ON MAXIMIZING METRICS FOR SYNTACTIC DISAMBIGUATION

Khalil Sima'an[*]
Language and Inference Technology Group
Institute for Logic, Language and Computation (ILLC)
University of Amsterdam, The Netherlands
simaan@science.uva.nl

Abstract

Given a probabilistic parsing model and an evaluation metric for scoring the match between parse-trees, e.g., PARSEVAL [Black et al., 1991], this paper addresses the problem of *how to select the on average best scoring parse-tree for an input sentence*. Common wisdom dictates that it is optimal to select the parse with the highest probability, regardless of the evaluation metric. In contrast, the Maximizing Metrics (MM) method [Goodman, 1998, Stolcke et al., 1997] proposes that an algorithm that optimizes the evaluation metric itself constitutes the optimal choice. We study the MM method within parsing. We observe that the MM does *not* always hold for tree-bank models, and that optimizing weak metrics is not interesting for semantic processing. Subsequently, we state an alternative proposition: the optimal algorithm must maximize the metric that scores parse-trees according to linguistically relevant features. We present new algorithms that optimize metrics that take into account increasingly more linguistic features, and exhibit experiments in support of our claim.

1 Introduction

A probabilistic grammar associates a probability $P(T|U)$ with every parse-tree T and sentence U. These conditional probabilities enable the selection of a single preferred parse-tree for the input sentence. The majority of existing work is based on the paradigm of selecting a parse T^* that has the Maximum A Posteriori (MAP) probability, i.e., $T^* = arg\max_T P(T|U)$. The latter approach is known as the Most Probable Parse (MPP) algorithm, and has been used in various models, e.g., [Collins, 1997].

Goodman [Goodman, 1996] proposes that it is better to employ a disambiguation algorithm that *optimizes the evaluation metric directly*, than use the general MPP approach. For instance, when a parser will be evaluated under the PARSEVAL [Black et al., 1991] measures of Labeled Recall (and Precision), Goodman proposes that the parser can better employ a disambiguation algorithm that Maximizes the expected Labeled constituent Recall rate (also called the MLR algorithm). We refer here to Goodman's proposition with the name the *Maximizing (Evaluation) Metrics hypothesis*. The same hypothesis underlies the work in [Stolcke et al., 1997], who develop new algorithms for speech-recognition to improve over MAP algorithms. Besides the desire to improve the accuracy, there are other reasons for exploring alternatives for the MPP/MAP algorithms. For instance, computing the MPP under the DOP model [Bod, 1998] is NP-Complete [Sima'an, 2002]. Analogously, computing the MAP word-sequence from an input word-graph under a Hidden Markov Model (HMM) is also NP-Complete [Goodman, 1998].

[*]I thank Joshua Goodman, Philip Resnik, Detlef Prescher and the IWPT reviewers for their corrections and comments.

The present paper addresses the question which disambiguation algorithm scores best under a given evaluation scheme and under what conditions? After providing some preliminaries (section 2), we specify the Maximizing Metrics hypothesis and algorithms (section 3). Subsequently, we address the question whether the Maximizing Metrics hypothesis holds with respect to *practical* probabilistic models acquired from tree-banks (section 4). In particular, we concentrate on a study of the MLR and the MPP algorithms. We specify the conditions on the probability models under which the MLR constitutes a suitable alternative to the MPP, and provide evidence that these conditions are not always met by existing models obtained from tree-banks. Subsequently we present new algorithms that optimize tree-matching metrics that take increasingly more linguistically relevant features of trees into account (section 5). These algorithms are more interesting than the LR algorithm because they provide better results than the LR on a range of more interesting metrics (from a linguistic point of view). We show empirically that the new algorithms often improve on both the MPP and the MLR algorithms (section 6). Finally, in section 7 we discuss the conclusions from this paper.

2 Preliminaries

For reasons of exposition, throughout this paper we assume Probabilistic Context-Free Grammars (PCFGs) as the formal model. The results apply to various other models (see [Goodman, 1998]), and in particular they are interesting for the Data Oriented Parsing (DOP) model [Bod, 1998], where the general disambiguation approach is NP-Complete [Sima'an, 2002].

The notation w_i^j stands for the ordered sequence $w_i, \cdots, w_j$. The notation $\Upsilon[A]$, where A is a proposition, stands for the indicator function: $if\ (A == true)\ \Upsilon[A] = 1$, and otherwise $\Upsilon[A] = 0$.

Throughout the paper V_N and V_T stand for the finite sets of nonterminal and terminal symbols respectively. Also the set $\mathcal{T}$ stands for the set of all finite branching parse-trees over V_N and V_T. To descriminate between nonterminal and terminal symbols, we use $A, B, C, \cdots$ for nonterminal symbols, and $a, b, c, \cdots$ for terminals. We also use $\alpha, \beta, \cdots$ for sequences of symbols from $(V_N \cup V_T)^+$.

