Training a Log-Linear Parser with Loss Functions via Softmax-Margin

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

Log-linear parsing models are often trained by optimizing likelihood, but we would prefer to optimise for a task-specific metric like F-measure. Softmax-margin is a convex objective for such models that minimises a bound on expected risk for a given loss function, but its naïve application requires the loss to decompose over the predicted structure, which is not true of F-measure. We use softmax-margin to optimise a log-linear CCG parser for a variety of loss functions, and demonstrate a novel dynamic programming algorithm that enables us to use it with F-measure, leading to substantial gains in accuracy on CCGBank. When we embed our loss-trained parser into a larger model that includes supertagging features incorporated via belief propagation, we obtain further improvements and achieve a labelled/unlabelled dependency F-measure of 89.3%/94.0% on gold part-of-speech tags, and 87.2%/92.8% on automatic part-of-speech tags, the best reported results for this task.

1 Introduction

Parsing models based on Conditional Random Fields (CRFs; Lafferty et al., 2001) have been very successful (Clark and Curran, 2007; Finkel et al., 2008). In practice, they are usually trained by maximising the conditional log-likelihood (CLL) of the training data. However, it is widely appreciated that optimizing for task-specific metrics often leads to better performance on those tasks (Goodman, 1996; Och, 2003).

An especially attractive means of accomplishing this for CRFs is the softmax-margin (SMM) objective (Sha and Saul, 2006; Povey and Woodland, 2008; Gimpel and Smith, 2010a) (§2). In addition to retaining a probabilistic interpretation and optimizing towards a loss function, it is also convex, making it straightforward to optimise. Gimpel and Smith (2010a) show that it can be easily implemented with a simple change to standard likelihood-based training, provided that the loss function decomposes over the predicted structure.

Unfortunately, the widely-used F-measure metric does not decompose over parses. To solve this, we introduce a novel dynamic programming algorithm that enables us to compute the exact quantities needed under the softmax-margin objective using F-measure as a loss (§3). We experiment with this and several other metrics, including precision, recall, and decomposable approximations thereof. Our ability to optimise towards exact metrics enables us to verify the effectiveness of more efficient approximations. We test the training procedures on the state-of-the-art Combinatory Categorial Grammar (CCG; Steedman 2000) parser of Clark and Curran (2007), obtaining substantial improvements under a variety of conditions. We then embed this model into a more accurate model that incorporates additional supertagging features via loopy belief propagation. The improvements are additive, obtaining the best reported results on this task (§4).

2 Softmax-Margin Training

The softmax-margin objective modifies the standard likelihood objective for CRF training by reweighting

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 333–343,
Edinburgh, Scotland, UK, July 27–31, 2011. ©2011 Association for Computational Linguistics

each possible outcome of a training input according to its *risk*, which is simply the loss incurred on a particular example. This is done by incorporating the loss function directly into the linear scoring function of an individual example.

Formally, we are given m training pairs $(x^{(1)}, y^{(1)})...(x^{(m)}, y^{(m)})$, where each $x^{(i)} \in \mathcal{X}$ is drawn from the set of possible inputs, and each $y^{(i)} \in \mathcal{Y}(x^{(i)})$ is drawn from a set of possible instance-specific outputs. We want to learn the K parameters θ of a log-linear model, where each $\lambda_k \in \theta$ is the weight of an associated feature $h_k(x, y)$. Function $f(x, y)$ maps input/output pairs to the vector $h_1(x, y)...h_K(x, y)$, and our log-linear model assigns probabilities in the usual way.

$$p(y|x) = \frac{\exp\{\theta^\mathsf{T} f(x, y)\}}{\sum_{y' \in \mathcal{Y}(x)} \exp\{\theta^\mathsf{T} f(x, y')\}} \quad (1)$$

The conditional log-likelihood objective function is given by Eq. 2 (Figure 1). Now consider a function $\ell(y, y')$ that returns the loss incurred by choosing to output y' when the correct output is y. The softmax-margin objective simply modifies the unnormalised, unexponentiated score $\theta^\mathsf{T} f(x, y')$ by adding $\ell(y, y')$ to it. This yields the objective function (Eq. 3) and gradient computation (Eq. 4) shown in Figure 1.

This straightforward extension has several desirable properties. In addition to having a probabilistic interpretation, it is related to maximum margin and minimum-risk frameworks, it can be shown to minimise a bound on expected risk, and it is convex (Gimpel and Smith, 2010b).

We can also see from Eq. 4 that the only difference from standard CLL training is that we must compute feature expectations with respect to the cost-augmented scoring function. As Gimpel and Smith (2010a) discuss, if the loss function decomposes over the predicted structure, we can treat its decomposed elements as unweighted features that fire on the corresponding structures, and compute expectations in the normal way. In the case of our parser, where we compute expectations using the inside-outside algorithm, a loss function decomposes if it decomposes over spans or productions of a CKY chart.

