PCFG Induction for Unsupervised Parsing and Language Modelling

James Scicluna
Université de Nantes,
CNRS, LINA, UMR6241,
F-44000, France
james.scicluna@univ-nantes.fr

Colin de la Higuera
Université de Nantes,
CNRS, LINA, UMR6241,
F-44000, France
cdlh@univ-nantes.fr

Abstract

The task of unsupervised induction of probabilistic context-free grammars (PCFGs) has attracted a lot of attention in the field of computational linguistics. Although it is a difficult task, work in this area is still very much in demand since it can contribute to the advancement of language parsing and modelling. In this work, we describe a new algorithm for PCFG induction based on a principled approach and capable of inducing accurate yet compact artificial natural language grammars and typical context-free grammars. Moreover, this algorithm can work on large grammars and datasets and infers correctly even from small samples. Our analysis shows that the type of grammars induced by our algorithm are, in theory, capable of modelling natural language. One of our experiments shows that our algorithm can potentially outperform the state-of-the-art in unsupervised parsing on the WSJ10 corpus.

1 Introduction

The task of unsupervised induction of PCFGs has attracted a lot of attention in the field of computational linguistics. This task can take the form of either parameter search or structure learning. In parameter search, a CFG is fixed and the focus is on assigning probabilities to this grammar using Bayesian methods (Johnson et al., 2007) or maximum likelihood estimation (Lari and Young, 1990). In structure learning, the focus is on building the right grammar rules from scratch. We take the latter approach.

Unsupervised structure learning of (P)CFGs is a notoriously difficult task (de la Higuera, 2010; Clark and Lappin, 2010), with theoretical results showing that in general it is either impossible to achieve (Gold, 1967; de la Higuera, 1997) or requires impractical resources (Horning, 1969; Yang, 2012). At the same time, it is well known that context-free structures are needed for better language parsing and modelling, since less expressive models (such as HMMs) are not good enough (Manning and Schütze, 2001; Jurafsky and Martin, 2008). Moreover, the trend is towards unsupervised (rather than supervised) learning methods due to the lack in most languages of annotated data and the applicability in wider domains (Merlo et al., 2010). Thus, despite its difficulty, unsupervised PCFG grammar induction (or induction of other similarly expressive models) is still an important task in computational linguistics.

In this paper, we describe a new algorithm for PCFG induction based on a principled approach and capable of inducing accurate yet compact grammars. Moreover, this algorithm can work on large grammars and datasets and infers correctly even from small samples. We show that our algorithm is capable of achieving competitive results in both unsupervised parsing and language modelling of typical context-free languages and artificial natural language grammars. We also show that the type of grammars we propose to learn are, in theory, capable of modelling natural language.

2 Preliminaries

2.1 Grammars and Languages

A *context-free grammar* (CFG) is a 4-tuple $\langle N, \Sigma, P, I \rangle$, where N is the set of non-terminals, Σ the set of terminals, P the set of production rules and I a set of starting non-terminals (i.e. multiple starting non-terminals are possible). The language generated from a particular non-terminal A is $L(A) = \{w | A \stackrel{*}{\Rightarrow} w\}$ and the language generated by a grammar G is $L(G) = \bigcup_{S \in I} L(S)$. A CFG is in *Chomsky Normal Form (CNF)* if ev-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1353–1362,
October 25-29, 2014, Doha, Qatar.

ery production rule is of the form $N \rightarrow NN$ or $N \rightarrow \Sigma$.

A *probabilistic context-free grammar* (PCFG) is a CFG with a probability value assigned to every rule and every starting non-terminal. The probability of a leftmost derivation from a PCFG is the product of the starting non-terminal probability and the production probabilities used in the derivation. The probability of a string generated by a PCFG is the sum of all its leftmost derivations' probabilities. The stochastic language generated from a PCFG G is $(L(G), \phi_G)$, where ϕ_G is the distribution over Σ^* defined by the probabilities assigned to the strings by G. For a PCFG to be *consistent*, the probabilities of the strings in its stochastic language must add up to 1 (Wetherell, 1980). Any PCFG mentioned from now onwards is assumed to be consistent.

2.2 Congruence Relations

A *congruence relation* $\sim$ on Σ^* is any equivalence relation on Σ^* that respects the following condition: if $u \sim v$ and $x \sim y$ then $ux \sim vy$. The congruence classes of a congruence relation are simply its equivalence classes. The congruence class of $w \in \Sigma^*$ w.r.t. a congruence relation $\sim$ is denoted by $[w]_\sim$. The set of *contexts* of a substring w with respect to a language L, denoted $Con(w, L)$, is $\{(l, r) \in \Sigma^* \times \Sigma^* \mid lwr \in L\}$. Two strings u and v are *syntactically congruent* with respect to L, written $u \equiv_L v$, if $Con(u, L) = Con(v, L)$. This is a congruence relation on Σ^*. The *context distribution* of a substring w w.r.t. a stochastic language (L, ϕ), denoted $C_w^{(L,\phi)}$, is a distribution whose support is all the possible contexts over alphabet Σ (i.e. $\Sigma^* \times \Sigma^*$) and is defined as follows:

$$C_w^{(L,\phi)}(l, r) = \frac{\phi(lwr)}{\sum_{l',r' \in \Sigma^*} \phi(l'wr')}$$

Two strings u and v are *stochastically congruent* with respect to (L, ϕ), written $u \cong_{(L,\phi)} v$, if $C_u^{(L,\phi)}$ is equal to $C_v^{(L,\phi)}$. This is a congruence relation on Σ^*.