Probabilistic Context-Free Grammars: As usual, a PCFG is a five-tuple (V_N, V_T, S, $\mathcal{R}$, P), with $S \in V_N$ being the start symbol and $\mathcal{R} \in V_N \times (V_N \cup V_T)^+$ is the set of productions. We denote production using the arrow e.g. $A \rightarrow \alpha$, and use the notation $A \overset{+}{\Rightarrow} \alpha$ for all derivations leading from A to α by at least one application of a production. We write $\{A \overset{+}{\Rightarrow} \alpha\}$ to *stress* the fact that this notation specifies a set of derivations.

The function $P : \mathcal{R} \rightarrow (0, 1]$ is a probability mass function such that $\forall A \in V_N : \sum_{A \rightarrow \alpha \in V_N} P(A \rightarrow \alpha) = 1$. The latter implies that the probability of a derivation involving a sequence of productions is estimated as the product of the probabilities of these productions (thereby assuming independence between the productions).

We will use the notation $L(G)$ and $\mathcal{T}(G)$ to denote respectively the string and tree languages of the

PCFG G. We overload this notation sometimes by writing $\mathcal{T}(u)$, where u is an utterance, to denote the set of parses that the grammar (which is kept implicit) generates for sentence u.

Parse-tree representation ($\mathcal{C}(T)$): Let there be given a parse-tree T such that the non-leaf nodes are labeled with non-terminal symbols (V_N) and the leaf nodes are labeled with terminal symbols (V_T). Also let w_1^n be the yield of T, i.e. the sequence of labels of the leaf-nodes (from left to right). It is common to represent the parse-tree T as a set of constituents $\mathcal{C}(T)$ as follows: each non-terminal node μ (i.e. constituent) in T is represented by a tuple $\langle i, X, j\rangle$, where X is the label of μ and w_i^{j-1} is the yield of the subtree of T that is rooted at μ. For simplifying the notation, however, and unless stated otherwise, we will ommit $\mathcal{C}()$ and write T instead, under the understanding that this *set of constituents* representation is assumed implicitly.

Evaluation metrics: Current parser evaluation practice is based on the PARSEVAL [Black et al., 1991] measures of Labeled Recall and Precision, which we define next together with the Exact Tree Match Rate. Let there be given a gold-standard test-set (a multiset) $\langle \mathcal{U}_1, T_C^1\rangle, \cdots, \langle \mathcal{U}_n, T_C^n\rangle$, whereby $\mathcal{U}_i$ stands for the i-th sentence and T_C^i for the tree-bank parse-tree for that sentence. Also let the parser output[1] for the same sequence of sentences be $T_g^1, \cdots, T_g^n$ (i.e. T_g^i is the parser's "guessed" parse-trees for sentence $\mathcal{U}_i$). Let us denote the multisets of parses as follows $\{T_C^i\} = \{T_C^1, \cdots, T_C^n\}$, and $\{T_g^i\} = \{T_g^1, \cdots, T_g^n\}$. The labeled Constituent Recall and Precision Rates (LR/ LP) and Exact Tree Match Rate (EM) are defined as follows:

$$Labeled\ Recall \qquad LR(\{T_g^i\},\{T_C^i\}) \overset{def}{=} \frac{\sum_i |T_C^i \cap T_g^i|}{\sum_i |T_C^i|} = \frac{\sum_i \sum_{\{i,X,j\}\in T_g} \Upsilon[\{i,X,j\}\in T_C]}{\sum_i |T_C^i|}$$

$$Labeled\ Precision \qquad LP(\{T_g^i\},\{T_C^i\}) \overset{def}{=} \frac{\sum_i |T_C^i \cap T_g^i|}{\sum_i |T_g^i|} = \frac{\sum_i \sum_{\{i,X,j\}\in T_g} \Upsilon[\{i,X,j\}\in T_C]}{\sum_i |T_g^i|}$$

$$Exact\ Match \qquad EM(\{T_g^i\},\{T_C^i\}) \overset{def}{=} \frac{\sum_i \Upsilon[T_C^i == T_g^i]}{n}$$

where $|T|$ denotes the cardinality of the set of constituents that represents T.

3 Maximizing Metrics

We start out by formalizing the Maximizing Metrics hypothesis in order to facilitate accurate discussion. Let there be given a conditional probability mass function $P(T|w_1^n)$, where $T \in \mathcal{T}$ and $w_1^n \in {V_T}^+$, and an evaluation metric $Match : \mathcal{T} \times \mathcal{T} \rightarrow [0, 1]$. Now let $\mathcal{T}(w_1^n)$ represent the finite set of parse-trees that have a yield equal to the sentence $w_1^n = w_1, \cdots, w_n$, from which we want to select the preferred parse-tree for w_1^n.

