possible to produce a sentence-spanning derivation from the tag sequences preferred by the supertagger, since it does not enforce grammaticality. A workaround for this problem is the *adaptive supertagging* (AST) approach of Clark and Curran (2004a). It is based on a step function over supertagger beam widths, relaxing the pruning threshold for lexical categories only if the parser fails to find an analysis. The process either succeeds and returns a parse after some iteration or gives up after a predefined number of iterations. As Clark and Curran (2004a) show, most sentences can be parsed with a very small number of supertags per word. However, the technique is inherently approximate: it will return a lower probability parse under the parsing model if a higher probability parse can only be constructed from a supertag sequence returned by a subsequent iteration. In this way it prioritizes speed over exactness, although the tradeoff can be modified by adjusting the beam step function. Regardless, the effect of the approximation is unbounded.

We will also explore *reverse adaptive supertagging*, a much less aggressive pruning method that seeks only to make sentences parseable when they otherwise would not be due to an impractically large search space. Reverse AST starts with a wide beam, narrowing it at each iteration only if a maximum chart size is exceeded. In this way it prioritizes exactness over speed.

3 Oracle Parsing

What is the effect of these approximations? To answer this question we computed oracle best and worst values for labelled dependency F-score using the algorithm of Huang (2008) on the hybrid model of Clark and Curran (2007), the best model of their C&C parser. We computed the oracle on our development data, Section 00 of CCGbank (Hockenmaier and Steedman, 2007), using both AST and Reverse AST beams settings shown in Table 1.

The results (Table 2) show that the oracle best accuracy for reverse AST is more than 3% higher than the aggressive AST pruning.[1] In fact, it is almost as high as the upper bound oracle accuracy of 97.73% obtained using *perfect* supertags—in other words, the search space for reverse AST is theoretically near-optimal.[2] We also observe that the oracle

[1] The numbers reported here and in later sections differ slightly from those in a previously circulated draft of this paper, for two reasons: we evaluate only on sentences for which a parse was returned instead of all parses, to enable direct comparison with Clark and Curran (2007); and we use their hybrid model instead of their normal-form model, except where noted. Despite these changes our main findings remained unchanged.

[2] This idealized oracle reproduces a result from Clark and Cur-

Condition	Parameter	Iteration 1	2	3	4	5
`AST`	β (beam width)	0.075	0.03	0.01	0.005	0.001
	k (dictionary cutoff)	20	20	20	20	150
`Reverse`	β	0.001	0.005	0.01	0.03	0.075
	k	150	20	20	20	20

Table 1: Beam step function used for standard (`AST`) and less aggressive (`Reverse`) AST throughout our experiments. Parameter β is a beam threshold while k bounds the use of a part-of-speech tag dictionary, which is used for words seen less than k times.

	Viterbi F-score			Oracle Max F-score			Oracle Min F-score			
	LF	LP	LR	LF	LP	LR	LF	LP	LR	cat/word
AST	**87.38**	**87.83**	86.93	94.35	95.24	93.49	54.31	54.81	53.83	1.3-3.6
Reverse	87.36	87.55	**87.17**	**97.65**	**98.21**	**97.09**	18.09	17.75	18.43	3.6-1.3

Table 2: Comparison of adaptive supertagging (AST) and a less restrictive setting (Reverse) with Viterbi and oracle F-scores on CCGbank Section 00. The table shows the labelled F-score (LF), precision (LP) and recall (LR) and the the number of lexical categories per word used (from first to last parsing attempt).