2.3 Congruential Grammars

Clark (2010a) defines *Congruential CFGs* (C-CFGs) as being all the CFGs G which, for any non-terminal A, if $u \in L(A)$ then $L(A) \subseteq [u]_{\equiv_{L(G)}}$ (where $[u]_{\equiv_{L(G)}}$ is the syntactic congruence class of u w.r.t. the language of G). This class of grammars was defined with learnability in mind. Since these grammars have a direct relationship between congruence classes and the non-terminals, their learnability is reduced to that of finding the correct congruence classes (Clark, 2010a).

This class of grammars is closely related to the class of NTS-grammars (Boasson and Sénizergues, 1985). Any C-CFG is an NTS-grammar but not vice-versa. However, it is not known whether languages generated by C-CFGs are all NTS-languages (Clark, 2010a). Note that NTS-languages are a subclass of deterministic context-free languages and contain the regular languages, the substitutable (Clark and Eyraud, 2007) and k-l-substitutable context-free languages (Yoshinaka, 2008), the very simple languages and other CFLs such as the Dyck language (Boasson and Sénizergues, 1985).

We define a slightly more restrictive class of grammars, which we shall call *Strongly Congruential CFGs* (SC-CFGs). A CFG G is a SC-CFG if, for any non-terminal A, if $u \in L(A)$ then $L(A) = [u]_{\equiv_{L(G)}}$. The probabilistic equivalent of this is the class of *Strongly Congruential PCFGs* (SC-PCFGs), defined as all the PCFGs G which, for any non-terminal A, if $u \in L(A)$ then $L(A) = [u]_{\cong_{(L(G),\phi)}}$. In other words, the non-terminals (i.e. syntactic categories in natural language) of these grammars directly correspond to classes of substitutable strings (i.e. substitutable words and phrases in NL). One may ask whether this is too strict a restriction for natural language grammars. We argue that it is not, for the following reasons.

First of all, this restriction complies with the approach taken by American structural linguists for the identification of syntactic categories, as shown by Rauh (2010): *"[Zellig and Fries] identified syntactic categories as distribution classes, employing substitution tests and excluding semantic properties of the items analysed. Both describe syntactic categories exclusively on the basis of their syntactic environments and independently of any inherent properties of the members of these categories"*.

Secondly, we know that such grammars are capable of describing languages generated by grammars that contain typical natural language grammatical structures (see Section 4.1; artificial natural language grammars NL1-NL7, taken from various sources, generate languages which can be described by SC-PCFGs).

3 Algorithm

COMINO (our algorithm) induces SC-PCFGs from a positive sample S. The steps involved are:

1. Inducing the stochastically congruent classes of all the substrings of S
2. Selecting which of the induced classes are non-terminals and subsequently building a CFG.
3. Assigning probabilities to the induced CFG.

The approach we take is very different from traditional grammar induction approaches, in which grouping of substitutable substrings is done incrementally as the same groups are chosen to represent non-terminals. We separate these two task so that learning takes place in the grouping phase whilst selection of non-terminals is done independently by solving a combinatorial problem.

For the last step, the standard EM-algorithm for PCFGs (Lari and Young, 1990) is used. In Sections 3.1 and 3.2, the first and second steps of the algorithm are described in detail. We analyse our algorithm in Section 3.3.

3.1 Inducing the Congruence Classes

We describe in Algorithm 1 how the congruence classes are induced.

Algorithm 1: Learn Congruence Classes

Input: A multiset S; parameters: n, d, i; distance function `dist` on local contexts of size k

Output: The congruence classes $\mathcal{CC}$ over the substrings of S

1 $Subs \leftarrow$ Set of all substrings of S ;
2 $\mathcal{CC} \leftarrow \{\{w\} \mid w \in Subs\}$;
3 **while** *True* **do**
4 $\quad Pairs \leftarrow \{(x, y) \mid x, y \in \mathcal{CC},\ x \neq y, |S|_x \geq n\ ,\ |S|_y \geq n\}$;
5 $\quad$ **if** $|Pairs| = 0$ **then exitloop** ;
6 $\quad$ Order $Pairs$ based on $\texttt{dist}_k$;
7 $\quad (x, y) \leftarrow Pairs[0]$;
8 $\quad init = \{[w]_{\mathcal{CC}} \mid w \in S\}$;
9 $\quad$ **if** $\texttt{dist}_k(x, y) \geq d$ **and** $|init| \leq i$ **then exitloop** ;
10 $\quad \mathcal{CC} \leftarrow \texttt{Merge}(x, y, \mathcal{CC})$;
11 **end**
12 **return** $\mathcal{CC}$;