The Maximizing Metrics hypothesis states that the following parse-tree T^* performs optimally under

[1]The parser output is not allowed to be empty.

the evaluation metric $Match$:

$$T^{*} \;=\; \arg\max_{T_g \in \mathcal{T}(w_1^n)} \mathcal{E}_P(Match(T_g, T_C)) \;=\; \arg\max_{T_g \in \mathcal{T}(w_1^n)} \sum_{T_C \in \mathcal{T}} P(T_C|w_1^n) \times Match(T_g, T_C)$$

where $\mathcal{E}_P$ is the expectation value under $P(T_C|w_1^n)$. In fact, this also means that an algorithm that maximizes the expectation value of the metric $Match$ will perform at least as good as algorithms that maximize other, possibly more stringent, metrics. Next we derive two example algorithms, the MPP and the MLR algorithms, and review this hypothesis by inspecting how these algorithms relate to one another.

3.1 Most Probable Parse (MPP)

The Most Probable Parse (MPP) can be derived as follows:

$$(1) \qquad T^{mpp} \;=\; \arg\max_{T_g \in \mathcal{T}(w_1^n)} \mathcal{E}_P(EM(T_g, T_C))$$

$$(2) \qquad \;=\; \arg\max_{T_g \in \mathcal{T}(w_1^n)} \sum_{T_C} P(T_C|w_1^n)\; \Upsilon_{[T_g == T_C]} \;\approx\; \arg\max_{T_g \in \mathcal{T}(w_1^n)} P(T_g|w_1^n)$$

In the first equation we use the definition of expectation and of the EM rate. The last step is an approximation because instead of comupting no parse at all when ($\forall T_g \in \mathcal{T}(w_1^n) : (T_g \neq T_C)$), the algorithm computes the most probable available parse.

It is common to assume that $P(T_g|w_1^n)$ is computed by a language model, e.g., a PCFG. Formula 2 can be implemented in a polynomial time algorithm under PCFGs, e.g. [Manning and Schutze, 1999], but is known to be NP-Complete under the DOP model.

3.2 Maximum Labeled Recall rate Parse (MLR)

The MLR algorithm (cf. [Goodman, 1998], pp. 104) is derived from the following optimization formula:

$$(3) \qquad T^{mlr} \;=\; \arg\max_{T_g \in \mathcal{T}(w_1^n)} \mathcal{E}_P\left(\frac{|T_C \cap T_g|}{|T_C|}\right)$$

Like Goodman, we assume binary branching trees in order to keep the discussion simple. For any two binary parse-trees T_C^1 and T_C^2 of the same input sentence: $|T_C^1| = |T_C^2|$. Therefore, we may substitute for $|T_C|$ a new function $N(w_1^n)$, which denotes the number of constituents in any binary tree of w_1^n. Using the definition of $|T_g \cap T_C|$, and by rearranging the summations, equation 3 can be rewritten in steps as follows

$$T^{mlr} \;=\; \arg\max_{T_g \in \mathcal{T}(w_1^n)} \sum_{T_C} P(T_C|w_1^n)\, \frac{|T_g \cap T_C|}{N(w_1^n)} \;=\; \arg\max_{T_g} \sum_{T_C} P(T_C|w_1^n) \sum_{\langle i,X,j\rangle \in T_g} \frac{\Upsilon_{[\langle i,X,j\rangle \in T_C]}}{N(w_1^n)}$$

$$\;=\; \arg\max_{T_g} \sum_{\langle i,X,j\rangle \in T_g} \frac{\sum_{T_C} P(T_C|w_1^n)\; \Upsilon_{[\langle i,X,j\rangle \in T_C]}}{N(w_1^n)}$$

For convineince, we define $g(i,X,j) \stackrel{def}{=} \sum_{T_C} P(T_C|w_1^n)\ \Upsilon_{[\langle i,X,j\rangle \in T_C]})$. Furthermore, we define the Expected Labeled Recall rate ($\mathcal{ER}_{\mathcal{LR}}$):

(4) $$\mathcal{ER}_{\mathcal{LR}}(T_g|w_1^n) \quad \stackrel{def}{=} \quad \sum_{\langle i,X,j\rangle \in T_g} \frac{g(i,X,j)}{N(w_1^n)}$$

