Upper Bounds for Unsupervised Parsing with Unambiguous Non-Terminally Separated Grammars

Franco M. Luque and **Gabriel Infante-Lopez**
Grupo de Procesamiento de Lenguaje Natural
Universidad Nacional de Córdoba & CONICET
Argentina
{francolq|gabriel}@famaf.unc.edu.ar

Abstract

Unambiguous Non-Terminally Separated (UNTS) grammars have properties that make them attractive for grammatical inference. However, these properties do not state the maximal performance they can achieve when they are evaluated against a gold treebank that is not produced by an UNTS grammar. In this paper we investigate such an upper bound. We develop a method to find an upper bound for the unlabeled $F1$ performance that any UNTS grammar can achieve over a given treebank. Our strategy is to characterize all possible versions of the gold treebank that UNTS grammars can produce and to find the one that optimizes a metric we define. We show a way to translate this score into an upper bound for the $F1$. In particular, we show that the $F1$ parsing score of any UNTS grammar can not be beyond 82.2% when the gold treebank is the WSJ10 corpus.

1 Introduction

Unsupervised learning of natural language has received a lot of attention in the last years, e.g., Klein and Manning (2004), Bod (2006a) and Seginer (2007). Most of them use sentences from a treebank for training and trees from the same treebank for evaluation. As such, the best model for unsupervised parsing is the one that reports the best performance.

Unambiguous Non-Terminally Separated (UNTS) grammars have properties that make them attractive for grammatical inference. These grammars have been shown to be PAC-learnable in polynomial time (Clark, 2006), meaning that under certain circumstances, the underlying grammar can be learned from a sample of the underlying language. Moreover, UNTS grammars have been successfully used to induce grammars from unannotated corpora in competitions of learnability of formal languages (Clark, 2007).

UNTS grammars can be used for modeling natural language. They can be induced using any training material, the induced models can be evaluated using trees from a treebank, and their performance can be compared against state-of-the-art unsupervised models. Different learning algorithms might produce different grammars and, consequently, different scores. The fact that the class of UNTS grammars is PAC learnable does not convey any information on the possible scores that different UNTS grammars might produce. From a performance oriented perspective it might be possible to have an upper bound over the set of possible scores of UNTS grammars. Knowing an upper bound is complementary to knowing that the class of UNTS grammars is PAC learnable.

Such upper bound has to be defined specifically for UNTS grammars and has to take into account the treebank used as test set. The key question is how to compute it. Suppose that we want to evaluate the performance of a given UNTS grammar using a treebank. The candidate grammar produces a tree for each sentence and those trees are compared to the original treebank. We can think that the candidate grammar has produced a new version of the treebank, and that the score of the grammar is a measure of the closeness of the new treebank to the original treebank. Finding the best upper bound is equivalent to finding the closest UNTS version of the treebank to the original one.

Such bounds are difficult to find for most classes of languages because the search space is the set of all possible versions of the treebank that might have been produced by any grammar in the class under study. In order to make the problem tractable, we need the formalism to have an easy way to characterize all the versions of a treebank

Proceedings of the EACL 2009 Workshop on Computational Linguistic Aspects of Grammatical Inference, pages 58–65,
Athens, Greece, 30 March 2009.

it might produce. UNTS grammars have a special characterization that makes the search space easy to define but whose exploration is NP-hard.

In this paper we present a way to characterize UNTS grammars and a metric function to measure the closeness between two different version of a treebank. We show that the problem of finding the closest UNTS version of the treebank can be described as Maximum Weight Independent Set (MWIS) problem, a well known NP-hard problem (Karp, 1972). The exploration algorithm returns a version of the treebank that is the closest to the gold standard in terms of our own metric.

We show that the $F1$-measure is related to our measure and that it is possible to find and upper bound of the $F1$-performance for all UNTS grammars. Moreover, we compute this upper bound for the WSJ10, a subset of the Penn Treebank (Marcus et al., 1994) using POS tags as the alphabet. The upper bound we found is 82.2% for the $F1$ measure. Our result suggest that UNTS grammars are a formalism that has the potential to achieve state-of-the-art unsupervised parsing performance but does not guarantee that there exists a grammar that can actually achieve the 82.2%.