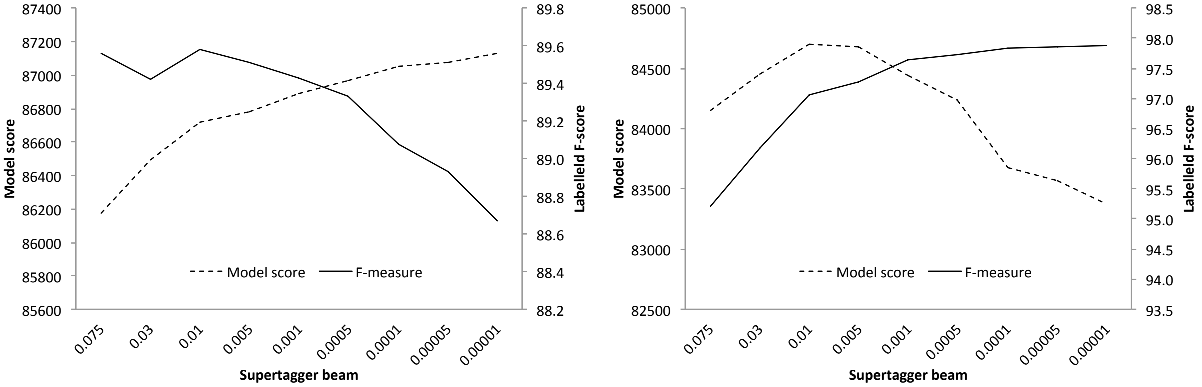


Figure 1: Comparison between model score and Viterbi F-score (left); and between model score and oracle F-score (right) for different supertagger beams on a subset of CCGbank Section 00.

worst accuracy is much lower in the reverse setting. It is clear that the supertagger pipeline has two effects: while it beneficially prunes many bad parses, it harmfully prunes some very good parses. We can also see from the scores of the Viterbi parses that while the reverse condition has access to much better parses, the model doesn't actually find them. This mirrors the result of Clark and Curran (2007) that they use to justify AST.

Digging deeper, we compared parser model score against Viterbi F-score and oracle F-score at a variety of fixed beam settings (Figure 1), considering only the subset of our development set which could be parsed with *all* beam settings. The inverse relationship between model score and F-score shows that the supertagger restricts the parser to mostly good parses (under F-measure) that the model would otherwise disprefer. Exactly this effect is exploited in the pipeline model. However, when the supertagger makes a mistake, the parser cannot recover.

4 Integrated Supertagging and Parsing

The supertagger obviously has good but not perfect predictive features. An obvious way to exploit this without being bound by its decisions is to incorporate these features directly into the parsing model.

ran (2004b). The reason that using the gold-standard supertags doesn't result in 100% oracle parsing accuracy is that some of the development set parses cannot be constructed by the learned grammar.

In our case both the parser and the supertagger are feature-based models, so from the perspective of a single parse tree, the change is simple: the tree is simply scored by the weights corresponding to all of its active features. However, since the features of the supertagger are all Markov features on adjacent supertags, the change has serious implications for search. If we think of the supertagger as defining a weighted regular language consisting of all supertag sequences, and the parser as defining a weighted mildly context-sensitive language consisting of only a *subset* of these sequences, then the search problem is equivalent to finding the optimal derivation in the weighted intersection of a regular and mildly context-sensitive language. Even allowing for the observation of Fowler and Penn (2010) that our practical CCG is context-free, this problem still reduces to the construction of Bar-Hillel et al. (1964), making search very expensive. Therefore we need approximations.

Fortunately, recent literature has introduced two relevant approximations to the NLP community: loopy belief propagation (Pearl, 1988), applied to dependency parsing by Smith and Eisner (2008); and dual decomposition (Dantzig and Wolfe, 1960; Komodakis et al., 2007; Sontag et al., 2010, *inter alia*), applied to dependency parsing by Koo et al. (2010) and lexicalized CFG parsing by Rush et al. (2010). We apply both techniques to our integrated supertagging and parsing model.

4.1 Loopy Belief Propagation

Belief propagation (BP) is an algorithm for computing marginals (i.e. expectations) on structured models. These marginals can be used for decoding (parsing) in a minimum-risk framework (Smith and Eisner, 2008); or for training using a variety of algorithms (Sutton and McCallum, 2010). We experiment with both uses in §5. Many researchers in NLP are familiar with two special cases of belief propagation: the forward-backward and inside-outside algorithms, used for computing expectations in sequence models and context-free grammars, respectively.[3] Our use of belief propagation builds directly on these two familiar algorithms.

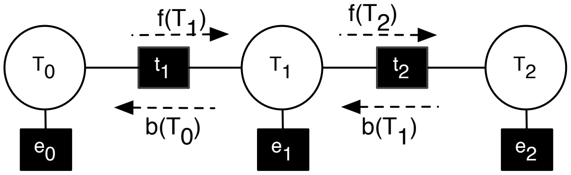


Figure 2: Supertagging factor graph with messages. Circles are variables and filled squares are factors.

BP is usually understood as an algorithm on bipartite *factor graphs*, which structure a global function into local functions over subsets of variables (Kschischang et al., 1998). Variables maintain a *belief* (expectation) over a distribution of values and BP passes *messages* about these beliefs between variables and factors. The idea is to iteratively update each variable's beliefs based on the beliefs of neighboring variables (through a shared factor), using the sum-product rule.