Given a PCFG μ with a start symbol S, Goodman observes that, under a common assumption[2], $g(i,X,j)$ is approximately equal to the probability of the set of all PCFG derivations of w_1^n that pass through constituent $\langle i,X,j\rangle$, i.e. $P_\mu(\{S \stackrel{+}{\Rightarrow} w_1^{i-1}Xw_j^n\}|w_1^n)$. Therefore:

$$\begin{aligned} g(i,X,j) \quad &\approx \quad P_\mu(\{S \stackrel{+}{\Rightarrow} w_1^{i-1}Xw_j^n\}|w_1^n) \\ &= \quad \frac{P_\mu(\{S \stackrel{+}{\Rightarrow} w_1^{i-1}Xw_j^n\})\ P_\mu(\{X \stackrel{+}{\Rightarrow} w_i^{j-1}\})}{P_\mu(w_1^n)} \quad = \quad \frac{\mathcal{I}_\mu(i,X,j) \times \mathcal{O}_\mu(i,X,j)}{\mathcal{I}_\mu(1,S,n+1)} \end{aligned}$$

where: $\mathcal{I}_\mu(i,X,j) \stackrel{def}{=} P_\mu(S \stackrel{+}{\Rightarrow} w_1^{i-1}Xw_j^n)$ and $\mathcal{O}_\mu(i,X,j) \stackrel{def}{=} P_\mu(X \stackrel{+}{\Rightarrow} w_i^{j-1})$. For calculating the $\mathcal{I}_\mu$ and $\mathcal{O}_\mu$ values, there exist well-known algorithms, usually used together as the machinary underlying the Inside-Outside algorithm for parameter estimation [Lari and Young, 1991, Goodman, 1998]. The MLR algorithm exploits the Inside and Outside probabilities for disambiguation[3]. Using a dynamic programming algorithm over a chart, these calculations can be done in time cubic in sentence length.

4 Which algorithm for which metric?

It is known that an algorithm that selects the MPP under the given model for every input sentence, necessarily minimizes the expected percentage of parses containing any errors at all [Duda et al., 2001]. The latter holds when the model is a good approximation of the true distribution over parse-trees and sentences. As [Goodman, 1998] observes, however, the MPP is not guaranteed to achieve optimal *expected* LR rate. The main question now concerns the Maximizing Metrics (MM) hypothesis: does maximizing the expected $\mathcal{ER}_{\mathcal{LR}}$ guarantee minimal error in terms of LR? We are interested here in actual models of natural language parsing; these models often tend to be *only weak approximations* of the true distributions, e.g. the models acquired from large tree-banks. Does the MM hypothesis hold under these circumstances? We show here that the MM hypothesis need not always hold and that the empirically optimal algorithm (in terms of some evaluation metric) is one that strikes a balance between the amount of available data and the amout of linguistic knowledge that it employs for disambiguation. Furthermore, we argue that for actual applications it might be more beneficial to employ algorithms that achieve good scores on a range of different evaluation metrics than on a single metric such as LR.

4.1 Conditions for the MLR to perform

It is known that linguistic parse-trees express dependencies between different parts and constituents. A suitable model captures these dependencies through the probability of co-occurence of constituents. In

[2]That the tree-bank is a sample from the given PCFG.
[3]We stress that the $Inside\text{-}Outside$ calculations in the MLR algorithm do not concern parameter estimation but the calaculation of the LR metric during parsing.

contrast, the MLR algorithm, as we show next, might relax these dependencies to any extent, because it optimizes an evaluation metric that does so too.

Define the set of derivations that pass through constituent $A = \langle i, X, j \rangle$ of sentence w_1^n by the notation $\mathcal{D}^A \overset{def}{=} \{S \overset{+}{\Rightarrow} w_1^{i-1} X w_j^n\}$. Given a parse-tree (represented as a set of constituents) $\mathcal{C}(T) = \{A_m \mid 1 \leq m \leq n\}$, where $A_m = \langle i_m, X_m, j_m \rangle$ for all m, we know that

(5) $$P(T|w_1^n) \quad = \quad P(\bigcap_{m=1}^{n} \mathcal{D}^{A_m} \mid w_1^n)$$

In words this simply says that the probability of a parse-tree T given the sentence w_1^n is equal to the probability of the intersection between all sets of derivations $\mathcal{D}^{A_1}, \ldots, \mathcal{D}^{A_n}$ (which results in the case of a PCFG in a single derivation that generates T).

In contrast, according to equation 4, the MLR algorithm compares trees using

(6) $$\mathcal{ER}_{\mathcal{LR}}(T|w_1^n) \quad = \quad \frac{\sum_{m=1}^{n} P(\mathcal{D}^{A_m}|w_1^n)}{n}$$

Note that $\mathcal{ER}_{\mathcal{LR}}(T|w_1^n)$ is the average probability of the cooccurence of any of the constituents of T with w_1^n. The $\mathcal{ER}_{\mathcal{LR}}$, being an average over many nodes, will often be robust to data sparseness. As an approximation of tree probability, however, $\mathcal{ER}_{\mathcal{LR}}(T|w_1^n)$ might not be suitable for PCFGs of natural language. This is because the MLR algorithm makes the assumption

$$P(\bigcap_{m=1}^{n} \mathcal{D}^{A_m}|w_1^n) \quad \approx \quad \frac{\sum_{m=1}^{n} P(\mathcal{D}^{A_m}|w_1^n)}{n}$$

This assumption need not always hold (as $P(A \cap B) = P(A) + P(B) - P(A \cup B)$). For example, the dependency between two constituents, one on the right-hand side and the other on the left-hand side of a production, is not necessarily captured by the $\mathcal{ER}_{\mathcal{LR}}$. Nevertheless, for disambiguation it is sufficient that the rank order of parse trees using the $\mathcal{ER}_{\mathcal{LR}}$ is approximately the same as the actual rank order.