To the best of our knowledge, there is no previous research on finding upper bounds for performance over a concrete class of grammars. In Klein and Manning (2004), the authors compute an upper bound for parsing with binary trees a gold treebank that is not binary. This upper bound, that is 88.1% for the WSJ10, is for any parser that returns binary trees, including the concrete models developed in the same work. But their upper bound does not use any specific information of the concrete models that may help them to find better ones.

The rest of the paper is organized as follows. Section 2 presents our characterization of UNTS grammars. Section 3 introduces the metric we optimized and explains how the closest version of the treebank is found. Section 4 explains how the upper bound for our metric is translated to an upper bound of the $F1$ score. Section 5 presents our bound for UNTS grammars using the WSJ10 and finally Section 6 concludes the paper.

2 UNTS Grammars and Languages

Formally, a context free grammar $G = (\Sigma, N, S, P)$ is said to be Non-Terminally Separated (NTS) if, for all $X, Y \in N$ and $\alpha, \beta, \gamma \in (\Sigma \cup N)^*$ such that $X \stackrel{*}{\Rightarrow} \alpha\beta\gamma$ and $Y \stackrel{*}{\Rightarrow} \beta$, we have that $X \stackrel{*}{\Rightarrow} \alpha Y \gamma$ (Clark, 2007). Unambiguous NTS (UNTS) grammars are those NTS grammars that parses unambiguously every instance of the language.

Given any grammar G, a substring s of $r \in L(G)$ is called a *constituent* of r if and only if there is an X in N such that $S \stackrel{*}{\Rightarrow} uXv \stackrel{*}{\Rightarrow} usv = r$. In contrast, a string s is called a non-constituent or *distituent* of $r \in L(G)$ if s is not a constituent of r. We say that s is a constituent of a language $L(G)$ if for every r that contains s, s is a constituent of r. In contrast, s is a distituent of $L(G)$ if for every r where s occurs, s is a distituent of r.

An interesting characterization of finite UNTS grammars is that every substring that appear in some string of the language is always a constituent or always a distituent. In other words, if there is a string r in $L(G)$ for which s is a constituent, then s is a constituent of $L(G)$. By means of this property, if we ignore the non-terminal labels, a finite UNTS language is fully determined by its set of constituents C. We can show this property for finite UNTS languages. We believe that it can also be shown for non-finite cases, but for our purposes the finite cases suffices, because we use grammars to parse finite sets of sentences, specifically, the sentences of test treebanks. We know that for every finite subset of an infinite language produced by a UNTS grammar G, there is a UNTS grammar G' whose language is finite and that parses the finite subset as G. If we look for the upper bound among the grammars that produce a finite language, this upper bound is also an upper bound for the class of infinite UNTS grammars.

The UNTS characterization plays a very important role in the way we look for the upper bound. Our method focuses on how to determine which of the constituents that appear in the gold are actually the constituents that produce the upper bound. Suppose that a given gold treebank contains two strings α and β such that they *occur overlapped*. That is, there exist non-empty strings α', γ, β' such that $\alpha = \alpha'\gamma$ and $\beta = \gamma\beta'$ and $\alpha'\gamma\beta'$ occurs in the treebank. If C is the set of constituents of a UNTS grammar it can not have both α and β. It might have one or the other, but if both belong to C the resulting language can not be UNTS. In order to find the closest UNTS grammar we design a procedure that looks for the subset of all substrings that occur in the sentences of the gold treebank that can be the constituent set C

of a grammar. We do not explicitly build a UNTS grammar, but find the set C that produces the best score.