3 Loss Functions for Parsing

Ideally, we would like to optimise our parser towards a task-based evaluation. Our CCG parser is evaluated on labeled, directed dependency recovery using F-measure (Clark and Hockenmaier, 2002). Under this evaluation we will represent output y' and ground truth y as variable-sized sets of dependencies. We can then compute precision $P(y, y')$, recall $R(y, y')$, and F-measure $F_1(y, y')$.

$$P(y, y') = \frac{|y \cap y'|}{|y'|} \quad (5)$$

$$R(y, y') = \frac{|y \cap y'|}{|y|} \quad (6)$$

$$F_1(y, y') = \frac{2PR}{P + R} = \frac{2|y \cap y'|}{|y| + |y'|} \quad (7)$$

These metrics are positively correlated with performance – they are *gain* functions. To incorporate them in the softmax-margin framework we reformulate them as loss functions by subtracting from one.

3.1 Computing F-Measure-Augmented Expectations at the Sentence Level

Unfortunately, none of these metrics decompose over parses. However, the individual statistics that are used to compute them *do* decompose, a fact we will exploit to devise an algorithm that computes the necessary expectations. Note that since y is fixed, F_1 is a function of two integers: $|y \cap y'|$, representing the number of correct dependencies in y'; and $|y'|$, representing the total number of dependencies in y', which we will denote as n and d, respectively.[1] Each pair $\langle n, d \rangle$ leads to a different value of F_1. Importantly, both n and d decompose over parses.

The key idea will be to treat F_1 as a non-local feature of the parse, dependent on values n and d.[2] To compute expectations we split each span in an otherwise usual inside-outside computation by all pairs $\langle n, d \rangle$ incident at that span.

Formally, our goal will be to compute expectations over the sentence $a_1...a_L$. In order to abstract away from the particulars of CCG we present the algorithm in relatively familiar terms as a variant of

[1] For *numerator* and *denominator*.

[2] This is essentially the same trick used in the oracle F-measure algorithm of Huang (2008), and indeed our algorithm is a sum-product variant of that max-product algorithm.

$$\min_{\theta} \sum_{i=1}^{m} \left[-\theta^{\mathsf{T}} f(x^{(i)}, y^{(i)}) + \log \sum_{y \in \mathcal{Y}(x^{(i)})} \exp\{\theta^{\mathsf{T}} f(x^{(i)}, y)\} \right] \tag{2}$$

$$\min_{\theta} \sum_{i=1}^{m} \left[-\theta^{\mathsf{T}} f(x^{(i)}, y^{(i)}) + \log \sum_{y \in \mathcal{Y}(x^{(i)})} \exp\{\theta^{\mathsf{T}} f(x^{(i)}, y) + \ell(y^{(i)}, y)\} \right] \tag{3}$$

$$\frac{\partial}{\partial \lambda_k} = \sum_{i=1}^{m} \left[-h_k(x^{(i)}, y^{(i)}) + \sum_{y \in \mathcal{Y}(x^{(i)})} \frac{\exp\{\theta^{\mathsf{T}} f(x^{(i)}, y) + \ell(y^{(i)}, y)\}}{\sum_{y' \in \mathcal{Y}(x^{(i)})} \exp\{\theta^{\mathsf{T}} f(x^{(i)}, y') + \ell(y^{(i)}, y')\}} h_k(x^{(i)}, y) \right] \tag{4}$$

Figure 1: Conditional log-likelihood (Eq. 2), Softmax-margin objective (Eq. 3) and gradient (Eq. 4).

the classic inside-outside algorithm (Baker, 1979). We use the notation $a : A$ for lexical entries and $BC \Rightarrow A$ to indicate that categories B and C combine to form category A via forward or backward composition or application.[3] The weight of a rule is denoted with w. The classic algorithm associates inside score $I(A_{i,j})$ and outside score $O(A_{i,j})$ with category A spanning sentence positions i through j, computed via the following recursions.

$$\begin{aligned}
I(A_{i,i+1}) =& w(a_{i+1} : A) \\
I(A_{i,j}) =& \sum_{k,B,C} I(B_{i,k}) I(C_{k,j}) w(BC \Rightarrow A) \\
I(GOAL) =& I(S_{0,L}) \\
O(GOAL) =& 1 \\
O(A_{i,j}) =& \sum_{k,B,C} O(C_{i,k}) I(B_{j,k}) w(AB \Rightarrow C) + \\
& \sum_{k,B,C} O(C_{k,j}) I(B_{k,i}) w(BA \Rightarrow C)
\end{aligned}$$

The expectation of A spanning positions i through j is then $I(A_{i,j})O(A_{i,j})/I(GOAL)$.