4.2 A counter example

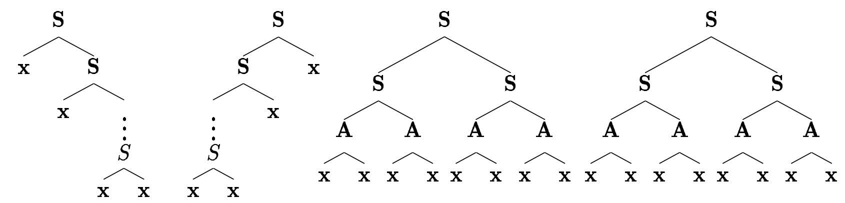

Figure 1: A corpus with 4 trees for the sentence $x\ x\ x\ x\ x\ x\ x\ x$: the left most tree and the one directly to its right are supposed to be right linear and left linear respectively, but some of the internal nodes labeled S (7 in total in each of them) are omitted for space reasons.

In figure 1, we exhibit a toy corpus of correct parse-trees, where S and A are the non-terminals and x the terminal symbol. Table 1 shows the rules (and their respective probabilities) of the PCFG obained from this corpus (referred to as “tree-bank grammar” [Charniak, 1996]).

$S \rightarrow x\, x$	$2/20$	$S \rightarrow S\, x$	$6/20$	$S \rightarrow A\, A$	$2/10$
$S \rightarrow x\, S$	$6/20$	$S \rightarrow S\, S$	$2/20$	$A \rightarrow x\, x$	$8/8$

Table 1: PCFG rules with their probabilities

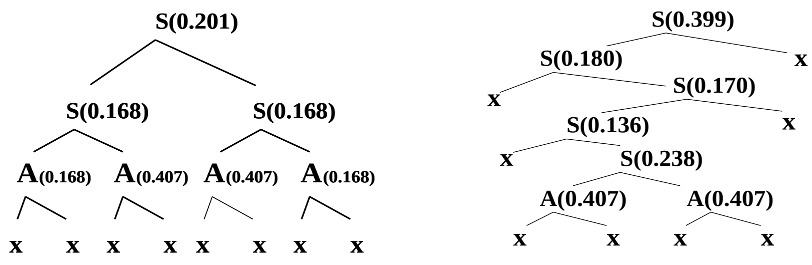

Figure 2: **(Left)** the MPP, **(Right)** the LRR

Let us consider the input sentence $x^8 = x\ x\ x\ x\ x\ x\ x\ x$ for which the PCFG generates 500 different parse-trees. Naturally, there is a clear preference in the corpus for the balanced structures shown at the right in figure 1. According to our PCFG, the MPP, the left tree in figure 2, has probability: $P(S \rightarrow S\ S|S) \times P(S \rightarrow A\ A|S)^2 \times P(A \rightarrow x\ x|A)^4$, i.e. $1/10 \times 1/5^2 \times 1^4 = 1/250$. In contrast, the probability of the tree to the right of figure 2 is: $3/10^2 \times 3/10^2 \times 1/5 = 81/5000$. Remarkably, the latter parse-tree is preferred by the MLR algorithm. Figure 2 shows both trees with the nodes decorated with the respective $g(i, X, j)$ rates[4]. The MPP has a labeled recall rate of ≈ 1.69, and the LR parse has a higher rate of ≈ 1.94.

What happened in this example? The MLR algorithm tends to compute the $\mathcal{ER}_{\mathcal{LR}}$ as the average probability of an individual constituent to co-coccur with the utterance. The co-occurence of the different constituents that constitute a parse is not taken into account directly. As the grammar overgenerates badly, the average over individual constituents becomes a bad estimate of the probability of co-occurence of the multiple consituents together with the utterance.

Intermediate summary: When the language model is too weak, the MLR algorithm is not guaranteed to be optimal with respect to the LR metric. Furthermore, it has often been observed [Carroll et al., 2002] that the LR evaluation metric does not capture qualities of trees that might be linguistically relevant. The LR metric takes only superficial aspects of parse-trees into account. Therefore, we propose here that in *practical situations*, i.e. tree-bank grammars, optimizing metrics that are suitable for expressing some linguistically relevant features of parse-trees might deliver better results than the MLR , both in terms of LR and in terms of *other, more interesting metrics*. In the remainder of this paper we explore this hypothesis. We present various algorithms and empirical comparisons to support the preceding study.