We say that two strings α and β are *compatible* in a language L if they do not occur overlapped in L, and hence they both can be members of C. If we think of L as a subset of an infinite language, it is not possible to check that two overlapping strings do not appear overlapped in the "real" language and hence that they are actually compatible. Nevertheless, we can guarantee compatibility between two strings α, β by requiring that they do not overlap at all, this is, that there are no non-empty strings α', γ, β' such that $\alpha = \alpha'\gamma$ and $\beta = \gamma\beta'$. We call this type of compatibility *strong compatibility*. Strong compatibility ensures that two strings can belong to C regardless of L. In our experiments we focus on finding the best set C of compatible strings.

Any set of compatible strings C extracted from the gold treebank can be used to produce a new version of the treebank. For example, Figure 1 shows two trees from the WSJ Penn Treebank. The string "in the dark" occurs as a constituent in (a) and as a distituent in (b). If C contains "in the dark", it can not contain "the dark clouds" given that they overlap in the yield of (b). As a consequence, the new treebank correctly contains the subtree in (a) but not the one in (b). Instead, the yield of (b) is described as in (c) in the new treebank.

C defines a new version of the treebank that satisfies the UNTS property. Our goal is to obtain a treebank T' such that (a) T' and T are treebanks over the same set of sentences, (b) T' is UNTS, and (c) T' is the closest treebank to T in terms of performance. The three of them imply that any other UNTS grammar is not as similar as the one we found.

3 Finding the Best UNTS Grammar

As our goal is to find the closest grammar in terms of performance, we need to define first a weight for each possible grammar and second, an algorithm that searches for the grammar with the best weight. Ideally, the weight of a candidate grammar should be in terms of $F1$, but we can show that optimization of this particular metric is computationally hard. Instead of defining $F1$ as their score, we introduce a new metric that is easier to optimize, we find the best grammar for this metric, and we show that the possible values of $F1$ can be bounded by a function that takes this score as argument. In this section we present our metric and the technique we use to find a grammar that reports the best value for our metric.

If the original treebank T is not produced by any UNTS grammar, then there are strings in T that are constituents in some sentences and that are distituents in some other sentences. For each one of them we need a procedure to decide whether they are members of C or not. If a string α appears a significant number of times more as a constituent than as a distituent the procedure may choose to include it in C at the price of being wrong a few times. That is, the new version of T has all occurrences of α either as constituents or as distituents. The treebank that has all of its occurrences as constituents differs from the original in that there are some occurrences of α that were originally distituents and are marked as constituents. Similarly, if α is marked as distituent in the new treebank, it has occurrences of α that were constituents in T.

The decision procedure becomes harder when all the substrings that appear in the treebank are considered. The increase in complexity is a consequence of the number of decisions the procedure needs to take and the way these decisions interfere one with another. We show that the problem of determining the set C is naturally embedded in a graph NP-hard problem. We define a way to look for the optimal grammars by translating our problem to a well known graph problem. Let L be the the set of sentences in a treebank, and let $S(L)$ be all the possible non-empty proper substrings of L. We build a weighted undirected graph G in terms of the treebank as follows. Nodes in G correspond to strings in $S(L)$. The weight of a node is a function $w(s)$ that models our interest of having s selected as a constituent; $w(s)$ is defined in terms of some information derived from the gold treebank T and we discuss it later in this section. Finally, two nodes a and b are connected by an edge if their two corresponding strings conflict in a sentence of T (i.e., they are not compatible in L).

Not all elements of L are in $S(L)$. We did not include L in $S(L)$ for two practical reasons. The first one is that to require L in $S(L)$ is too restrictive. It states that all strings in L are in fact constituents. If two string ab and bc of L occur overlapped in a third string abc then there is no UNTS grammar capable of having the three of

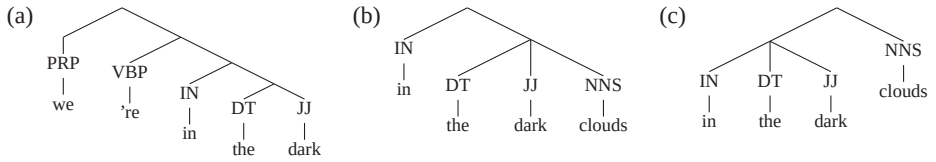


Figure 1: (a) and (b) are two subtrees that show "in the dark" as a constituent and as a distituent respectively. (c) shows the result of choosing "in the dark" as a constituent.

them as constituents. The second one is that including them produces graphs that are too sparse. If they are included in the graph, we know that any solution should contain them, consequently, all their neighbors do not belong to any solution and they can be removed from the graph. Our experiments show that the graph that results from removing nodes related to nodes representing strings in L are too small to produce any interesting result.