Our algorithm extends these computations to state-split items $A_{i,j,n,d}$.[4] Using functions $n_+(\cdot)$ and $d_+(\cdot)$ to respectively represent the number of correct and total dependencies introduced by a parsing action, we present our algorithm in Fig. 3. The final inside equation and initial outside equation incorporate the loss function for all derivations having a particular F-score, enabling us to obtain the desired expectations. A simple modification of the goal equations enables us to optimise precision, recall or a weighted F-measure.

To analyze the complexity of this algorithm, we must ask: how many pairs $\langle n, d \rangle$ can be incident at each span? A CCG parser does not necessarily return one dependency per word (see Figure 2 for an example), so d is not necessarily equal to the sentence length L as it might be in many dependency parsers, though it is still bounded by $\mathcal{O}(L)$. However, this behavior is sufficiently uncommon that we expect all parses of a sentence, good or bad, to have close to L dependencies, and hence we expect the range of d to be constant on average. Furthermore, n will be bounded from below by zero and from above by $\min(|y|, |y'|)$. Hence the set of all possible F-measures for all possible parses is bounded by $\mathcal{O}(L^2)$, but on average it should be closer to $\mathcal{O}(L)$. Following McAllester (1999), we can see from inspection of the free variables in Fig. 3 that the algorithm requires worst-case $\mathcal{O}(L^7)$ and average-case $\mathcal{O}(L^5)$ time complexity, and worse-case $\mathcal{O}(L^4)$ and average-case $\mathcal{O}(L^3)$ space complexity.

Note finally that while this algorithm computes exact sentence-level expectations, it is approximate at the corpus level, since F-measure does not decompose over sentences. We give the extension to exact corpus-level expectations in Appendix A.

3.2 Approximate Loss Functions

We will also consider approximate but more efficient alternatives to our exact algorithms. The idea is to use cost functions which only utilise statistics

[3] These correspond respectively to unary rules $A \rightarrow a$ and binary rules $A \rightarrow BC$ in a Chomsky normal form grammar.

[4] Here we use *state-splitting* to refer to splitting an item $A_{i,j}$ into many items $A_{i,j,n,d}$, one for each $\langle n, d \rangle$ pair.

$$I(A_{i,i+1,n,d}) = w(a_{i+1} : A) \quad \text{iff} n = n_+(a_{i+1} : A), d = d_+(a_{i+1} : A)$$

$$I(A_{i,j,n,d}) = \sum_{k,B,C} \sum_{\substack{\{n',n'':n'+n''+n_+(BC \Rightarrow A)=n\}, \\ \{d',d'':d'+d''+d_+(BC \Rightarrow A)=d\}}} I(B_{i,k,n',d'}) I(C_{k,j,n'',d''}) w(BC \Rightarrow A)$$

$$I(GOAL) = \sum_{n,d} I(S_{0,L,n,d}) \left(1 - \frac{2n}{d + |y|}\right)$$

$$O(S_{0,N,n,d}) = \left(1 - \frac{2n}{d + |y|}\right)$$

$$O(A_{i,j,n,d}) = \sum_{k,B,C} \sum_{\substack{\{n',n'':n'-n''-n_+(AB \Rightarrow C)=n\}, \\ \{d',d'':d'-d''-d_+(AB \Rightarrow C)=d\}}} O(C_{i,k,n',d'}) I(B_{j,k,n'',d''}) w(AB \Rightarrow C) +$$

$$\sum_{k,B,C} \sum_{\substack{\{n',n'':n'-n''-n_+(BA \Rightarrow C)=n\}, \\ \{d',d'':d'-d''-d_+(BA \Rightarrow C)=d\}}} O(C_{k,j,n',d'}) I(B_{k,i,n'',d''}) w(BA \Rightarrow C)$$

Figure 3: State-split inside and outside recursions for computing softmax-margin with F-measure.

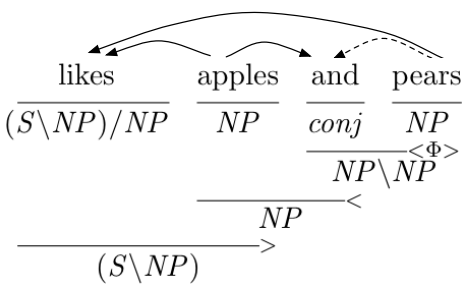

Figure 2: Example of flexible dependency realisation in CCG: Our parser (Clark and Curran, 2007) creates dependencies arising from coordination once all conjuncts are found and treats "and" as the syntactic head of coordinations. The coordination rule (Φ) does not yet establish the dependency "and - pears" (dotted line); it is the backward application ($<$) in the larger span, "apples and pears", that establishes it, together with "and - pears". CCG also deals with unbounded dependencies which potentially lead to more dependencies than words (Steedman, 2000); in this example a unification mechanism creates the dependencies "likes - apples" and "likes - pears" in the forward application ($>$). For further examples and a more detailed explanation of the mechanism as used in the C&C parser refer to Clark et al. (2002).

available within the current local structure, similar to those used by Taskar et al. (2004) for tracking constituent errors in a context-free parser. We design three simple losses to approximate precision, recall and F-measure on CCG dependency structures.

