A Survey of Unsupervised Grammar Induction

Baskaran Sankaran

November 25, 2010

Contents

1 Introduction

Automatic induction of natural language grammars by machines has retained sustained interest in the field of Computational Linguistics (CL) and remains an open problem till date. It also has significant consequences in several other areas including cognitive science, linguistics and psycholinguistics to name a few. From a domain independent perspective, the goal of an unsupervised grammar induction system is to mimic children: learning a grammar that can generate infinite number of grammatically valid utterances from a finite amount of data. Looked from the perspective of Natural Language Processing (NLP) - an unsupervised grammar induction algorithm could be useful for numerous applications such as language modelling and Statistical Machine Translation (SMT).

The research on unsupervised induction of natural language grammar started attracting wider interest with the introduction of inside-outside algorithm (Baker, 1979) for generating probabilistic context-free grammars (PCFG). This report presents an exhaustive survey of the different techniques used in unsupervised grammar induction, given a finite and large enough number of example sentences in a language. The wider applicability of the grammar induction in applications with multi-lingual setting, such as machine translation, raises questions that are even more exciting and challenging than the traditional grammar induction problem. Consider for example the question of whether we can learn synchronous grammar rules that can translate from a source to a target language given a finite $bitext$. A brief survey of the works focussing on applications of unsupervised grammar induction, specifically on its intersection with the statistical machine translation is also presented.

There have been broadly two types of methods that have been proposed, the first one assumes a fixed model for the grammar and searches for optimal parameters (for the assumed structure) that maximizes the data likelihood using the well-known EM algorithm. The second class of works also include a structural search typically using the idea of distributional clustering to group terminal sequences. The methods also differ in whether they consider grammar induction as a monolithic task or as consisting of two sub-problems of constituent identification and constituent labelling (see Section 1.2) and this has an effect on whether it produces CFG-style rules or just constituent/destituent sequences of one of more types.

Typically unsupervised approaches to grammar induction algorithm use raw text without any annotations. However to avoid the sparsity issue associated with words, several approaches use the Parts-Of-Speech (POS) tags instead of the raw sentences and induce structure over the POS tag sequence. Often these POS tags are manually annotated in the corpus, bringing in an element of human effort, though some of the recent works experiment with automatically induced POS tags or word-classes. In another dimension, these approaches use different grammar formalisms with phrase-structure and dependency grammars being the important ones and these will be noted at relevant sections in the report. Evaluation of unsupervised grammar induction is also challenging because of the differences in the type of structure learnt or grammar formalisms used.

The report is structured as below: a brief section on motivation discussing the significance of unsupervised grammar induction problem on different fields follows this. A formal problem definition is then presented to complete this section. The two classes of methods proposed for this problem are presented in Sections 2 and 3 followed by a detailed discussion of the results in Section 4. A comprehensive survey of the different evaluation metrics proposed for this task is summarized in Section 5. Turning towards applications Section 6 sketches the research direction for synchronous grammar induction targeted for SMT and finally Section 7 concludes the report.

1.1 Motivation

Historically, the question of how children learn their first language merely by listening to the speech of their parents or others in the surroundings, has fascinated the researchers in different disciplines including Cognitive Science and Linguistics. These speeches provide a set of positive examples for the child. The mistakes made by them, especially during the initial years are few and most often these are not corrected by their parents, which might serve as a negative example for the child in the learning process. The *Poverty of Stimulus* theory postulated by Chomsky (1980) suggest some innate support in the brain that enables the learning of infinitely powerful grammar from the finite data. This theory was further strengthened by examples of complex sentence structures, which when heard by a child, would require it to recognize and process specific syntactic units in order to generalize into grammar from the examples. The arguments against the poverty of stimulus theory questions some of the assumptions made by Chomsky especially the point about negative evidence and cite the availability of *indirect* negative evidence (Pullum and Scholz, 2002), which is additionally supported by the reasonable success of unsupervised grammar induction algorithms in learning hierarchical structures from the positive examples. However, the argument remains open and is also controversial.

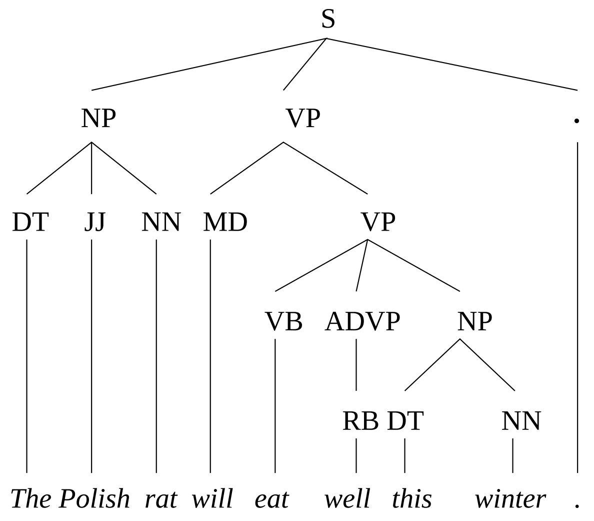
Secondly, the long and extensive research by linguists in studying the syntax of natural languages demonstrates the regularities in them in the form of hidden structures. These structures can take on different forms such as different types of constituent phrases (NP, VP etc.) in phrase-structure grammar or the head-dependent relations (sentence predicate and subject noun, determiner and noun etc.) in dependency grammar, among others. These grammar formalisms capture different aspects about the syntax of a language and hence have validity in their own terms.