By means of representing the treebank as a graph, selecting a set of constituents $C \subseteq S(L)$ is equivalent to selecting an independent set of nodes in the graph. An *independent set* is a subset of the set of nodes that do not have any pair of nodes connected by an edge. Clearly, there are exponentially many possible ways to select an independent set, and each of these sets represents a set of constituents. But, since we are interested in the best set of constituents, we associate to each independent set C the weight $W(C)$ defined as $\sum_{s \in C} w(s)$. Our aim is then to find a set C_{max} that maximizes this weight. This problem is a well known problem of graph theory known in the literature as the Maximum Weight Independent Set (MWIS) problem. This problem is also known to be NP-hard (Karp, 1972).

We still have to choose a definition for $w(s)$. We want to find the grammar that maximizes $F1$. Unfortunately, $F1$ can not be expressed in terms of a sum of weights. Maximization of $F1$ is beyond the expressiveness of our model, but our strategy is to define a measure that correlates with $F1$ and that can be expressed as a sum of weights.

In order to introduce our measure, we first define $c(s)$ and $d(s)$ as the number of times a string s appears in the gold treebank T as a constituent and as a distituent respectively. Observe that if we choose to include s as a constituent of C, the resulting treebank T' contains all the $c(s) + d(s)$ occurrences of s as a constituent. $c(s)$ of the s occurrences in T' are constituents as they are in T and $d(s)$ of the occurrences are constituents in T' but are in fact distituents in T. We want to maximize $c(s)$ and minimize $d(s)$ at the same time. This can be done by defining the contribution of a string s to the overall score as

$$w(s) = c(s) - d(s).$$

With this definition of w, the weight $W(C) = \sum_{s \in C} w(s)$ becomes the number of constituents of T' that are in T minus the number of constituents that do not. If we define the number of *hits* to be $H(C) = \sum_{s \in C} c(s)$ and the number of *misses* to be $M(C) = \sum_{s \in C} d(s)$ we have that

$$W(C) = H(C) - M(C). \quad (1)$$

As we confirm in Section 5, graphs tend to be very big. In order to reduce the size of the graphs, if a string s has $w(s) \leq 0$, we do not include its corresponding node in the graph. An independent set that does not include s has an equal or higher W than the same set including s.

For example, let T be the treebank in Figure 2 (a). The sets of substrings such that $w(c) \geq 0$ is $\{da, cd, bc, cda, ab, bch\}$. The graph that corresponds to this set of strings is given in Figure 3. Nodes corresponding to strings $\{dabch, bcda, abe, abf, abg, bci, daj\}$ are not shown in the figure because the strings do not belong to $S(L)$. The figure also shows the weights associated to the substrings according to their counts in Figure 2 (a). The shadowed nodes correspond to the independent set that maximizes W. The trees in the Figure 2 (b) are the sentences of the treebank parsed according the optimal independent set.

4 An Upper Bound for F1

Even though finding the independent set that maximizes W is an NP-Hard problem, there are instances where it can be effectively computed, as we show in the next section. The set C_{max} maximizes W for the WSJ10 and we know that all others C produces a lower value of W. In other words, the set C_{max} produce a treebank T_{max} that