Let $T(y)$ be the set of parsing actions required to build parse y. Our decomposable approximation to precision simply counts the number of incorrect dependencies using the local dependency counts, $n_+(\cdot)$ and $d_+(\cdot)$.

$$DecP(y) = \sum_{t \in T(y)} d_+(t) - n_+(t) \tag{8}$$

To compute our approximation to recall we require the number of gold dependencies, $c_+(\cdot)$, which should have been introduced by a particular parsing action. A gold dependency is due to be recovered by a parsing action if its head lies within one child span and its dependent within the other. This yields a decomposed approximation to recall that counts the number of missed dependencies.

$$DecR(y) = \sum_{t \in T(y)} c_+(t) - n_+(t) \tag{9}$$

Unfortunately, the flexible handling of dependencies in CCG complicates our formulation of c_+, rendering it slightly more approximate. The unification mechanism of CCG sometimes causes dependencies to be realised later in the derivation, at a point when both the head and the dependent are in the same span, violating the assumption used to compute c_+ (see again Figure 2). Exceptions like this can cause mismatches between n_+ and c_+. We set $c_+ = n_+$ whenever $c_+ < n_+$ to account for these occasional discrepancies.

Finally, we obtain a decomposable approximation to F-measure.

$$DecF1(y) = DecP(y) + DecR(y) \tag{10}$$

4 Experiments

Parsing Strategy. CCG parsers use a pipeline strategy: we first multitag each word of the sentence with a small subset of its possible lexical categories using a *supertagger*, a sequence model over these categories (Bangalore and Joshi, 1999; Clark, 2002). Then we parse the sentence under the requirement that the lexical categories are fixed to those preferred by the supertagger. In our experiments we used two variants on this strategy.

First is the *adaptive supertagging* (AST) approach of Clark and Curran (2004). 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 (2004) show, most sentences can be parsed with very tight beams.

Reverse adaptive supertagging is a much less aggressive method that seeks only to make sentences parsable 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. Table 1 shows beam settings for both strategies.

Adaptive supertagging aims for speed via pruning while the reverse strategy aims for accuracy by exposing the parser to a larger search space. Although Clark and Curran (2007) found no actual improvements from the latter strategy, we will show that with our softmax-margin-trained models it can have a substantial effect.

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), which contains features over both normal-form derivations and CCG dependencies. The parser relies solely on the supertagger for pruning, using exact CKY for search over the pruned space. Training requires 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 2) 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. We evaluate on labelled and unlabelled predicate argument structure recovery and supertag accuracy.

4.1 Training with Maximum F-measure Parses

So far we discussed how to optimise towards task-specific metrics via changing the training objective. In our first experiment we change the *data* on which we optimise CLL. This is a kind of simple baseline to our later experiments, attempting to achieve the same effect by simpler means. Specifically, we use the algorithm of Huang (2008) to generate oracle F-measure parses for each sentence. Updating towards these oracle parses corrects the reachability problem in standard CLL training. Since the supertagger is used to prune the training forests, the correct parse is sometimes pruned away – reducing data utilisation to 91%. Clark and Curran (2007) correct for this by adding the gold tags to the parser input. While this increases data utilisation, it biases the model by training in an idealised setting not available at test time. Using oracle parses corrects this bias while permitting 99% data utilisation. The labelled F-score of the oracle parses lies at 98.1%. Though we expected that this might result in some improvement, results (Table 3) show that this has no

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 number of lexical categories considered for each word.

Condition	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
`C&C '07`	β	0.0045	0.0055	0.01	0.05	0.1		
	k	20	20	20	20	20		

Table 2: Beam step functions used for training: The first row shows the large scale settings used for most experiments and the standard C&C settings. (cf. Table 1)

	LF	LP	LR	UF	UP	UR	Data Util (%)
Baseline	87.40	87.85	86.95	93.11	93.59	92.63	91%
Max-F Parses	87.46	87.95	86.98	93.09	93.61	92.57	99%
CCGbank+Max-F	87.45	87.96	86.94	93.09	93.63	92.55	99%

Table 3: Performance on section 00 of CCGbank when comparing models trained with treebank-parses (Baseline) and maximum F-score parses (Max-F) using adaptive supertagging as well as a combination of CCGbank and Max-F parses. Evaluation is based on labelled and unlabelled F-measure (LF/UF), precision (LP/UP) and recall (LR/UR).

effect. However, it does serve as a useful baseline.