[4]The calculations are space demanding to repeat here: one proceeds by looking up for every constituent all parse-trees in which it appears, sum up the probabilities of these parse-trees and devide this by the sentence probability (4.539801e-02) to obtain the $g(i, X, j)$ for that constituent. Alternatively, we use here a computer program implementing the Inside and Outside algorithms to do this.

5 New algorithms

We present alternatives for the MLR : algorithms that constitute a middle way between the MPP and the MLR algorithms because they take into account more of the linguistically relevant features of trees. These algorithms, we claim, are more interesting from a practical point of view than the MLR algorithm for two reasons: (1) they are less prone to the weakness of the model than the MLR because they are slightly stricter, and (2) they deliver more interesting parse-trees because they take into consideration more linguistic aspects than the MLR . For example, the new algorithms might be more suitable than the MLR for applications where the parser-output is used for computing compositional semantics. We concentrate on two kinds of linguistic dependencies in parse-trees, to exemplify our point:

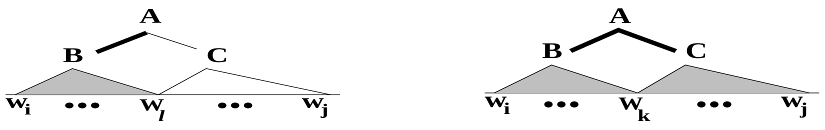

Figure 3: Two kinds of relations

Parent-child: These are abstractions of bilexical-dependencies [Collins, 1997]. Let $\langle i, A, j\rangle \in \mathcal{C}(T)$ be a constituent in T and let $\langle k, B, l\rangle \in \mathcal{C}(T)$, for $i \leq k < l \leq j$, be its left/right child constituent (e.g. left tree in figure 5). Let us denote this relation between a parent and its child by $\langle i, A_B^O, j\rangle \in T$, where $O \in \{left, right\}$. To measure the amount of match between parse-trees at the level of parent-child dependencies we have to adapt their representation as follows: $\mathcal{PC}(T) \overset{def}{=} \{\langle i, A_B^O, j\rangle \mid \forall\langle i, A_B^O, j\rangle \in T\}$. We now define a new measure of tree-match called the Parent-Child match as follows: $PC(T_g, T_C) \overset{def}{=} \mathcal{E}(\frac{|\mathcal{PC}(T_C)\cap\mathcal{PC}(T_g)|}{|\mathcal{PC}(T_C)|})$.

Frames: As proxi for sub-categorization frames, the notion of a frame in Phrase-Structure coincides with a context-free production. Often it is assumed that frames constitute the syntactic units over which semantic interpretation can take place. Let $\langle i, A, j\rangle \in \mathcal{C}(T)$ and let $\langle k+1, C, j\rangle$, $\langle i, B, k\rangle \in \mathcal{C}(T)$ be its child nodes, where $i < k < j$ (right tree in figure 5). We will write this relation (a Context-Free production) as follows: $\langle i, A \rightarrow B\ C, j\rangle \in T$. Another representation of T is given by $\mathcal{CF}(T) \overset{def}{=} \{\langle i, A \rightarrow B\ C, j\rangle \mid \forall\langle i, A \rightarrow B\ C, j\rangle \in T\}$. A correspoding tree-match metric $CF(T_g, T_C) \overset{def}{=} \mathcal{E}(\frac{|\mathcal{CF}(T_C)\cap\mathcal{CF}(T_g)|}{|\mathcal{CF}(T_C)|})$.

From these definitions we can derive two new algorithms, respectively the $MPCh$ and the $MCFP$ algorithms, by optimizing the respective metrics, just like the MLR algorithm was derived in section 3.2 by optimizing the LR metric. Formally speaking, we may reuse the MLR algorithm if we modify the PCFG such that the non-terminal set consists of new non-terminals each representing a parent-child dependency or a PCFG production, respectively. Hence, similar to the $\mathcal{ER}_{\mathcal{LR}}(T_g|w_1^n)$ (see equation 4), we will have two terms to optimize

$$\mathcal{ER}_{\mathcal{PC}}(T_g|w_1^n) = \sum_{\langle i,X_Y^O,j\rangle\in T_g} \frac{\sum_{T_C} P(T_C|w_1^n)\ \Upsilon[\langle i,X_Y^O,j\rangle\in\mathcal{PC}(T_C)]}{|\mathcal{PC}(T_C)|}$$

$$\mathcal{ER}_{\mathcal{CF}}(T_g|w_1^n) = \sum_{\langle i,X\rightarrow Y,j\rangle\in T_g} \frac{\sum_{T_C} P(T_C|w_1^n)\ \Upsilon[\langle i,X\rightarrow Y,j\rangle\in\mathcal{CF}(T_C)]}{|\mathcal{CF}(T_C)|}$$

6 Empirical experiments

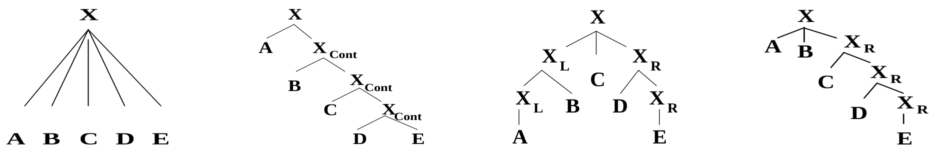

Figure 4: (From left to right) original n-ary rule, its right linearization, head-driven linearization when C is head-child, and head-driven linearization when B is head-child.