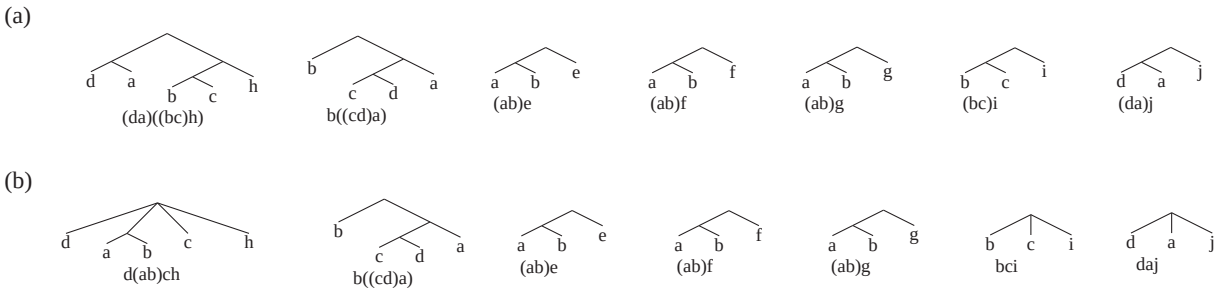


Figure 2: (a) A gold treebank. (b) The treebank generated by the grammar $C = L \cup \{cd, ab, cda\}$.

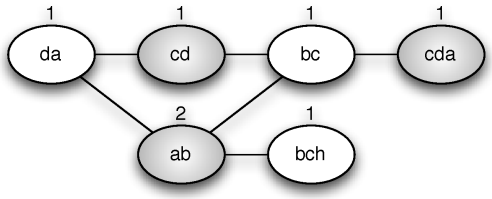


Figure 3: Graph for the treebank of Figure 2.

is the closest UNTS version to the WSJ10 in terms of W. We can compute the precision, recall and $F1$ for C_{max} but there is no warranty that the $F1$ score is the best for all the UNTS grammars. This is the case because $F1$ and W do not define the same ordering over the family of candidate constituent sets C: there are gold treebanks T (used for computing the metrics), and sets C_1, C_2 such that $F1(C_1) < F1(C_2)$ and $W(C_1) > W(C_2)$. For example, consider the gold treebank T in Figure 4 (a). The table in Figure 4 (b) displays two sets C_1 and C_2, the treebanks they produce, and their values of $F1$ and W. Note that C_2 is the result of adding the string ef to C_1, also note that $c(ef) = 1$ and $d(ef) = 2$. This improves the $F1$ score but produces a lower W.

The $F1$ measure we work with is the one defined in the recent literature of unsupervised parsing (Klein and Manning, 2004). $F1$ is defined in terms of Precision and Recall as usual, and the last two measures are micro-averaged measures that include full-span brackets, and that ignore both unary branches and brackets of span one. For simplicity, the previous example does not count the full-span brackets.

As the example shows, the upper bound for W might not be an upper bound of $F1$, but it is possible to find a way to define an upper bound of $F1$ using the upper bound of W. In this section we define a function f with the following property. Let X and Y be the sets of W-weights and $F1$-weights for all possible UNTS grammars respectively. Then, if w is an upper bound of X, then $f(w)$ is an upper bound of Y. The function f is defined as follows:

$$f(w) = F1\left(\frac{1}{2 - \frac{w}{K}}, 1\right) \quad (2)$$

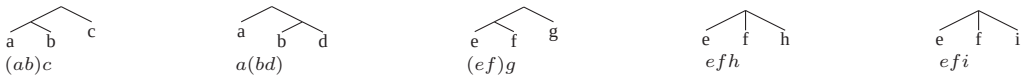
where $F1(p, r) = \frac{2pr}{p+r}$, and $K = \sum_{s \in S_T} c(s)$ is the total number of constituents in the gold treebank T. From it, we can also derive values for precision and recall: precision $\frac{1}{2-\frac{w}{K}}$ and recall 1. A recall of 1 is clearly an upper bound for all the possible values of recall, but the value given for precision is not necessarily an upper bound for all the possible values of precision. It might exist a grammar having a higher value of precision but whose $F1$ has to be below our upper bound.

The rest of section shows that $f(W)$ is an upper bound for $F1$, the reader not interested in the technicalities can skip it.