For example, Figures (1a) and (1b) show the syntactic structure of a sentence in phrase-structure and dependency grammar formalisms. The phrase structure tree in Figure (1a) gives the constituent phrases (noun phrase: *The Polish rat*; verb phrase: *eat will in the winter*) and also encodes the order in which the tree is derived from the top symbol S. In contrast, the dependency structure captures the relations between a word and its dependents (ex. the auxiliary verb *will* has *rat*, *eat* and . as dependents; determiner *this* is the dependent of the head noun *winter*). One of the differences between phrase-structure and dependency representations is that the latter does not model the derivation order explicitly, which is sometimes also considered as an advantage. The dependency structure can also be formulated as context-free rules and subsequently represented in an alternate tree structure as in Figure (1c), which is very similar to phrase-structure representation in Figure (1a).

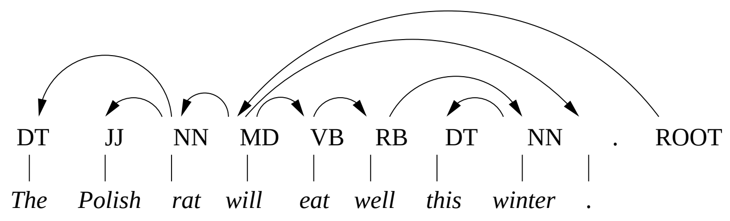
While, this clearly show the existence of the hidden structures in languages, the key question is whether these structures could be automatically discovered from finite data. A related question is to identify the grammar formalism that is best suited for such learning, which typically depends on the application that uses the induced grammar.

Thirdly, grammar induction is the key step that helps in analyzing text syntactically and is helpful in further downstream processing tasks such as semantic understanding or mapping to a target language syntax for producing better translation or for simply checking the grammaticality of the sentences. Supervised approaches to grammar induction rely on the availability of treebank data where the sentences are manually annotated with syntactic information in the form of hierarchical trees as in Figure (1a). The major drawback with these methods is their dependence on such manually annotated resources and such resources are available for English and few other resource-rich languages. Similarly, the domain of the annotated data is an important factor: for example a treebank consisting mainly of articles in a narrow domain, say business or air-traffic inquiries, may not model the sentence structures in other domains but in the same language. Hence these methods only have limited applicability in resource-poor languages or domains.

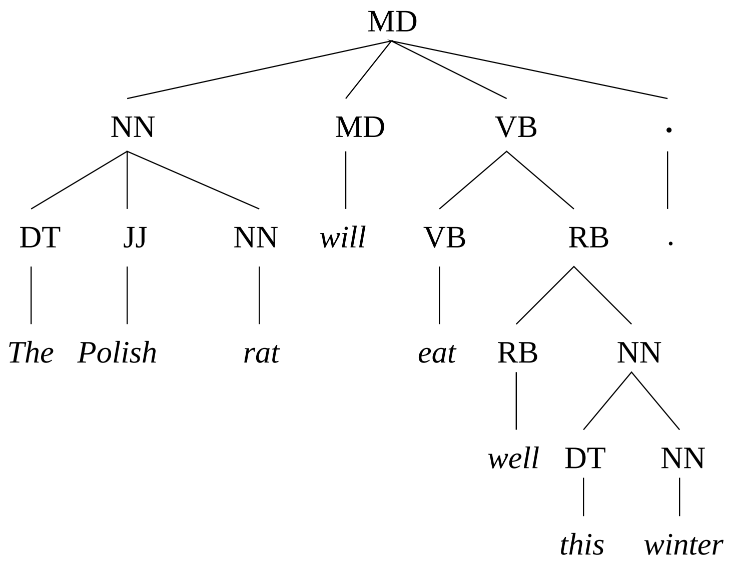
The idea of unsupervised grammar induction using only limited positive examples is thus an attractive proposition having significant implications from both practical and philosophical considerations. As discussed in this report, various research works has shown promising results but still offer significant scope for improvement.



(a) Constituency Tree



(b) Dependency Structure



(c) Dependency CF Tree

Figure 1: Different types of Structures in a Sentence

1.2 Problem Definition

A grammar G represents a language L in terms of its smaller constituent blocks and describes how these blocks interact recursively to build sentences that are grammatically valid in that language. In other words, the grammar G specifies how the terminal symbol sequences in a given sentence, can be parenthesized (possibly with embedded structures), such that no two parentheses cross each other and the sequence of symbols within each parenthesis forms a linguistically coherent structure.

The problem of inducing a grammar - given a corpus C consisting of a finite number of valid sentences (positive examples) S and having a finite set of terminal symbols Σ, can be stated as below.