At the beginning, each substring (or phrase for natural language) in the sample is assigned its own congruence class (line 2). Then, pairs of *frequent* congruence classes are *merged* together depending on the *distance* between their *empirical context distribution*, which is calculated on *local contexts*. The following points explain each keyword:

- The *empirical context distribution* of a substring w is simply a probability distribution over all the contexts of w, where the probability for a context (l, r) is the number of occurrences of lwr in the sample divided by the number of occurrences of w. This is extended to congruence classes by treating each substring in the class as one substring (i.e. the sum of occurrences of lw_ir, for all w_i in the class, divided by the sum of occurrences of all w_i).
- Due to the problem of sparsity with contexts (in any reasonably sized corpus of natural language, very few phrases will have more than one occurrence of the same context), only *local contexts* are considered. The local contexts of w are the pairs of first k symbols (or words for natural language) preceding and following w. The lower k is, the less sparsity is a problem, but the empirical context distribution is less accurate. For natural language corpora, k is normally set to 1 or 2.
- A *frequent* congruence class is one whose substring occurrences in the sample add up to more than a pre-defined threshold n. Infrequent congruence classes are ignored due to their unreliable empirical context distribution. However, as more merges are made, more substrings are added to infrequent classes, thus increasing their frequency and eventually they might be considered as frequent classes.
- A *distance* function `dist` between samples of distributions over contexts is needed by the algorithm to decide which is the closest pair of congruence classes, so that they are merged together. We used L1-Distance and Pearson's chi-squared test for experiments in Sections 4.1 and 4.2 respectively.
- After each *merge*, other merges are logically deduced so as to ensure that the relation re-

mains a congruence[1]. In practice, the vast majority of the merges undertaken are logically deduced ones. This clearly relieves the algorithm from taking unnecessary decisions (thus reducing the chance of erroneous decisions). On the downside, one bad merge can have a disastrous ripple effect. Thus, to minimize as much as possible the chance of this happening, every merge undertaken is the best possible one at that point in time (w.r.t. the distance function used). The same idea is used in DFA learning (Lang et al., 1998).

This process is repeated until either 1) no pairs of frequent congruence classes are left to merge (line 5) or 2) the smallest distance between the candidate pairs is bigger or equal to a pre-defined threshold d and the number of congruence classes containing strings from the sample is smaller or equal to a pre-defined threshold i (line 9).

The first condition of point 2 ensures that congruence classes which are sufficiently close to each other are merged together. The second condition of point 2 ensures that the hypothesized congruence classes are generalized enough (i.e. to avoid undergeneralization). For natural language examples, one would expect that a considerable number of sentences are grouped into the same class because of their similar structure. Obviously, one can make use of only one of these conditions by assigning the other a parameter value which makes it trivially true from the outset (0 for d and $|Subs|$ for i).

3.2 Building the Context-Free Grammar

Deciding which substrings are constituents (in our case, this translates into choosing which congruence classes correspond to non-terminals) is a problematic issue and is considered a harder task than the previous step (Klein, 2004). A path followed by a number of authors consists in using an Ockham's razor or Minimal Description Length principle approach (Stolcke, 1994; Clark, 2001; Petasis et al., 2004). This generally leads to choosing as best hypothesis the one which best compresses the data. Applying this principle in our case would mean that the non-terminals should be assigned in such a way that the grammar built is *the smallest possible* one (in terms of the number of non-terminals and/or production rules) consistent with the congruence classes. To our knowledge, only local greedy search is used by systems in the literature which try to follow this approach.

We propose a new method for tackling this problem. We show that all the possible SC-CFGs in CNF consistent with the congruence classes directly correspond to all the solutions of a boolean formula built upon the congruence classes, where the variables of this formula correspond to non-terminals (and, with some minor adjustments, production rules as well). Thus, finding the smallest possible grammar directly translates into finding a solution which has the smallest possible amount of true variables. Finding a minimal solution for this type of formula is a known NP-Hard problem (Khanna et al., 2000). However, sophisticated linear programming solvers (Berkelaar et al., 2008) can take care of this problem. For small examples (e.g. all the examples in Table 1), these solvers are able to find an exact solution in a few seconds. Moreover, these solvers are capable of finding good approximate solutions to larger formulas containing a few million variables.

The formula contains one variable per congruence class. All variables corresponding to congruence classes containing strings from the sample are assigned the value `True` (since there must be a starting non-terminal that generates these strings). All variables corresponding to congruence classes containing symbols from Σ are assigned the value `True` (since for every $a \in \Sigma$, there must be a rule $A \rightarrow a$). Finally, and most importantly, for every congruence class $[w]$ and for every string w in $[w]$ ($|w| = n$), the following conditional statement is added to the formula:

$$v(w) \Rightarrow (v(w_{1,1}) \wedge v(w_{2,n})) \vee (v(w_{1,2}) \wedge v(w_{3,n})) \vee \ldots \vee (v(w_{1,n-1}) \wedge v(w_{n,n}))$$

where $v(x)$ is the variable corresponding to the congruence class $[x]$ and $w_{i,j}$ is the substring of w from the i^{th} to the j^{th} symbol of w. This statement is representing the fact that if a congruence class $[w]$ is chosen as a non-terminal then for each string in $w \in [w]$, there must be at least one CNF rule $A \rightarrow BC$ that generates w and thus there must be at least one division of w into $w_{1,k}w_{k+1,n}$ such that B corresponds to $[w_{1,k}]$ and C corresponds to $[w_{k+1,n}]$.