The key insight for the proof is that both metrics $F1$ and W can be written in terms of precision and recall. Let T be the treebank that is used to compute all the metrics. And let T' be the treebank produced by a given constituent set C. If a string s belongs to C, then its $c(s) + d(s)$ occurrences in T' are marked as constituents. Moreover, s is correctly tagged a $c(s)$ number of times while it is incorrectly tagged a $d(s)$ number of times. Using this, P, R and $F1$ can be computed for C as follows:

$$\begin{aligned} P(C) &= \frac{\sum_{s \in C} c(s)}{\sum_{s \in C} c(s)+d(s)} \\ &= \frac{H(C)}{H(C)+M(C)} \quad (3) \\ R(C) &= \frac{\sum_{s \in C} c(s)}{K} \\ &= \frac{H(C)}{K} \quad (4) \\ F1(C) &= \frac{2P(C)R(C)}{P(C)+R(C)} \\ &= \frac{2H(C)}{K+H(C)+M(C)} \end{aligned}$$

(a)



(b)

C	T'_C	P	R	$F1$	W
$C_1 = \{abc, abd, efg, efh, efi, ab\}$	$\{(ab)c, (ab)d, efg, efh, efi\}$	50%	33%	40%	$1 - 1 = 0$
$C_2 = \{abc, abd, efg, efh, efi, ab, ef\}$	$\{(ab)c, (ab)d, (ef)g, (ef)h, (ef)i\}$	40%	67%	50%	$2 - 3 = -1$

Figure 4: (a) A gold treebank. (b) Two grammars, the treebanks they generate, and their scores.

W can also be written in terms of P and R as

$$W(C) = (2 - \frac{1}{P(C)})R(C)K \qquad (5)$$

This formula is proved to be equivalent to Equation (1) by replacing $P(C)$ and $R(C)$ with equations (3) and (4) respectively. Using the last two equations, we can rewrite $F1$ and W taking p and r, representing values of precision and recall, as parameters:

$$F1(p, r) = \frac{2pr}{p + r}$$
$$W(p, r) = (2 - \frac{1}{p})rK \qquad (6)$$

Using these equations, we can prove that f correctly translates upper bounds of W to upper bounds of $F1$ using calculus. In contrast to $F1$, W not necessarily take values between 0 and 1. Instead, it takes values between K and $-\infty$. Moreover, it is negative when $p < \frac{1}{2}$, and goes to $-\infty$ when p goes to 0. Let C be an arbitrary UNTS grammar, and let p_C, r_C and w_C be its precision, recall and W-weight respectively. Let w be our upper bound, so that $w_C \leq w$. If $f1_C$ is defined as $F1(p_C, r_C)$ we need to show that $f1_C \leq f(w)$. We bound $f1_C$ in two steps. First, we show that

$$f1_C \leq f(w_C)$$

and second, we show that

$$f(w_C) \leq f(w).$$

The first inequality is proved by observing that $f1_C$ and $f(w_C)$ are the values of the function

$$f1(r) = F1\left(\frac{1}{2 - \frac{w_C}{Kr}}, r\right)$$

at the points $r = r_C$ and $r = 1$ respectively. This function corresponds to the line defined by the $F1$ values of all possible models that have a fixed weight $W = w_C$. The function is monotonically increasing in r, so we can apply it to both sides of the following inequality $r_C \leq 1$, which is trivially true. As result, we get $f1_C \leq f(w_C)$ as required. The second inequality is proved by observing that $f(w)$ is monotonically increasing in w, and by applying it to both sides of the hypothesis $w_c \leq w$.

5 UNTS Bounds for the WSJ10 Treebank

In this section we focus on trying to find real upper bounds building the graph for a particular treebank T. We find the best independent set, we build the UNTS version T_{max} of T and we compute the upper bound for $F1$. The treebank we use for experiments is the WSJ10, which consists of the sentences of the WSJ Penn Treebank whose length is at most 10 words after removing punctuation marks (Klein and Manning, 2004). We also removed lexical entries transforming POS tags into our terminal symbols as it is usually done (Klein and Manning, 2004; Bod, 2006a).