This results in the following equation for a message $m_{x\to f}(x)$ from a variable x to a factor f

$$m_{x\to f}(x) = \prod_{h\in n(x)\backslash f} m_{h\to x}(x) \qquad (1)$$

where $n(x)$ is the set of all neighbours of x. The message $m_{f\to x}$ from a factor to a variable is

$$m_{f\to x}(x) = \sum_{\sim\{x\}} f(X) \prod_{y\in n(f)\backslash x} m_{y\to f}(y) \qquad (2)$$

where $\sim\{x\}$ represents all variables other than x, $X = n(f)$ and $f(X)$ is the set of arguments of the factor function f.

Making this concrete, our supertagger defines a distribution over tags $T_0...T_I$, based on emission factors $e_0...e_I$ and transition factors $t_1...t_I$ (Figure 2). The message f_i a variable T_i receives from its neighbor to the left corresponds to the forward probability, while messages from the right correspond to backward probability b_i.

$$f_i(T_i) = \sum_{T_{i-1}} f_{i-1}(T_{i-1})e_{i-1}(T_{i-1})t_i(T_{i-1},T_i) \quad (3)$$

$$b_i(T_i) = \sum_{T_{i+1}} b_{i+1}(T_{i+1})e_{i+1}(T_{i+1})t_{i+1}(T_i,T_{i+1}) \quad (4)$$

[3] Forward-backward and inside-outside are formally shown to be special cases of belief propagation by Smyth et al. (1997) and Sato (2007), respectively.

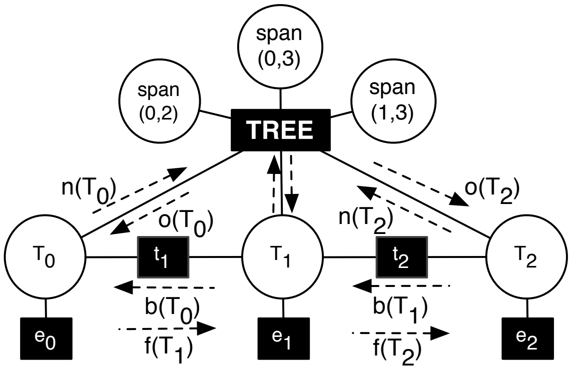


Figure 3: Factor graph for the combined parsing and supertagging model.

The current belief $B_x(x)$ for variable x can be computed by taking the normalized product of all its incoming messages.

$$B_x(x) = \frac{1}{Z} \prod_{h \in n(x)} m_{h \to x}(x) \tag{5}$$

In the supertagger model, this is just:

$$p(T_i) = \frac{1}{Z} f_i(T_i) b_i(T_i) e_i(T_i) \tag{6}$$

Our parsing model is *also* a distribution over variables T_i, along with an additional quadratic number of $span(i,j)$ variables. Though difficult to represent pictorially, a distribution over parses is captured by an extension to graphical models called case-factor diagrams (McAllester et al., 2008). We add this complex distribution to our model as a single factor (Figure 3). This is a natural extension to the use of complex factors described by Smith and Eisner (2008) and Dreyer and Eisner (2009).

When a factor graph is a tree as in Figure 2, BP converges in a single iteration to the exact marginals. However, when the model contains cycles, as in Figure 3, we can iterate message passing. Under certain assumptions this *loopy* BP it will converge to approximate marginals that are bounded under an interpretation from statistical physics (Yedidia et al., 2001; Sutton and McCallum, 2010).

The TREE factor exchanges *inside* n_i and *outside* o_i messages with the tag and span variables, taking into account beliefs from the sequence model. We will omit the unchanged outside recursion for brevity, but inside messages $n(C_{i,j})$ for category $C_{i,j}$ in $span(i,j)$ are computed using rule probabilities r as follows:

$$n(C_{i,j}) = \begin{cases} f_i(C_{i,j}) b_i(C_{i,j}) e_i(C_{i,j}) & \text{if j=i+1} \\ \sum\limits_{k,X,Y} n(X_{i,k}) n(Y_{k,j}) r(C_{i,j}, X_{i,k}, Y_{k,j}) & \end{cases} \tag{7}$$

Note that the only difference from the classic inside algorithm is that the recursive base case of a category spanning a single word has been replaced by a message from the supertag that contains both forward and backward factors, along with a unary emission factor, which doubles as a unary rule factor and thus contains the only shared features of the original models. This difference is also mirrored in the forward and backward messages, which are identical to Equations 3 and 4, except that they also incorporate outside messages from the tree factor.