4.2 Training with the Exact Algorithm

We first tested our assumptions about the feasibility of training with our exact algorithm by measuring the amount of state-splitting. Figure 4 plots the average number of splits per span against the relative span-frequency; this is based on a typical set of training forests containing over 600 million states. The number of splits increases exponentially with span size but equally so decreases the number of spans with many splits. Hence the small number of states with a high number of splits is balanced by a large number of spans with only a few splits: The highest number of splits per span observed with our settings was 4888 but we find that the average number of splits lies at 44. Encouragingly, this enables experimentation in all but very large scale settings.

Figure 5 shows the distribution of n and d pairs across all split-states in the training corpus; since

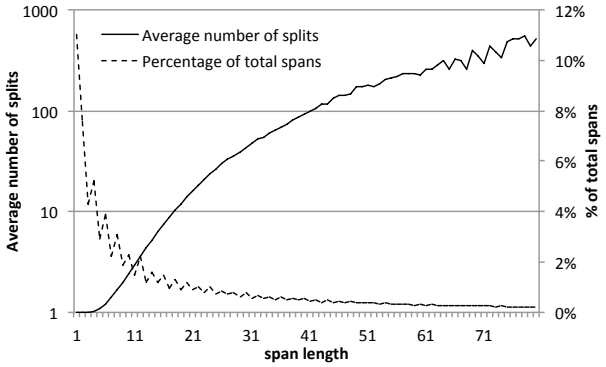

Figure 4: Average number of state-splits per span length as introduced by a sentence-level F-measure loss function. The statistics are averaged over the training forests generated using the settings described in §4.

n, the number of correct dependencies, over d, the number of all recovered dependencies, is precision, the graph shows that only a minority of states have either very high or very low precision. The range of values suggests that the softmax-margin criterion

will have an opportunity to substantially modify the expectations, hopefully to good effect.

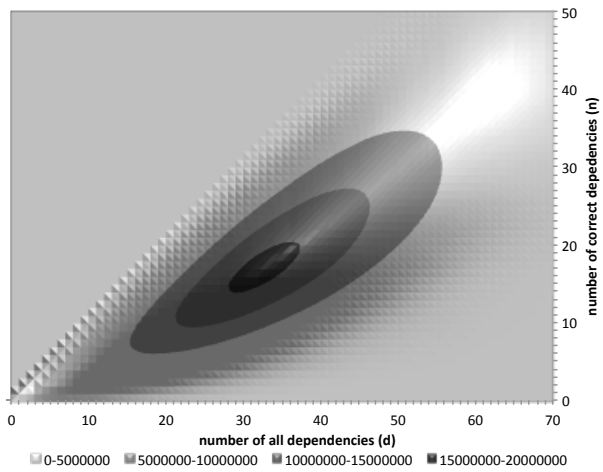

Figure 5: Distribution of states with d dependencies of which n are correct in the training forests.

We next turn to the question of optimization with these algorithms. Due to the significant computational requirements, we used the computationally less intensive normal-form model of Clark and Curran (2007) as well as their more restrictive training beam settings (Table 2). We train on all sentences of the training set as above and test with AST.

In order to provide greater control over the influence of the loss function, we introduce a multiplier τ, which simply amends the second term of the objective function (3) to:

$$\log \sum_{y \in Y(x^i)} \exp\{\theta^T f(x^i, y) + \tau \times \ell(y^i, y)\}$$

Figure 6 plots performance of the exact loss functions across different settings of τ on various evaluation criteria, for models restricted to at most 3000 items per chart at training time to allow rapid experimentation with a wide parameter set. Even in this constrained setting, it is encouraging to see that each loss function performs best on the criteria it optimises. The precision-trained parser also does very well on F-measure; this is because the parser has a tendency to perform better in terms of precision than recall.

4.3 Exact vs. Approximate Loss Functions

With these results in mind, we conducted a comparison of parsers trained using our exact and approximate loss functions. Table 4 compares their performance head to head when restricting training chart sizes to 100,000 items per sentence, the largest setting our computing resources allowed us to experiment with. The results confirm that the loss-trained models improve over a likelihood-trained baseline, and furthermore that the exact loss functions seem to have the best performance. However, the approximations are extremely competitive with their exact counterparts. Because they are also efficient, this makes them attractive for larger-scale experiments. Training time increases by an order of magnitude with exact loss functions despite increased theoretical complexity (§3.1); there is no significant change with approximate loss functions.

Table 5 shows performance of the approximate losses with the large scale settings initially outlined (§4). One striking result is that the softmax-margin trained models coax more accurate parses from the larger search space, in contrast to the likelihood-trained models. Our best loss model improves the labelled F-measure by over 0.8%.