The grammar extracted from the solution of this

[1] for example, if a congruence class contains the phrases "the big" and "that small", and another class contains "dog barked" and "cat meowed", it can be logically deduced that the phrases "the big dog barked","the big cat meowed", "that small dog barked" and "that small cat meowed" should be in the same class.

formula is made up of all the possible CNF production rules built from the chosen non-terminals. The starting non-terminals are those which correspond to congruence classes that contain at least one string from the sample.

The following is a run of the whole process on a simple example:

Sample $\{ab, aabb, aaabbb\}$

Congruence Classes

$1 : [a]$, $2 : [b]$, $3 : [ab, aabb, aaabbb]$, $4 : [aa]$, $5 : [bb]$, $6 : [aab, aaabb]$, $7 : [abb, aabbb]$, $8 : [aaa]$, $9 : [bbb]$, $10 : [aaab]$, $11 : [abbb]$

Boolean Formula

There is one conditional statement per substring. For example, $X_6 \Rightarrow (X_1 \wedge X_3) \vee (X_4 \wedge X_2)$ represents the two possible ways aab in congruence class 6 can be split ($a|ab$, $aa|b$).

Variables X_1, X_2 and X_3 are true.

$X_3 \Rightarrow (X_1 \wedge X_2)$
$X_3 \Rightarrow (X_1 \wedge X_7) \vee (X_4 \wedge X_5) \vee (X_6 \wedge X_2)$
$X_3 \Rightarrow (X_1 \wedge X_7) \vee (X_4 \wedge X_{11}) \vee (X_8 \wedge X_9) \vee (X_{10} \wedge X_5) \vee (X_6 \wedge X_2)$
$X_4 \Rightarrow (X_1 \wedge X_1)$
$X_5 \Rightarrow (X_2 \wedge X_2)$
$X_6 \Rightarrow (X_1 \wedge X_3) \vee (X_4 \wedge X_2)$
$X_6 \Rightarrow (X_1 \wedge X_3) \vee (X_4 \wedge X_7) \vee (X_8 \wedge X_5) \vee (X_{10} \wedge X_2)$
$X_7 \Rightarrow (X_1 \wedge X_5) \vee (X_3 \wedge X_2)$
$X_7 \Rightarrow (X_1 \wedge X_{11}) \vee (X_4 \wedge X_9) \vee (X_6 \wedge X_5) \vee (X_3 \wedge X_2)$
$X_8 \Rightarrow (X_1 \wedge X_4) \vee (X_4 \wedge X_1)$
$X_9 \Rightarrow (X_2 \wedge X_5) \vee (X_5 \wedge X_2)$
$X_{10} \Rightarrow (X_1 \wedge X_6) \vee (X_4 \wedge X_3) \vee (X_8 \wedge X_2)$
$X_{11} \Rightarrow (X_1 \wedge X_9) \vee (X_3 \wedge X_5) \vee (X_7 \wedge X_2)$

Solution

Running the solver on this formula will return the following true variables that make up a minimal solution: X_1, X_2, X_3 and X_7.

Grammar

For every statement $x \Rightarrow \ldots \vee (y \wedge z) \vee \ldots$ where x,y and z are true, a production rule $x \rightarrow yz$ is added. So, the following grammar is built:

X_3 is the starting non-terminal
$X_3 \rightarrow X_1X_7 \mid X_1X_2 \quad X_7 \rightarrow X_3X_2$
$X_1 \rightarrow a \quad X_2 \rightarrow b$

3.3 Analysis

In the first phase of the algorithm, we are grouping all the substrings of the sample S according to the congruence relation $\cong_{(L,\phi)}$, where (L, ϕ) is the target stochastic language (for natural language, this is the language model). To do so, we are assuming that S was i.i.d. generated from (L, ϕ). In the second phase, we are representing the space of all CFGs consistent with the classes obtained in phase one as different solutions to a boolean formula. Here we introduce our bias in favour of smaller grammars by finding a minimal solution to the formula. In the last phase, probabilities are assigned to the grammar obtained in phase two using the standard MLE algorithm for PCFGs.

Unlike many other systems, in our case the hypothesis space of grammars is well-defined. This allows us to analyse our algorithm in a theoretical framework and obtain theoretical learnability results. Moreover, this gives us an idea on the types of syntactical features our system is capable of learning.

Assuming our algorithm always takes correct merge decisions, the sample required for identification needs only to be structurally complete w.r.t. the target grammar (i.e. every production rules is used at least once in the generation of the sample). This means that, in theory, our algorithm can work with very small samples (polynomial size w.r.t. the number of rules in the target grammar).