Once all forward-backward and inside-outside probabilities have been calculated the belief of supertag T_i can be computed as the product of all incoming messages. The only difference from Equation 6 is the addition of the outside message.

$$p(T_i) = \frac{1}{Z} f_i(T_i) b_i(T_i) e_i(T_i) o_i(T_i) \tag{8}$$

The algorithm repeatedly runs forward-backward and inside-outside, passing their messages back and forth, until these quantities converge.

4.2 Dual Decomposition

Dual decomposition (Rush et al., 2010; Koo et al., 2010) is a decoding (i.e. search) algorithm for problems that can be decomposed into exactly solvable subproblems: in our case, supertagging and parsing. Formally, given Y as the set of valid parses, Z as the set of valid supertag sequences, and T as the set of supertags, we want to solve the following optimization for parser $f(y)$ and supertagger $g(z)$.

$$\arg\max_{y \in Y, z \in Z} f(y) + g(z) \tag{9}$$

$$\text{such that } y(i,t) = z(i,t) \text{ for all } (i,t) \in I \tag{10}$$

Here $y(i,t)$ is a binary function indicating whether word i is assigned supertag t by the parser, for the set $I = \{(i,t) : i \in 1 \ldots n, t \in T\}$ denoting the set of permitted supertags for each word; similarly $z(i,t)$ for the supertagger. To enforce the constraint that the parser and supertagger agree on a tag sequence we introduce Lagrangian multipliers $u = \{u(i,t) : (i,t) \in I\}$ and construct a dual objective over variables $u(i,t)$.

$$L(u) = \max_{y \in Y}(f(y) - \sum_{i,t} u(i,t)y(i,t)) \quad (11)$$
$$+ \max_{z \in Z}(f(z) + \sum_{i,t} u(i,t)z(i,t))$$

This objective is an upper bound that we want to make as tight as possible by solving for $\min_u L(u)$. We optimize the values of the $u(i,t)$ variables using the same algorithm as Rush et al. (2010) for their tagging and parsing problem (essentially a perceptron update).[4] An advantages of DD is that, on convergence, it recovers *exact* solutions to the combined problem. However, if it does not converge or we stop early, an approximation must be returned: following Rush et al. (2010) we used the highest scoring output of the parsing submodel over all iterations.

5 Experiments

Parser. We use the C&C parser (Clark and Curran, 2007) and its supertagger (Clark, 2002). Our baseline is the hybrid model of Clark and Curran (2007); our integrated model simply adds the supertagger features to this model. The parser relies solely on the supertagger for pruning, using CKY for search over the pruned space. Training requires repeated calculation of feature expectations over packed charts of derivations. For training, we limited the number of items in this chart to 0.3 million, and for testing, 1 million. We also used a more permissive training supertagger beam (Table 3) than in previous work (Clark and Curran, 2007). Models were trained with the parser's L-BFGS trainer.

Evaluation. We evaluated on CCGbank (Hockenmaier and Steedman, 2007), a right-most normal-form CCG version of the Penn Treebank. We use sections 02-21 (39603 sentences) for training, section 00 (1913 sentences) for development and section 23 (2407 sentences) for testing. We supply gold-standard part-of-speech tags to the parsers. Evaluation is based on labelled and unlabelled predicate argument structure recovery and supertag accuracy. We only evaluate on sentences for which an analysis was returned; the coverage for *all* parsers is 99.22% on section 00, and 99.63% on section 23.

Model combination. We combine the parser and the supertagger over the search space defined by the set of supertags within the supertagger beam (see Table 1); this avoids having to perform inference over the prohibitively large set of parses spanned by all supertags. Hence at each beam setting, the model operates over the same search space as the baseline; the difference is that we search with our integrated model.

5.1 Parsing Accuracy

We first experiment with the separately trained supertagger and parser, which are then combined using belief propagation (BP) and dual decomposition (DD). We run the algorithms for many iterations, and irrespective of convergence, for BP we compute the minimum risk parse from the current marginals, and for DD we choose the highest-scoring parse seen over all iterations. We measured the evolving accuracy of the models on the development set (Figure 4). In line with our oracle experiment, these results demonstrate that we can coax more accurate parses from the larger search space provided by the reverse setting; the influence of the supertagger features allow us to exploit this advantage.