4.4 Combination with Integrated Parsing and Supertagging

As a final experiment, we embed our loss-trained model into an integrated model that incorporates Markov features over supertags into the parsing model (Auli and Lopez, 2011). These features have serious implications on search: even allowing for the observation of Fowler and Penn (2010) that our CCG is weakly context-free, the search problem is equivalent to finding the optimal derivation in the weighted intersection of a regular and context-free language (Bar-Hillel et al., 1964), making search very expensive. Therefore parsing with this model requires approximations.

To experiment with this combined model we use loopy belief propagation (LBP; Pearl et al., 1985), previously applied to dependency parsing by Smith and Eisner (2008). A more detailed account of its application to our combined model can be found in (2011), but we sketch the idea here. We construct a graphical model with two factors: one is a distribu-

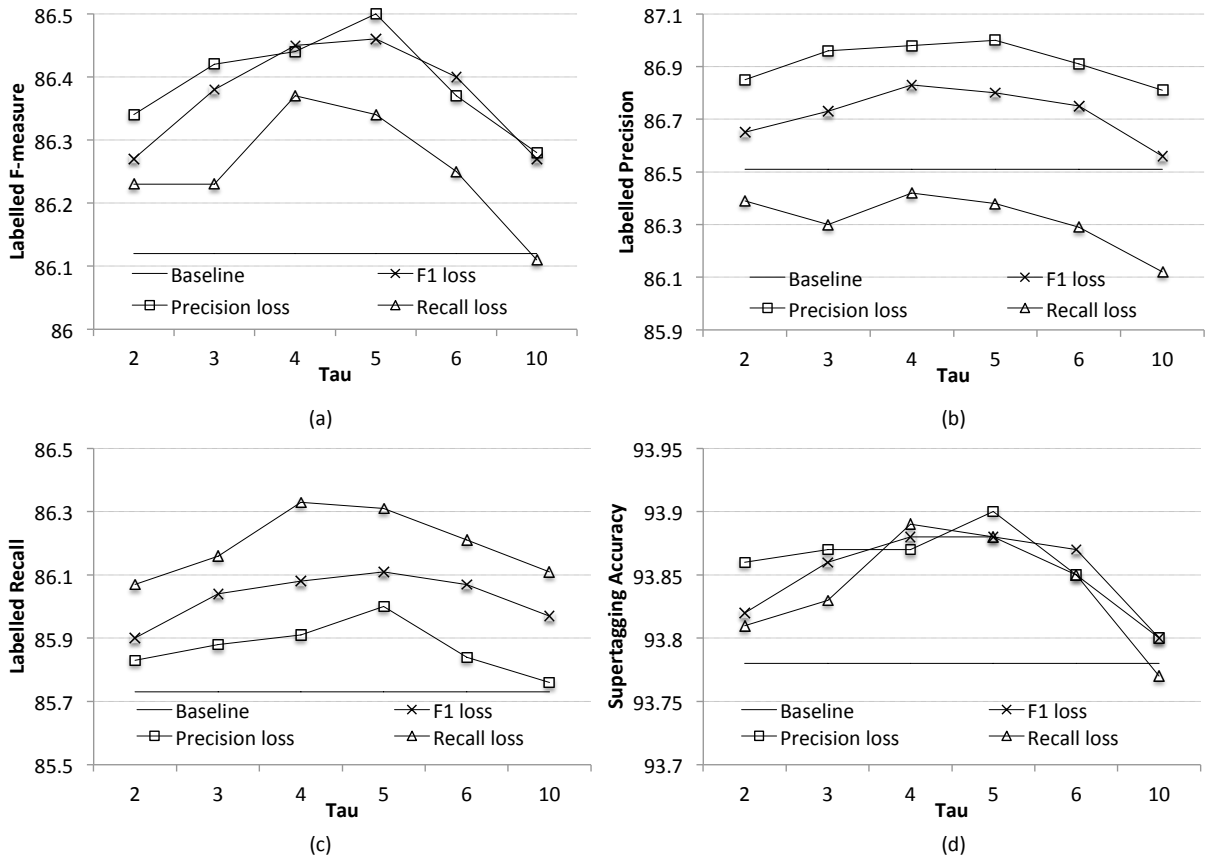

Figure 6: Performance of exact cost functions optimizing F-measure, precision and recall in terms of (a) labelled F-measure, (b) precision, (c) recall and (d) supertag accuracy across various settings of τ on the development set.

	section 00 (dev)						section 23 (test)					
	LF	LP	LR	UF	UP	UR	LF	LP	LR	UF	UP	UR
CLL	86.76	87.16	86.36	92.73	93.16	92.30	87.46	87.80	87.12	92.85	93.22	92.49
DecP	87.18	**87.93**	86.44	92.93	93.73	92.14	87.75	**88.34**	87.17	93.04	**93.66**	92.43
DecR	87.31	87.55	87.07	93.00	93.26	92.75	87.57	87.71	87.42	92.92	93.07	92.76
DecF1	87.27	87.78	86.77	93.04	93.58	92.50	87.69	88.10	87.28	93.04	93.48	92.61
P	87.25	87.85	86.66	92.99	**93.63**	92.36	**87.76**	88.23	87.30	**93.06**	93.55	92.57
R	**87.34**	87.51	**87.16**	92.98	93.17	**92.80**	87.57	87.62	**87.51**	92.92	92.98	**92.86**
F1	**87.34**	87.74	86.94	**93.05**	93.47	92.62	87.71	88.01	87.41	93.02	93.34	92.70