Some approaches in the literature assume that whenever a particular substring is a constituent in some sentence, then it is automatically a constituent in all other sentences (whenever it does not overlap with previously chosen constituents) (van Zaanen, 2001; Clark, 2001; Adriaans et al., 2000). In reality, this is clearly not the case. A simple experiment on the WSJ10 corpus reveals that only 16 of the most frequent 1009 POS sequences (occurring 10 or more times in the sample) which are at least once constituents, are in fact always constituents. This assumption does not hold for ambiguous grammars in our class.

The approach we take to solve the smallest grammar problem can be extended to other classes of grammars. A similar formula can be built for grammars whose non-terminals have a one-to-one correspondence with congruence classes containing features of their language (Clark, 2010b).

4 Experiments and Discussion

4.1 Experiments on Artificial Data

We tested our system on 11 typical context-free languages and 9 artificial natural language grammars taken from 4 different sources (Stolcke, 1994; Langley and Stromsten, 2000; Adriaans et al., 2000; Solan et al., 2005). The 11 CFLs include 7 described by unambiguous grammars:

UC1: $a^n b^n$ **UC2**: $a^n b^n c^m d^m$ **UC3**: $a^n b^m$ $n \geq m$ **UC4**: $a^p b^q$, $p \neq q$ **UC5**: Palindromes over alphabet $\{a, b\}$ with a central marker **UC6**:Palindromes over alphabet $\{a, b\}$ without a central marker **UC7**: Lukasiewicz language ($S \rightarrow aSS|b$)

and 4 described by ambiguous grammars:

AC1: $|w|_a = |w|_b$ **AC2**: $2|w|_a = |w|_b$ **AC3**: Dyck language **AC4**: Regular expressions.

The 9 artificial natural language grammars are:

NL1: Grammar 'a', Table 2 in (Langley and Stromsten, 2000) **NL2**: Grammar 'b', Table 2 in (Langley and Stromsten, 2000) **NL3**: Lexical categories and constituency, pg 96 in (Stolcke, 1994) **NL4**: Recursive embedding of constituents, pg 97 in (Stolcke, 1994) **NL5**: Agreement, pg 98 in (Stolcke, 1994) **NL6**: Singular/plural NPs and number agreement, pg 99 in (Stolcke, 1994) **NL7**: Experiment 3.1 grammar in (Adriaans et al., 2000) **NL8**: Grammar in Table 10 (Adriaans et al., 2000) **NL9**: TA1 grammar in (Solan et al., 2005).

The quality of the learned model depends on its capacity to predict the correct structure (parse trees) on the one hand and to predict the correct sentence probabilities on the other (i.e. assigns a probability distribution close to the target one). To evaluate parse trees, we follow suggestions given by van Zaanen and Geertzen (2008) and use micro-precision and micro-recall over all the non-trivial brackets. We take the harmonic mean of these two values to obtain the Unlabelled brackets F_1 score (UF_1). The learned distribution can be evaluated using perplexity (when the target distribution is not known) or some similarity metric between distributions (when the target distribution is known). In our case, the target distribution is

Ex.	$\|\Sigma\|$	$\|N\|$	$\|P\|$
UC1	2	3	4
UC2	4	7	9
UC3	2	3	5
UC4	2	5	9
UC5	2	3	5
UC6	2	3	8
UC7	2	2	3
AC1	2	4	9
AC2	2	5	11
AC3	2	3	5
AC4	7	8	13
NL1	9	8	15
NL2	8	8	13
NL3	12	10	18
NL4	13	11	22
NL5	16	12	23
NL6	19	17	32
NL7	12	3	9
NL8	30	10	35
NL9	50	45	81

Table 1: Size of the alphabet, number of non-terminals and productions rules of the grammars.

		Relative Entropy		UF_1	
Ex.	$\|S\|$	COMINO	ADIOS	COMINO	ABL
UC1	10	**0.029**	1.876	**100**	**100**
UC2	50	**0.0**	1.799	**100**	**100**
UC5	10	**0.111**	7.706	**100**	**100**
UC7	10	**0.014**	1.257	**100**	27.86
AC1	50	**0.014**	4.526	**52.36**	35.51
AC2	50	**0.098**	6.139	**46.95**	14.25
AC3	50	**0.057**	1.934	**99.74**	47.48
AC4	100	**0.124**	1.727	**83.63**	14.58
NL7	100	**0.0**	0.124	**100**	**100**
NL1	100	**0.202**	1.646	**24.08**	**24.38**
NL2	200	**0.333**	0.963	**45.90**	**45.80**
NL3	100	**0.227**	1.491	36.34	**75.95**
NL5	100	**0.111**	1.692	**88.15**	79.16
NL6	400	**0.227**	**0.138**	36.28	**100**
UC3	100	**0.411**	0.864	61.13	**100**
UC4	100	**0.872**	2.480	42.84	**100**
UC6	100	1.449	**1.0**	**20.14**	8.36
NL4	500	**1.886**	2.918	**65.88**	52.87
NL8	1000	**1.496**	**1.531**	**57.77**	50.04
NL9	800	1.701	**1.227**	12.49	**28.53**

Table 2: Relative Entropy and UF_1 results of our system COMINO vs ADIOS and ABL respectively. Best results are highlighted, close results (i.e. with a difference of at most 0.1 for relative entropy and 1% for UF_1) are both highlighted

known. We chose relative entropy[2] as a good measure of distance between distributions.