One behavior we observe in the graph is that the DD results tend to incrementally improve in accuracy while the BP results quickly stabilize, mirroring the result of Smith and Eisner (2008). This occurs because DD continues to find higher scoring parses at each iteration, and hence the results change. However for BP, even if the marginals have not converged, the minimum risk solution turns out to be fairly stable across successive iterations.

We next compare the algorithms against the baseline on our test set (Table 4). We find that the early stability of BP's performance generalises to the test set as does DD's improvement over several iterations. More importantly, we find that the applying

[4] The u terms can be interpreted as the *messages* from factors to variables (Sontag et al., 2010) and the resulting message passing algorithms are similar to the *max-product algorithm*, a sister algorithm to BP.

	Parameter	Iteration 1	2	3	4	5	6	7
Training	β	0.001	0.001	0.0045	0.0055	0.01	0.05	0.1
	k	150	20	20	20	20	20	20

Table 3: Beam step function used for training (cf. Table 1).

	section 00 (dev)						section 23 (test)					
	AST			Reverse			AST			Reverse		
	LF	UF	ST	LF	UF	ST	LF	UF	ST	LF	UF	ST
Baseline	87.38	93.08	94.21	87.36	93.13	93.99	87.73	93.09	94.33	87.65	93.06	94.01
C&C '07	87.24	93.00	94.16	-	-	-	87.64	93.00	94.32	-	-	-
BP$_{k=1}$	87.70	93.28	94.44	**88.35**	93.69	**94.73**	**88.20**	**93.28**	**94.60**	88.78	93.66	94.81
BP$_{k=25}$	87.70	93.31	94.44	88.33	**93.72**	94.71	88.19	93.27	94.59	88.80	93.68	94.81
DD$_{k=1}$	87.40	93.09	94.23	87.38	93.15	94.03	87.74	93.10	94.33	87.67	93.07	94.02
DD$_{k=25}$	**87.71**	**93.32**	**94.44**	88.29	93.71	94.67	88.14	93.24	94.59	**88.80**	**93.68**	**94.82**

Table 4: Results for individually-trained submodels combined using dual decomposition (DD) or belief propagation (BP) for k iterations, evaluated by labelled and unlabelled F-score (LF/UF) and supertag accuracy (ST). We compare against the previous best result of Clark and Curran (2007); our baseline is their model with wider training beams (cf. Table 3).

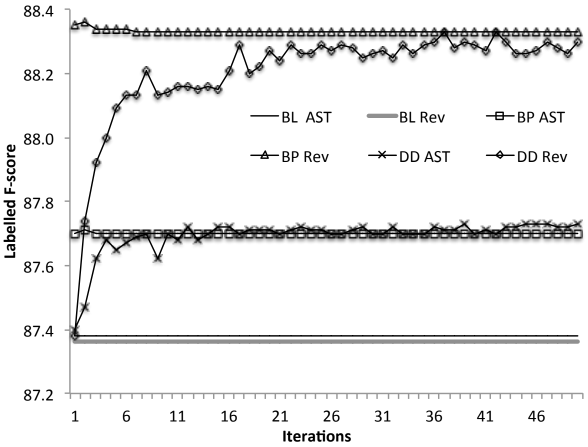


Figure 4: Labelled F-score of baseline (BL), belief propagation (BP), and dual decomposition (DD) on section 00.

our combined model using either algorithm *consistently* outperforms the baseline after only a few iterations. Overall, we improve the labelled F-measure by almost 1.1% and unlabelled F-measure by 0.6% over the baseline. To the best of our knowledge, the results obtained with BP and DD are the best reported results on this task using gold POS tags.

Next, we evaluate performance when using *automatic* part-of-speech tags as input to our parser and supertagger (Table 5). This enables us to compare against the results of Fowler and Penn (2010), who trained the Petrov parser (Petrov et al., 2006) on CCGbank. We outperform them on all criteria. Hence our combined model represents the best CCG parsing results under any setting.

Finally, we revisit the oracle experiment of §3 using our combined models (Figure 5). Both show an improved relationship between model score and F-measure.