Table 4: Performance of exact and approximate loss functions against conditional log-likelihood (CLL): decomposable precision (DecP), recall (DecR) and F-measure (DecF1) versus exact precision (P), recall (R) and F-measure (F1). Evaluation is based on labelled and unlabelled F-measure (LF/UF), precision (LP/UP) and recall (LR/UR).

	section 00 (dev)						section 23 (test)					
	AST			Reverse			AST			Reverse		
	LF	UF	ST	LF	UF	ST	LF	UF	ST	LF	UF	ST
CLL	87.38	93.08	94.21	87.36	93.13	93.99	87.73	93.09	94.33	87.65	93.06	94.01
DecP	87.35	92.99	94.25	87.75	93.25	94.22	**88.10**	93.26	94.51	88.51	93.50	94.39
DecR	87.48	93.00	94.34	87.70	93.16	94.30	87.66	92.83	94.38	87.77	92.91	94.22
DecF1	**87.67**	**93.23**	**94.39**	**88.12**	**93.52**	**94.46**	88.09	**93.28**	**94.50**	**88.58**	**93.57**	**94.53**

Table 5: Performance of decomposed loss functions in large-scale training setting. Evaluation is based on labelled and unlabelled F-measure (LF/UF) and supertag accuracy (ST).

tion over supertag variables defined by a supertagging model, and the other is a distribution over these variables and a set of span variables defined by our parsing model.[5] The factors communicate by passing messages across the shared supertag variables that correspond to their marginal distributions over those variables. Hence, to compute approximate expectations across the entire model, we run forward-backward to obtain posterior supertag assignments. These marginals are passed as inside values to the inside-outside algorithm, which returns a new set of posteriors. The new posteriors are incorporated into a new iteration of forward-backward, and the algorithm iterates until convergence, or until a fixed number of iterations is reached – we found that a single iteration is sufficient, corresponding to a truncated version of the algorithm in which posteriors are simply passed from the supertagger to the parser. To decode, we use the posteriors in a minimum-risk parsing algorithm (Goodman, 1996).

Our baseline models are trained separately as before and combined at test time. For softmax-margin, we combine a parsing model trained with F1 and a supertagger trained with Hamming loss. Table 6 shows the results: we observe a gain of up to 1.5% in labelled F1 and 0.9% in unlabelled F1 on the test set. The loss functions prove their robustness by improving the more accurate combined models up to 0.4% in labelled F1. Table 7 shows results with automatic part-of-speech tags and a direct comparison with the Petrov parser trained on CCGbank (Fowler and Penn, 2010) which we outpeform on all metrics.

[5]These complex factors resemble those of Smith and Eisner (2008) and Dreyer and Eisner (2009); they can be thought of as case-factor diagrams (McAllester et al., 2008)

5 Conclusion and Future Work

The softmax-margin criterion is a simple and effective approach to training log-linear parsers. We have shown that it is possible to compute exact sentence-level losses under standard parsing metrics, not only approximations (Taskar et al., 2004). This enables us to show the effectiveness of these approximations, and it turns out that they are excellent substitutes for exact loss functions. Indeed, the approximate losses are as easy to use as standard conditional log-likelihood.

Empirically, softmax-margin training improves parsing performance across the board, beating the state-of-the-art CCG parsing model of Clark and Curran (2007) by up to 0.8% labelled F-measure. It also proves robust, improving a stronger baseline based on a combined parsing and supertagging model. Our final result of 89.3%/94.0% labelled and unlabelled F-measure is the best result reported for CCG parsing accuracy, beating the original C&C baseline by up to 1.5%.

In future work we plan to scale our exact loss functions to larger settings and to explore training with loss functions within loopy belief propagation. Although we have focused on CCG parsing in this work, we expect our methods to be equally applicable to parsing with other grammar formalisms including context-free grammar or LTAG.