We start by finding the best independent set. To solve the problem in the practice, we convert it into an Integer Linear Programming (ILP) problem. ILP is also NP-hard (Karp, 1972), but there is software that implements efficient strategies for solving some of its instances (Achterberg, 2004).

ILP problems are defined by three parameters. First, there is a set of variables that can take values from a finite set. Second, there is an objective function that has to be maximized, and third, there is a set of constraints that must be satisfied. In our case, we define a binary variable $x_s \in \{0, 1\}$ for every node s in the graph. Its value is 1 or 0, that respectively determines the presence or absence of s in the set C_{max}. The objective function is

$$\sum_{s \in S(L)} x_s w(s)$$

The constraints are defined using the edges of the

graph. For every edge (s_1, s_2) in the graph, we add the following constraint to the problem:

$$x_{s_1} + x_{s_2} \leq 1$$

The 7422 trees of the WSJ10 treebank have a total of 181476 substrings of length ≥ 2, that form the set $S(L)$ of 68803 different substrings. The number of substrings in $S(L)$ does not grow too much with respect to the number of strings in L because substrings are sequences of POS tags, meaning that each substring is very frequent in the corpus. If substrings were made out of words instead of POS tags, the number of substrings would grow much faster, making the problem harder to solve. Moreover, removing the strings s such that $w(s) \leq 0$ gives a total of only 7029 substrings. Since there is a node for each substring, the resulting graph contains 7029 nodes. Recall that there is an edge between two strings if they occur overlapped. Our graph contains 1204 edges. The ILP version has 7029 variables, 1204 constraints and the objective function sums over 7029 variables. These numbers are summarized in Table 1.

The solution of the ILP problem is a set of 6583 variables that are set to one. This set corresponds to a set C_{max} of nodes in our graph of the same number of elements. Using C_{max} we build a new version T_{max} of the WSJ10, and compute its weight W, precision, recall and $F1$. Their values are displayed in Table 2. Since the elements of L were not introduced in $S(L)$, elements of L are not necessarily in C_{max}, but in order to compute precision and recall, we add them by hand. Strictly speaking, the set of constituents that we use for building T_{max} is C_{max} plus the full span brackets.

We can, using equation (2), compute the upper bound of $F1$ for all the possible scores of all UNTS grammars that use POS tags as alphabet:

$$f(w_{max}) = F1\left(\frac{1}{2 - \frac{w_{max}}{K}}, 1\right) = 82.2\%$$

The precision for this upper bound is

$$P(w_{max}) = \frac{1}{2 - \frac{w_{max}}{K}} = 69.8\%$$

while its recall is $R = 100\%$. Note from the previous section that $P(w_{max})$ is not an upper bound for precision but just the precision associated to the upper bound $f(w_{max})$.

Gold constituents	K	35302
Strings	$\|S(L)\|$	68803
Nodes		7029
Edges		1204

Table 1: Figures for the WSJ10 and its graph.

Hits	H	22169
Misses	M	2127
Weight	W	20042
Precision	P	91.2%
Recall	R	62.8%
F1	$F1$	74.4%

Table 2: Summary of the scores for C_{max}.

Table 3 shows results that allow us to compare the upper bounds with state-of-the-art parsing scores. BestW corresponds to the scores of T_{max} and UBoundF1 is the result of our translation function f. From the table we can see that an unsupervised parser based on UNTS grammars may reach a sate-of-the-art performance over the WSJ10. RBranch is a WSJ10 version where all trees are binary and right branching. DMV, CCM and DMV+CCM are the results reported in Klein and Manning (2004). U-DOP and UML-DOP are the results reported in Bod (2006b) and Bod (2006a) respectively. Incremental refers to the results reported in Seginer (2007).