5.2 Algorithmic Convergence

Figure 4 shows that parse accuracy converges after a few iterations. Do the *algorithms* converge? BP converges when the marginals do not change between iterations, and DD converges when both submodels agree on all supertags. We measured the convergence of each algorithm under these criteria over 1000 iterations (Figure 6). DD converges much faster, while BP in the reverse condition converges quite slowly. This is interesting when contrasted with its behavior on parse accuracy—its rate of convergence after one iteration is 1.5%, but its accuracy is already the highest at this point. Over the entire 1000 iterations, most sentences converge: all but 3 for BP (both in AST and reverse) and all but

	section 00 (dev)						section 23 (test)					
	LF	LP	LR	UF	UP	UR	LF	LP	LR	UF	UP	UR
Baseline	85.53	85.73	85.33	91.99	92.20	91.77	85.74	85.90	85.58	91.92	92.09	91.75
Petrov I-5	85.79	86.09	85.50	92.44	92.76	92.13	86.01	86.29	85.74	92.34	92.64	92.04
BP$_{k=1}$	**86.44**	**86.74**	**86.14**	**92.54**	**92.86**	**92.23**	**86.73**	**86.95**	**86.50**	**92.45**	**92.69**	**92.21**
DD$_{k=25}$	86.35	86.65	86.05	92.52	92.85	92.20	86.68	86.90	86.46	92.44	92.67	92.21

Table 5: Results on automatically assigned POS tags. *Petrov I-5* is based on the parser output of Fowler and Penn (2010); we evaluate on sentences for which *all* parsers returned an analysis (2323 sentences for section 23 and 1834 sentences for section 00).

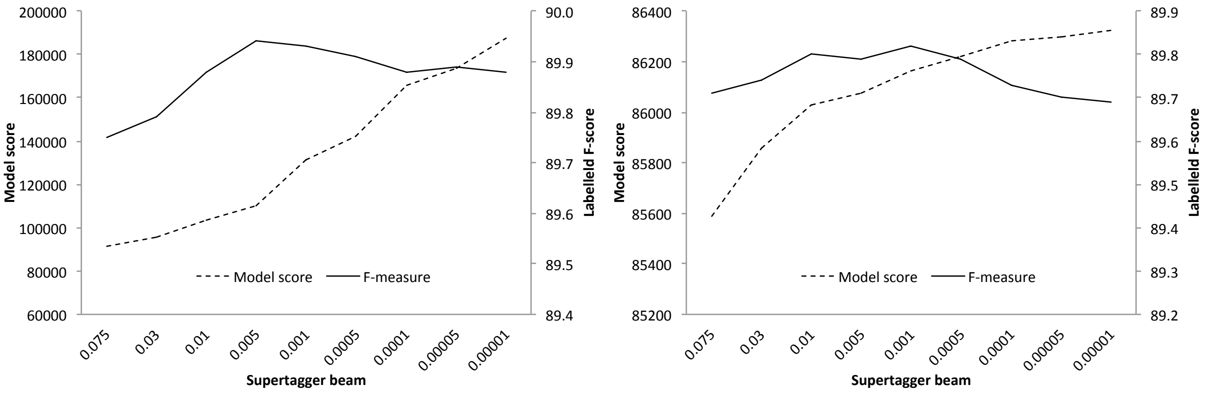


Figure 5: Comparison between model score and Viterbi F-score for the integrated model using belief propagation (left) and dual decomposition (right); the results are based on the same data as Figure 1.

.

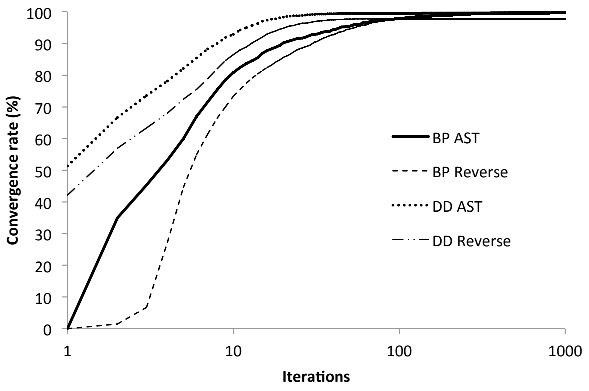


Figure 6: Rate of convergence for belief propagation (BP) and dual decomposition (DD) with maximum $k = 1000$.

41 (2.6%) for DD in reverse (6 in AST).

5.3 Parsing Speed

Because the C&C parser with AST is very fast, we wondered about the effect on speed for our model. We measured the runtime of the algorithms under the condition that we stopped at a particular iteration (Table 6). Although our models improve substantially over C&C, there is a significant cost in speed for the best result.

5.4 Training the Integrated Model

In the experiments reported so far, the parsing and supertagging models were trained separately, and only combined at test time. Although the outcome of these experiments was successful, we wondered if we could obtain further improvements by training the model parameters together.