Our UF_1 results over test sets of one thousand strings were compared to results obtained by ABL (van Zaanen, 2001), which is a system whose primary aim is that of finding good parse trees (rather than identifying the target language). Although ABL does not obtain state-of-the-art results on natural language corpora, it proved to be the best system (for which an implementation is readily available) for unsupervised parsing of sentences generated by artificial grammars. Results are shown in Table 1.

We calculated the relative entropy on a test set of one million strings generated from the target grammar. We compared our results with ADIOS (Solan et al., 2005), a system which obtains competitive results on language modelling (Waterfall et al., 2010) and whose primary aim is of correctly identifying the target language (rather than finding good parse trees). Results are shown in Table 1.

For the tests in the first section of Table 1 (i.e. above the first dashed line), our algorithm was capable of exactly identifying the structure of the target grammar. Notwithstanding this, the bracketing results for these tests did not always yield perfect scores. This happened whenever the target grammar was ambiguous, in which case the most probable parse trees of the target and learned grammar can be different, thus leading to incorrect bracketing. For the tests in the second section of Table 1 (i.e. between the two dashed lines), our algorithm was capable of exactly identifying the target language (but not the grammar). In all of these cases, the induced grammar was slightly smaller than the target one. For the remaining tests, our algorithm did not identify the target language. In fact, it always overgeneralised. The 3 typical CFLs UC3, UC4 and UC6 are not identified because they are not contained in our subclass of CFLs. Inspite of this, the relative entropy results obtained are still relatively good. Overall, it is fair to say that the results obtained by our system, for both language modelling and unsupervised parsing on artificial data, are competitive with the results obtained by other methods.

[2]The relative entropy (or Kullback-Leibler divergence) between a target distribution D and a hypothesized distribution D' is defined as $\sum_{w \in \Sigma^*} ln\Big(\frac{D(w)}{D'(w)}\Big) D(w)$. Add-one smoothing is used to solve the problem of zero probabilities.

4.2 Natural Language Experiments

We also experimented on natural language corpora. For unsupervised parsing, we tested our system on the WSJ10 corpus, using POS tagged sentences as input. Due to time efficiency, we changed the algorithm for finding congruence classes. Instead of always choosing the best possible merge w.r.t. the distance function, a distance threshold is set and all congruence classes whose distance is smaller than the threshold are merged. Also, we changed the distance function from L1-Distance to Pearson's χ^2 test.

In a first experiment (vaguely similar to the one done by Luque and López (2010)), we constructed the best possible SC-CFG consistent with the merges done in the first phase and assigned probabilities to this grammar using Inside-Outside. In other words, we ran the second phase of our system in a supervised fashion by using the treebank to decide which are the best congruence classes to choose as non-terminals. The CNF grammar we obtained from this experiment (COMINO-UBOUND) gives very good parsing results which outperform results from state-of-the-art systems DMV+CCM (Klein, 2004), U-DOP (Bod, 2006a), UML-DOP (Bod, 2006b) and Incremental (Seginer, 2007) as shown in Table 2. Moreover, the results obtained are very close to the best results one can ever hope to obtain from any CNF grammar on WSJ10 (CNF-UBOUND) (Klein, 2004). However, the grammar we obtain does not generalise enough and does not describe a good language model. In a second experiment, we ran the complete COMINO system. The grammar obtained from this experiment did not give competitive parsing results.

The first experiment shows that the merge decisions taken in the first phase do not hinder the possibility of finding a very good grammar for parsing. This means that the merge decisions taken by our system are good in general. Manual analysis on some of the merges taken confirms this. This experiment also shows that there exists a non-trivial PCFG in our restrictive class of grammars that is capable of achieving very good parsing results. This is a positive sign for the question of how adequate SC-PCFGs are for modelling natural languages. However, the real test remains that of finding SC-PCFGs that generate good bracketings *and* good language models. The second experiment shows that the second phase of our al-

Model	UP	UR	UF_1
State-of-the-art			
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
Upper bounds			
COMINO-UBOUND	75.8	96.9	85.1
CNF-UBOUND	78.8	100.0	88.1

Table 3: Parsing results on WSJ10. Note that *Incremental* is the only system listed as state-of-the-art which parses from plain text and can generate non-binary trees

gorithm is not giving good results. This means that the smallest possible grammar might not be the best grammar for parsing. Therefore, other criteria alongside the grammar size are needed when choosing a grammar consistent with the merges.