Our experiments deviate only slightly from Goodman's experiments in that here the disambiguation algorithms select the preferred parse from the space of trees that are generated by the PCFG at hand (i.e. we do not allow the MLR to assemble extra parse-trees). This is because we are concerned with the *disambiguation capability* of each of the algorithms over the *same set of parse-trees*, rather than its other capabilities. Moreover, the LR results improve only by a marginal amount by allowing new combinations of constituents (Goodman p.c.).

We use the Penn Wall Street Journal (WSJ) tree-bank [Marcus et al., 1993] sections 02-21 for training and the first 1000 trees of section 23 for testing. It is noteworthy that our experiments with the MM method are the first on a large tree-bank (Goodman's concerned the small ATIS). We conducted experiments with two different kinds of linguistic structure by transforming the training set in two alternative ways:

1) $\mathcal{M}_{RL}$ **parser** like Goodman [Goodman, 1998] we right linearize (hence $\mathcal{M}_{RL}$ parser) all n-ary rules (when $n > 2$) in the training tree-bank using the method shown in figure 4,

2) $\mathcal{M}_H$ **parser** we remove the unary productions and (1) extend the label of each node, except for the pre-terminal level, with the label of the mother node and also with the label of the pre-terminal of the head-word[5] (hence $\mathcal{M}_H$ parser), (2) linearize the n-ary rules around the head-child as shown in figure 4, and (3) encode the left/right subcat frames (Collins style) in the labels of the left/right linearization nodes.

[5]This is a simplification of the bilexical-dependency approach e.g. [Collins, 1997] in that we do not include the actual head-words on the labels of the nodes.

Metrics	MPP	MLR	$MPCh$	$MCFP$	MPP	MLR	$MPCh$	$MCFP$
Lab.Rec.	60.06	60.36	61.10	61.69	80.07	81.97	80.70	81.58
Lab.Prec.	62.90	65.55	65.25	65.22	80.04	81.73	80.65	81.47
F_β	61.44	62.84	63.10	**63.40**	80.05	**81.85**	80.67	81.52
Ex.Match	**5.4**	3.9	4.5	4.8	23.0	**23.4**	22.6	**23.4**
0-Cross	**17.8**	15.6	16.5	16.5	47.0	**48.2**	48.0	48.0

Table 2: **(Left)** Results on right linearized tree-bank, **(Right)** Results on semi-head lex. tree-bank.

These transforms result in two training tree-banks from which we obtained two different PCFGs by simple counting, i.e. relative frequency model without smoothing. Both resulting parsers parse word-sequences and so we use the method described in [Collins, 1997] for dealing with unknown words.

We employ the same parser implementation for the various algorithms. We evaluate the parser's output against[6] the gold standard set after the respective transformations are applied, i.e. for the $\mathcal{M}_{RL}$ parser we compare against binary branching trees, and for $\mathcal{M}_H$ parser against the trees obtained after removing the unary productions from the original WSJ trees. We use the program *evalb* for PARSEVAL evaluation[7].

Table 2 (left) shows the results of the $\mathcal{M}_{RL}$ parser. The MLR , $MPCh$ and $MCFP$ algorithms improve over the MPP in all metrics, especially the $MCFP$ algorithm improves on the MPP by up to 2% in terms of $F_\beta = \frac{2*LRP*LRR}{LRP+LRR}$. This supports the intuition underlying the Maximizing Metrics hypothesis. However, the results show that the new $MCFP$ and $MPCh$ improve over the MLR algorithm in terms of F_β (by 0.3-0.6%)! This experiment confirms our hypothesis, that optimizing a metric that takes more linguistic features of trees into account is more beneficial than maximizing the evaluation metric itself. Nevertheless, there seems to be a kind of balance between the amount of linguistic features accounted for by the algorithm and the sparsity of the data (this explains why the MPP is suboptimal). In this case there seems to be enough data to support a frames-based metric ($MCFP$) but not enough to support exact-match (MPP).

In light of these results, the experiment using the $\mathcal{M}_H$ parser becomes interesting. By design, the $\mathcal{M}_H$ parser employs complex non-terminals, each consisting of a concatenation of a tree-bank node-label, the parent and pre-head labels of that node. Hence, a $\mathcal{M}_H$ parser non-terminal label already captures more information than the parent-child relations of the $\mathcal{M}_{RL}$ parser, which the $MPCh$ and $MCFP$ algorithms aimed at extracting. As table 2 (right) shows, the MLR algorithm improves over all other algorithms, with the MPP being the least successfull on F_β. This result shows that the complex non-terminals of the $\mathcal{M}_H$ parser, that already encode parent-child relations, enable the MLR to perform well.