Since the gradients produced by (loopy) BP are approximate, for these experiments we used a stochastic gradient descent (SGD) trainer (Bottou, 2003). We found that the SGD parameters described by Finkel et al. (2008) worked equally well for our models, and, on the baseline, produced similar results to L-BFGS. Curiously, however, we found that the combined model does not perform as well when

	AST		Reverse	
	sent/sec	LF	sent/sec	LF
Baseline	65.8	87.38	5.9	87.36
BP$_{k=1}$	60.8	87.70	5.8	88.35
BP$_{k=5}$	46.7	87.70	4.7	88.34
BP$_{k=25}$	35.3	87.70	3.5	88.33
DD$_{k=1}$	64.6	87.40	5.9	87.38
DD$_{k=5}$	41.9	87.65	3.1	88.09
DD$_{k=25}$	32.5	87.71	1.9	88.29

Table 6: Parsing time in seconds per sentence (vs. F-measure) on section 00.

	AST			Reverse		
	LF	UF	ST	LF	UF	ST
Baseline	86.7	92.7	94.0	86.7	92.7	93.9
BP inf	86.8	92.8	94.1	87.2	93.1	94.2
BP train	86.3	92.5	93.8	85.6	92.1	93.2

Table 7: Results of training with SGD on approximate gradients from LPB on section 00. We test LBP in both inference and training (train) as well as in inference only (inf); a maximum number of 10 iterations is used.

the parameters are trained together (Table 7). A possible reason for this is that we used a stricter supertagger beam setting during training (Clark and Curran, 2007) to make training on a single machine practical. This leads to lower performance, particularly in the Reverse condition. Training a model using DD would require a different optimization algorithm based on Viterbi results (e.g. the perceptron) which we will pursue in future work.

6 Conclusion and Future Work

Our approach of combining models to avoid the pipeline problem (Felzenszwalb and McAllester, 2007) is very much in line with much recent work in NLP. Such diverse topics as machine translation (Dyer et al., 2008; Dyer and Resnik, 2010; Mi et al., 2008), part-of-speech tagging (Jiang et al., 2008), named entity recognition (Finkel and Manning, 2009) semantic role labelling (Sutton and McCallum, 2005; Finkel et al., 2006), and others have also been improved by combined models. Our empirical comparison of BP and DD also complements the theoretically-oriented comparison of marginal- and margin-based variational approximations for parsing described by Martins et al. (2010).

We have shown that the aggressive pruning used in adaptive supertagging significantly harms the oracle performance of the parser, though it mostly prunes bad parses. Based on these findings, we combined parser and supertagger features into a single model. Using belief propagation and dual decomposition, we obtained more principled—and more accurate—approximations than a pipeline. Models combined using belief propagation achieve very good performance immediately, despite an initial convergence rate just over 1%, while dual decomposition produces comparable results after several iterations, and algorithmically converges more quickly. Our best result of 88.8% represents the state-of-the art in CCG parsing accuracy.

In future work we plan to integrate the POS tagger, which is crucial to parsing accuracy (Clark and Curran, 2004b). We also plan to revisit the idea of combined training. Though we have focused on CCG in this work we expect these methods to be equally useful for other linguistically motivated but computationally complex formalisms such as lexicalized tree adjoining grammar.

Acknowledgements

We would like to thank Phil Blunsom, Prachya Boonkwan, Christos Christodoulopoulos, Stephen Clark, Michael Collins, Chris Dyer, Timothy Fowler, Mark Granroth-Wilding, Philipp Koehn, Terry Koo, Tom Kwiatkowski, André Martins, Matt Post, David Smith, David Sontag, Mark Steedman, and Charles Sutton for helpful discussion related to this work and comments on previous drafts, and the anonymous reviewers for helpful comments. We also acknowledge funding from EPSRC grant EP/P504171/1 (Auli); the EuroMatrixPlus project funded by the European Commission, 7th Framework Programme (Lopez); and the resources provided by the Edinburgh Compute and Data Facility (`http://www.ecdf.ed.ac.uk`). The ECDF is partially supported by the eDIKT initiative (`http://www.edikt.org.uk`).

References

S. Bangalore and A. K. Joshi. 1999. Supertagging: An Approach to Almost Parsing. *Computational Linguis-*

tics, 25(2):238–265, June.

Y. Bar-Hillel, M. Perles, and E. Shamir. 1964. On formal properties of simple phrase structure grammars. In *Language and Information: Selected Essays on their Theory and Application*, pages 116–150.