4.3 Discussion and Future Work

In order to improve our system, we think that our algorithm has to take a less conservative merging strategy in the first phase. Although the merges being taken are mostly correct, our analysis shows that not enough merging is being done. The problematic case is that of taking merge decisions on (the many) infrequent long phrases. Although many logically deduced merges involve infrequent phrases and also help in increasing the frequency of some long phrases, this proved to be not enough to mitigate this problem. As for future work, we think that clustering techniques can be used to help solve this problem.

A problem faced by the system is that, in certain cases, the statistical evidence on which merge decisions are taken does not point to the intuitively expected merges. As an example, consider the two POS sequences "DT NN" and "DT JJ NN" in the WSJ corpus. Any linguist would agree that these sequences are substitutable (in fact, they have lots of local contexts in common). However, statistical evidence points otherwise, since their context distributions are not close enough. This happens because, in certain positions of a sentence, "DT NN" is far more likely to occur than "DT JJ NN" (w.r.t. the ratio of their total frequencies) and in other positions, "DT JJ NN" occurs more than expected. The following table shows the frequencies of these two POS sequences over the whole WSJ corpus and their frequencies in contexts (#,VBD) and (IN,#) (the symbol # represents the end or beginning of a sentence):

	Totals	(#,VBD)	(IN,#)
"DT NN"	42,222	1,034	2,123
"DT JJ NN"	15,243	152	1,119
Ratios	3.16	6.80	1.90

It is clear that the ratios do not match, thus leading to context distributions which are not close enough. Thus, this shows that basic sequences such as "DT NN" and "DT JJ NN", which linguists would group into the same concept NP, are statistically derived from different sub-concepts of NP. Our algorithm is finding these sub-concepts, but it is being evaluated on concepts (such as NP) found in the treebank (created by linguists).

From the experiments we did on artificial natural language grammars, it resulted that the target grammar was always slightly bigger than the learned grammar. Although in these cases we still managed to identify the target language or have a good relative entropy result, the bracketing results were in general not good. This and our second experiment on the WSJ10 corpus show that the smallest possible grammar might not be the best grammar for bracketing. To not rely solely on finding the smallest grammar, a bias can be added in favour of congruence classes which, according to constituency tests (like the Mutual Information criterion in Clark (2001)), are more likely to contain substrings that are constituents. This can be done by giving different weights to the congruence class variables in the formula and finding the solution with the smallest sum of weights of its true variables.

The use of POS tags as input can also have its problems. Although we solve the lexical sparsity problem with POS tags, at the same time we lose a lot of information. In certain cases, one POS sequence can include raw phrases which ideally are not grouped into the same congruence class. To mitigate this problem, we can use POS tags only for rare words and subdivide or ignore POS tags for frequent words such as determinants and prepositions. This will reduce the number of raw phrases represented by POS sequences whilst keeping lexical sparsity low.

5 Conclusion

We defined a new class of PCFGs that adequately models natural language syntax. We described a learning algorithm for this class which scales well to large examples and is even capable of learning from small samples. The grammars induced by this algorithm are compact and perform well on unsupervised parsing and language modelling of typical CFLs and artificial natural language grammars.

Acknowledgements

The authors acknowledge partial support by the Région des Pays de la Loire.

References

Pieter W. Adriaans, Marten Trautwein, and Marco Vervoort. Towards High Speed Grammar Induction on Large Text Corpora. In Václav Hlavác, Keith G. Jeffery, and Jirí Wiedermann, editors, *SOFSEM*, volume 1963 of *Lecture Notes in Computer Science*, pages 173–186. Springer, 2000.

Michel Berkelaar et al. lpSolve: Interface to Lp solve v. 5.5 to solve linear/integer programs. *R package version*, 5(4), 2008.

Luc Boasson and Géraud Sénizergues. NTS Languages Are Deterministic and Congruential. *J. Comput. Syst. Sci.*, 31(3):332–342, 1985.

Rens Bod. Unsupervised Parsing with U-DOP. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 85–92, Stroudsburg, PA, USA, 2006a. Association for Computational Linguistics.

Rens Bod. An All-Subtrees Approach to Unsupervised Parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 865–872. Association for Computational Linguistics, 2006b.

Alexander Clark. *Unsupervised Language Acquisition: Theory and Practice*. PhD thesis, University of Sussex, 2001.

Alexander Clark. Distributional Learning of Some Context-Free Languages with a Minimally Adequate Teacher. In Sempere and García (2010), pages 24–37.

Alexander Clark. Towards General Algorithms for Grammatical Inference. In Marcus Hutter, Frank Stephan, Vladimir Vovk, and Thomas Zeugmann, editors, *ALT*, volume 6331 of *Lecture Notes in Computer Science*, pages 11–30. Springer, 2010b.

Alexander Clark and Rémi Eyraud. Polynomial Identification in the Limit of Substitutable Context-free Languages. *Journal of Machine Learning Research*, 8:1725–1745, 2007.

Alexander Clark and Shalom Lappin. Unsupervised Learning and Grammar Induction. In Alexander Clark, Chris Fox, and Shalom Lappin, editors, *The Handbook of Computational Linguistics and Natural Language Processing*, pages 197–220. Wiley-Blackwell, 2010.