In terms of exact-match, we see that the MPP scores as best in the first experiment, but as worst in the second. In the first experiment there is a natural flow of improvement on exact-match from MLR , through $MPCh$ and $MCFP$ up to the MPP . In the second experiment, in contrast, we see that

[6]Our evaluation against the original tree-bank shows that the relative differences in performance remain very much the same.
[7]http://www.research.att.com/ mcollins/

the MLR and $MCFP$ score equivalently, and better than the MPP . This result clearly exposes the Achilles' heal of the MPP : sparse-data. Sparse-data is at its worst in the case of exact-match because the exactly matching parse-tree is often not in the tree-language of the grammar. Hence, another parse gets selected: the most probable available parse.

These experiments warrant two conclusions: (1) in practical situations it is worth considering different algorithms as the MLR might be suboptimal even in terms of LR, and (2) algorithms that take more linguistic aspects into account could be more interesting to consider for various applications than the MLR as these algorithms tend to score better than the MLR in terms of more relevant evaluation metrics (e.g. those that make semantic interpretation of the output parses much easier).

7 Discussion

When the probabilistic model is a weak approximation of the data, it is wise to employ algorithms that optimize other, more linguistically relevant evaluation metrics. We conjecture that our result also applies to [Stolcke et al., 1997], who suggest that it is better to optimize the word-error rate of a speech-recognition model, instead of selecting the most probable sequence of words under that model (i.e. Maximum A Posteriori - MAP - solution).

We have shown that the empirically optimal algorithm is one that strikes a balance between data-sparseness and the amount of linguistic knowledge taken into account by that algorithm. We exemplified this through various algorithms that optimize new, tree evaluation metrics that match trees with respect to an increasing amount of linguistic evidence in these trees.

We believe that there are computational as well as empirical benefits to be gained by employing the algorithms discussed in the present paper. For some kinds of models, e.g. the Data Oriented Parsing (DOP) model and Hidden Markov Models (HMMs), the MAP approach can be NP-Complete, and efficient approximations via sampling do not alleviate this. In a larger number of cases, sparse-data often imply that the MAP solution is not among those that are available to the model, and only a less optimal solution exists. By optimizing the suitable metric, which might or might not coincide with the final evaluation metric, it is possible to obtain more optimal disambiguators.

References

[Black et al., 1991] Black et al., E. (1991). A procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars. In *Proceedings of the February 1991 DARPA Speech and Natural Language Workshop*, pages 306–311, San Mateo, CA. Morgan Kaufman.

[Bod, 1998] Bod, R. (1998). *Beyond Grammar: An Experience-Based Theory of Language*. CSLI Publications, California.

[Carroll et al., 2002] Carroll, J., Frank, A., Lin, D., Prescher, D., and Uszkoreit, H., editors (2002). *Beyond PARSEVAL: Towards Improved Evaluation Measures for Parsing Systems, Proceedings of Workshop of the Third LREC Conference 2002*. LREC, Las Palmas, Canary Islands, Spain.

[Charniak, 1996] Charniak, E. (1996). Tree-bank Grammars. In *Proceedings AAAI'96*, Portland, Oregon.

[Collins, 1997] Collins, M. (1997). Three generative, lexicalized models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL) and the 8th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 16–23, Madrid, Spain.

[Duda et al., 2001] Duda, R., Hart, P., and Stork, D. (2001). *Pattern Classification*. John Wiley & Sons, NY, USA.

[Goodman, 1998] Goodman, J. (1998). *Parsing Inside-Out*. PhD thesis, Department of Computer Science, Harvard University, Cambridge, Massachusetts.

[Goodman, 1996] Goodman, J. T. (1996). Parsing algorithms and metrics. In Joshi, A. and Palmer, M., editors, *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL'96)*, pages 177–183, San Francisco. Morgan Kaufmann Publishers.

[Lari and Young, 1991] Lari, K. and Young, S. (1991). Applications of stochastic context-free grammars using the inside-outside algorithm. *Computer, Speech and Language*, 5:237–257.

[Manning and Schutze, 1999] Manning, C. and Schutze, H. (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA.

[Marcus et al., 1993] Marcus, M., Santorini, B., and Marcinkiewicz, M. (1993). Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19:313–330.

[Sima'an, 2002] Sima'an, K. (2002). Computational complexity of probabilistic disambiguation. *Grammars*, 5(2):125–151.

[Stolcke et al., 1997] Stolcke, A., König, Y., and Weintraub, M. (1997). Explicit word error minimization in N-best list rescoring. In *Proc. Eurospeech '97*, pages 163–166, Rhodes, Greece.