L. Bottou. 2003. Stochastic learning. In *Advanced Lectures in Machine Learning*, pages 146–168.

S. Clark and J. R. Curran. 2004a. The importance of supertagging for wide-coverage CCG parsing. In *COLING*, Morristown, NJ, USA.

S. Clark and J. R. Curran. 2004b. Parsing the WSJ using CCG and log-linear models. In *Proc. of ACL*, pages 104–111, Barcelona, Spain.

S. Clark and J. R. Curran. 2007. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4):493–552.

S. Clark. 2002. Supertagging for Combinatory Categorial Grammar. In *TAG+6*.

G. B. Dantzig and P. Wolfe. 1960. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111.

M. Dreyer and J. Eisner. 2009. Graphical models over multiple strings. In *Proc. of EMNLP*.

C. Dyer and P. Resnik. 2010. Context-free reordering, finite-state translation. In *Proc. of HLT-NAACL*.

C. J. Dyer, S. Muresan, and P. Resnik. 2008. Generalizing word lattice translation. In *Proc. of ACL*.

P. F. Felzenszwalb and D. McAllester. 2007. The Generalized A* Architecture. In *Journal of Artificial Intelligence Research*, volume 29, pages 153–190.

J. R. Finkel and C. D. Manning. 2009. Joint parsing and named entity recognition. In *Proc. of NAACL*. Association for Computational Linguistics.

J. R. Finkel, C. D. Manning, and A. Y. Ng. 2006. Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines. In *Proc. of EMNLP*.

J. R. Finkel, A. Kleeman, and C. D. Manning. 2008. Feature-based, conditional random field parsing. In *Proceedings of ACL-HLT*.

T. A. D. Fowler and G. Penn. 2010. Accurate context-free parsing with combinatory categorial grammar. In *Proc. of ACL*.

J. Hockenmaier and M. Steedman. 2002. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proc. of ACL*.

J. Hockenmaier and M. Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

L. Huang. 2008. Forest Reranking: Discriminative parsing with Non-Local Features. In *Proceedings of ACL-08: HLT*.

W. Jiang, L. Huang, Q. Liu, and Y. Lü. 2008. A cascaded linear model for joint Chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL-08: HLT*.

N. Komodakis, N. Paragios, and G. Tziritas. 2007. MRF optimization via dual decomposition: Message-passing revisited. In *Proc. of Int. Conf. on Computer Vision (ICCV)*.

T. Koo, A. M. Rush, M. Collins, T. Jaakkola, and D. Sontag. 2010. Dual Decomposition for Parsing with Non-Projective Head Automata. In *In Proc. EMNLP*.

F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. 1998. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47:498–519.

J. K. Kummerfeld, J. Rosener, T. Dawborn, J. Haggerty, J. R. Curran, and S. Clark. 2010. Faster parsing by supertagger adaptation. In *Proc. of ACL*.

A. F. T. Martins, N. A. Smith, E. P. Xing, P. M. Q. Aguiar, and M. A. T. Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proc. of EMNLP*.

D. McAllester, M. Collins, and F. Pereira. 2008. Case-factor diagrams for structured probabilistic modeling. *Journal of Computer and System Sciences*, 74(1):84–96.

H. Mi, L. Huang, and Q. Liu. 2008. Forest-based translation. In *Proc. of ACL-HLT*.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proc. of ACL*.

A. M. Rush, D. Sontag, M. Collins, and T. Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *In Proc. EMNLP*.

T. Sato. 2007. Inside-outside probability computation for belief propagation. In *Proc. of IJCAI*.

D. A. Smith and J. Eisner. 2008. Dependency parsing by belief propagation. In *Proc. of EMNLP*.

P. Smyth, D. Heckerman, and M. Jordan. 1997. Probabilistic independence networks for hidden Markov probability models. *Neural computation*, 9(2):227–269.

D. Sontag, A. Globerson, and T. Jaakkola. 2010. Introduction to dual decomposition. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*. MIT Press.

M. Steedman. 2000. *The syntactic process*. MIT Press, Cambridge, MA.

C. Sutton and A. McCallum. 2005. Joint parsing and semantic role labelling. In *Proc. of CoNLL*.

C. Sutton and A. McCallum. 2010. An introduction to conditional random fields. arXiv:stat.ML/1011.4088.

J. Yedidia, W. Freeman, and Y. Weiss. 2001. Generalized belief propagation. In *Proc. of NIPS*.