We believe that our upper bound is a generous one and that it might be difficult to achieve it for two reasons. First, since the WSJ10 corpus is a rather flat treebank, from the 68803 substrings only 10% of them are such that $c(s) > d(s)$. Our procedure has to decide among this 10% which of the strings are constituents. An unsupervised method has to choose the set of constituents from the set of all 68803 possible substrings. Second, we are supposing a recall of 100% which is clearly too optimistic. We believe that we can find a tighter upper bound by finding an upper bound for recall, and by rewriting f in equation (2) in terms of the upper bound for recall.

It must be clear the scope of the upper bound we found. First, note that it has been computed over the WSJ10 treebank using the POS tags as the alphabet. Any other alphabet we use, like for example words, or pairs of words and POS tags, changes the relation of compatibility among the substrings, making a completely different universe

Model	UP	UR	F1
RBranch	55.1	70.0	61.7
DMV	46.6	59.2	52.1
CCM	64.2	81.6	71.9
DMV+CCM	69.3	88.0	77.6
U-DOP	70.8	88.2	78.5
UML-DOP			82.9
Incremental	75.6	76.2	75.9
BestW(UNTS)	91.2	62.8	**74.4**
UBoundF1(UNTS)	69.8	100.0	**82.2**

Table 3: Performance on the WSJ10 of the most recent unsupervised parsers, and our upper bounds on UNTS.

of UNTS grammars. Second, our computation of the upper bound was not made for supersets of the WSJ10. Supersets such as the entire Penn Treebank produce bigger graphs because they contain longer sentences and various different sequences of substrings. As the maximization of W is an NP-hard problem, the computational cost of solving bigger instances grows exponentially. A third limitation that must be clear is about the models affected by the bound. The upper bound, and in general the method, is only applicable to the class of formal UNTS grammars, with only some very slight variants mentioned in the previous sections. Just moving to probabilistic or weighted UNTS grammars invalidates all the presented results.

6 Conclusions

We present a method for assessing the potential of UNTS grammars as a formalism for unsupervised parsing of natural language. We assess their potential by finding an upper bound of their performance when they are evaluated using the WSJ10 treebank. We show that any UNTS grammars can achieve at most 82.2% of $F1$ measure, a value comparable to most state-of-the-art models. In order to compute this upper bound we introduced a measure that does not define the same ordering among UNTS grammars as the $F1$, but that has the advantage of being computationally easier to optimize. Our measure can be used, by means of a translation function, to find an upper bound for $F1$. We also showed that the optimization procedure for our metric maps into an NP-Hard problem, but despite this fact we present experimental results that compute the upper bound for the WSJ10 when POS tags are treated as the grammar alphabet.

From a more abstract perspective, we introduced a different approach to assess the usefulness of a grammatical formalism. Usually, formalism are proved to have interesting learnability properties such as PAC-learnability or convergence of a probabilistic distribution. We present an approach that even though it does not provide an effective way of computing the best grammar in an unsupervised fashion, it states the upper bound of performance for all the class of UNTS grammars.

Acknowledgments

This work was supported in part by grant PICT 2006-00969, ANPCyT, Argentina. We would like to thank Pablo Rey (UDP, Chile) for his help with ILP, and Demetrio Martín Vilela (UNC, Argentina) for his detailed review.

References

Tobias Achterberg. 2004. SCIP - a framework to integrate Constraint and Mixed Integer Programming. Technical report.

Rens Bod. 2006a. An all-subtrees approach to unsupervised parsing. In *Proceedings of COLING-ACL 2006*.

Rens Bod. 2006b. Unsupervised parsing with U-DOP. In *Proceedings of CoNLL-X*.

Alexander Clark. 2006. PAC-learning unambiguous NTS languages. In *Proceedings of ICGI-2006*.

Alexander Clark. 2007. Learning deterministic context free grammars: The Omphalos competition. *Machine Learning*, 66(1):93–110.

Richard M. Karp. 1972. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of ACL 42*.

Mitchell P. Marcus, Beatrice Santorini, and Mary A. Marcinkiewicz. 1994. Building a large annotated corpus of english: The Penn treebank. *Computational Linguistics*, 19(2):313–330.

Yoav Seginer. 2007. Fast unsupervised incremental parsing. In *Proceedings of ACL 45*.