Alexander Clark, François Coste, and Laurent Miclet, editors. *Grammatical Inference: Algorithms and Applications, 9th International Colloquium, ICGI 2008, Saint-Malo, France, September 22-24, 2008, Proceedings*, volume 5278 of *Lecture Notes in Computer Science*, 2008. Springer.

Colin de la Higuera. Characteristic Sets for Polynomial Grammatical Inference. *Machine Learning*, 27(2):125–138, 1997.

Colin de la Higuera. *Grammatical Inference: Learning Automata and Grammars*. 2010.

E. Mark Gold. Language Identification in the Limit. *Information and Control*, 10(5):447–474, 1967.

James Jay Horning. *A Study of Grammatical Inference*. PhD thesis, 1969.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. Bayesian Inference for PCFGs via Markov Chain Monte Carlo. In Candace L. Sidner, Tanja Schultz, Matthew Stone, and ChengXiang Zhai, editors, *HLT-NAACL*, pages 139–146. The Association for Computational Linguistics, 2007.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition) (Prentice Hall Series in Artificial Intelligence)*. Prentice Hall, 2 edition, 2008.

Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The Approximability of Constraint Satisfaction Problems. *SIAM J. Comput.*, 30(6):1863–1920, 2000.

Dan Klein. *The Unsupervised Learning of Natural Language Structure*. PhD thesis, Stanford University, 2004.

Kevin J. Lang, Barak A. Pearlmutter, and Rodney A. Price. Results of the Abbadingo One DFA Learning Competition and a New Evidence-Driven State Merging Algorithm. In Vasant Honavar and Giora Slutzki, editors, *ICGI*, volume 1433 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 1998.

Pat Langley and Sean Stromsten. Learning Context-Free Grammars with a Simplicity Bias. In Ramon López de Mántaras and Enric Plaza, editors, *ECML*, volume 1810 of *Lecture Notes in Computer Science*, pages 220–228. Springer, 2000.

Karim Lari and Steve J. Young. The Estimation of Stochastic Context-Free Grammars using the Inside-Outside Algorithm. *Computer Speech & Language*, 4(1):35 – 56, 1990.

Franco M. Luque and Gabriel G. Infante López. Bounding the Maximal Parsing Performance of Non-Terminally Separated Grammars. In Sempere and García (2010), pages 135–147.

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 2001.

Paola Merlo, Harry Bunt, and Joakim Nivre. Current Trends in Parsing Technology. In *Trends in Parsing Technology*, pages 1–17. Springer, 2010.

Georgios Petasis, Georgios Paliouras, Constantine D. Spyropoulos, and Constantine Halatsis. eg-GRIDS: Context-Free Grammatical Inference from Positive Examples Using Genetic Search. In Georgios Paliouras and Yasubumi Sakakibara, editors, *ICGI*, volume 3264 of *Lecture Notes in Computer Science*, pages 223–234. Springer, 2004.

Gisa Rauh. *Syntactic Categories: Their Identification and Description in Linguistic Theories*. Oxford Surveys in Syntax & Morphology No.7. OUP Oxford, 2010.

Yoav Seginer. Fast Unsupervised Incremental Parsing. In John A. Carroll, Antal van den Bosch, and Annie Zaenen, editors, *ACL*. The Association for Computational Linguistics, 2007.

José M. Sempere and Pedro García, editors. *Grammatical Inference: Theoretical Results and Applications, 10th International Colloquium, ICGI 2010, Valencia, Spain, September 13-16, 2010. Proceedings*, volume 6339 of *Lecture Notes in Computer Science*, 2010. Springer.

Zach Solan, David Horn, Eytan Ruppin, and Shimon Edelman. Unsupervised learning of natural languages. *Proceedings of the National Academy of Sciences of the United States of America*, 102(33):11629–11634, 2005.

Andreas Stolcke. *Bayesian learning of probabilistic language models*. PhD thesis, University of California, Berkeley, 1994.

Menno van Zaanen. *Bootstrapping Structure into Language: Alignment-Based Learning*. PhD thesis, University of Leeds, 2001.

Menno van Zaanen and Jeroen Geertzen. Problems with Evaluation of Unsupervised Empirical Grammatical Inference Systems. In Clark et al. (2008), pages 301–303.

Heidi R. Waterfall, Ben Sandbank, Luca Onnis, and Shimon Edelman. An Empirical Generative Framework for Computational Modeling of Language Acquisition. *Journal of Child Language*, 37:671–703, 6 2010.

Charles S. Wetherell. Probabilistic Languages: A Review and Some Open Questions. *ACM Comput. Surv.*, 12(4):361–379, 1980.

Charles Yang. Computational Models of Syntactic Acquisition. *Wiley Interdisciplinary Reviews: Cognitive Science*, 3(2):205–213, 2012.

Ryo Yoshinaka. Identification in the Limit of k, l-Substitutable Context-Free Languages. In Clark et al. (2008), pages 266–279.