Acknowledgements

We would like to thank Stephen Clark, Christos Christodoulopoulos, Mark Granroth-Wilding, Gholamreza Haffari, Alexandre Klementiev, Tom Kwiatkowski, Kira Mourao, Matt Post, and Mark Steedman for helpful discussion related to this work and comments on previous drafts, and the

	section 00 (dev)						section 23 (test)					
	AST			Reverse			AST			Reverse		
	LF	UF	ST	LF	UF	ST	LF	UF	ST	LF	UF	ST
CLL	87.38	93.08	94.21	87.36	93.13	93.99	87.73	93.09	94.33	87.65	93.06	94.01
BP	87.67	93.26	94.43	88.35	93.72	94.73	88.25	93.33	94.60	88.86	93.75	94.84
+DecF1	**87.90**	**93.40**	**94.52**	**88.58**	**93.88**	**94.79**	88.32	93.32	94.66	89.15	93.89	94.98
+SA	87.73	93.28	94.49	88.40	93.71	94.75	**88.47**	**93.48**	**94.71**	**89.25**	**93.98**	**95.01**

Table 6: Performance of combined parsing and supertagging with belief propagation (BP); using decomposed-F1 as parser-loss function and supertag-accuracy (SA) as loss in the supertagger.

	section 00 (dev)						section 23 (test)					
	LF	LP	LR	UF	UP	UR	LF	LP	LR	UF	UP	UR
CLL	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	85.73	92.34	92.64	92.04
BP	86.45	86.75	86.17	92.60	92.92	92.29	86.84	87.08	86.61	92.57	92.82	92.32
+DecF1	**86.73**	**87.07**	**86.39**	**92.79**	**93.16**	**92.43**	87.08	87.37	86.78	92.68	93.00	92.37
+SA	86.51	86.86	86.16	92.60	92.98	92.23	**87.20**	**87.50**	**86.90**	**92.76**	**93.08**	**92.44**

Table 7: Results on automatically assigned POS tags. *Petrov I-5* is based on the parser output of Fowler and Penn (2010); evaluation is based on sentences for which all parsers returned an analysis.

anonymous reviewers for helpful comments. We also acknowledge funding from EPSRC grant EP/P504171/1 (Auli); and the resources provided by the Edinburgh Compute and Data Facility.

A Computing F-Measure-Augmented Expectations at the Corpus Level

To compute exact corpus-level expectations for softmax-margin using F-measure, we add an additional transition before reaching the $GOAL$ item in our original program. To reach it, we must parse every sentence in the corpus, associating statistics of aggregate $\langle n, d\rangle$ pairs for the entire training set in intermediate symbols $\Gamma^{(1)}...\Gamma^{(m)}$ with the following inside recursions.

$$
\begin{aligned}
I(\Gamma^{(1)}_{n,d}) &= I(S^{(1)}_{0,|x^{(1)}|,n,d}) \\
I(\Gamma^{(\ell)}_{n,d}) &= \sum_{n',n'':n'+n''=n} I(\Gamma^{(\ell-1)}_{n',d'}) I(S^{(\ell)}_{0,N,n'',d''}) \\
I(GOAL) &= \sum_{n,d} I(\Gamma^{(m)}_{n,d}) \left(1 - \frac{2n}{d+|y|}\right)
\end{aligned}
$$

Outside recursions follow straightforwardly. Implementation of this algorithm would require substantial distributed computation or external data structures, so we did not attempt it.

References

M. Auli and A. Lopez. 2011. A Comparison of Loopy Belief Propagation and Dual Decomposition for Integrated CCG Supertagging and Parsing. In *Proc. of ACL*, June.

J. K. Baker. 1979. Trainable grammars for speech recognition. *Journal of the Acoustical Society of America*, 65.

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

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

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 and J. Hockenmaier. 2002. Evaluating a Wide-Coverage CCG Parser. In *Proceedings of the LREC 2002 Beyond Parseval Workshop*, pages 60–66, Las Palmas, Spain.

S. Clark, J. Hockenmaier, and M. Steedman. 2002. Building deep dependency structures with a wide-coverage CCG parser. In *Proc. of ACL*.

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

M. Dreyer and J. Eisner. 2009. Graphical models over multiple strings. 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*.

K. Gimpel and N. A. Smith. 2010a. Softmax-margin CRFs: training log-linear models with cost functions. In *HLT '10: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

K. Gimpel and N. A. Smith. 2010b. Softmax-margin training for structured log-linear models. Technical Report CMU-LTI-10-008, Carnegie Mellon University.

J. Goodman. 1996. Parsing algorithms and metrics. In *Proc. of ACL*, pages 177–183, Jun.

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*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*, pages 282–289.

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.

D. McAllester. 1999. On the complexity analysis of static analyses. In *Proc. of Static Analysis Symposium*, volume 1694/1999 of *LNCS*. Springer Verlag.

F. J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL*, Jul.

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

D. Povey and P. Woodland. 2008. Minimum phone error and I-smoothing for improved discrimative training. In *Proc. of ICASSP*.

F. Sha and L. K. Saul. 2006. Large margin hidden Markov models for automatic speech recognition. In *Proc. of NIPS*.

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

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

B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. 2004. Max-margin parsing. In *Proc. of EMNLP*, pages 1–8, Jul.