Induce a grammar G by first identifying sequences of terminal symbols α that form minimal and grammatically coherent units and then by grouping these sequences based on some syntactic notion of *similarity* into $|X|$ categories such that each category X_j (where $j = 1, \dots, |X|$) represents a distinct syntactic type. The above definition looks at the problem from the perspective of *clustering*.

The syntactical notion of similarity mentioned above reflects the nature of information captured by the grammar formalism used. It represents the different syntactic categories for phrase-structure grammars, whereas for dependency grammars it represents the relationships between heads and their corresponding dependents.

Formally the goal is to induce a set of production rules R that describe how to generate valid sentences S in language L, by starting from a special start symbol X_s that is distinct from each X_j (where $j = 1, \dots, |X|$) and recursively rewriting the non-terminals in the right-side of productions until no non-terminal remains. This latter one serves as a more formal definition of the task and also captures the *search* perspective of grammar induction.

Thus, the productions R capture the relationship between set of non-terminals X and non-terminal, terminal sequences $(X \cup \Sigma)\star$ by recursively rewriting the non-terminals as $X_i \rightarrow \alpha$ or $X_i \rightarrow \alpha X_j \beta$, where $X_i, X_j \in X$ and $\alpha, \beta \in \Sigma\star$. Additionally, the production rules R will typically have probabilities associated with them, signifying the likelihood of the individual rules being used in generating C.

The supervised approaches exploit the availability of annotated data commonly known as *tree-banks* for learning the grammar, whereas unsupervised approaches attempt to induce grammar directly from the raw text or more specifically the POS sequences, in order to avoid the sparsity issues associated with the words. In practical setting, the nature of grammar induced vary among different unsupervised approaches mainly due to differences in either the method employed (*search* or *clustering*) or the grammar formalism used or both.

1.2.1 Phrase Structure Induction

As noted earlier, phrase structure representation of a language encodes the constituent phrases of a given sentence by their syntactic categories (NP, VP etc.) and additionally order them by a series of production rules R indicating how the sentence is derived by combining them. Thus the phrase structure representation has an underlying grammar defined by the context-free rules. In a probabilistic setting, the rules in the grammar are assigned a probability signifying how often the rule is used in the language L generated by G.

For each string q_1^n generated by $L(G)$, only the generated string $q = q_1 \dots q_n$ is observed, while the set of parse trees T that yield q is unobserved. The probability of an observed sequence can be written as:

$$P_\Theta(q) = \sum_T P_\Theta(q, T) \tag{1}$$

where, the joint probability distribution $P_\Theta(q, T)$ can be parameterized by making a Markov assumption such that the rules in a derivation to be independent of each other.

$$P_\Theta(q, T) = \prod_r \Theta(r)^{c(r:q,T)} \tag{2}$$

The goal of the PCFG training is to find the parameters Θ that maximizes the likelihood of a corpus C.

$$L(\Theta) = \sum_{q \in C} \sum_{T \in \mathcal{T}(q)} P_\Theta(T, q|\Theta) \tag{3}$$

Towards this end, first the expected value Θ' of log-likelihood can be calculated, which can then be maximized to yield an updated value $\bar{\Theta}$ as:

$$Q(\Theta'|\Theta) = \sum_{q \in C} \sum_{T \in \mathcal{T}(q)} P_\Theta(T|q) \log \frac{P_{\Theta'}(T, q)}{P_\Theta(T, q)} \tag{4}$$

$$\bar{\Theta} = \underset{\Theta'}{\operatorname{argmax}}\, Q(\Theta'|\Theta) \tag{5}$$

Q can be maximized by taking the derivative of Equation4 and setting it to zero. In computing this, the expected counts of the rules used in the parse trees T that derive q needs to be computed. Now instead of summing over all parse trees $T \in \mathcal{T}(q)$, the Markov assumption mentioned above can be exploited to compute the expected count efficiently.

For example, consider the rule $X_i \rightarrow X_j X_k$ used to generate the sequence q_s^t (i.e $q_s q_{s+1} \dots q_{t-1} q_t$) in the derivation of the observed sentence q. Using independence assumption, the probability of generating the observation sequence q from the start symbol S can be written as:

$$\begin{aligned}P_\Theta(S \Rightarrow^* q_1 \dots q_n) = P_\Theta(q) \\ &= P_\Theta(S \Rightarrow q_1 \dots q_{s-1}\ X_i\ q_{t+1} \dots q_n) \times \\ &\quad p_\Theta(X_i \rightarrow X_j X_k) \times P(X_j \Rightarrow q_s \dots q_r) \times P(X_k \Rightarrow q_{r+1} \dots q_t)\end{aligned}$$

The inside-outside algorithm explained in Section 2, provides an efficient method for computing the expected counts by using the inside and outside probabilities.

2 Parameter Search with Fixed Models

One way of inducing a probabilistic grammar is to fix the structure of a grammar and then to find the set of optimal parameters such that the resulting grammar best explains the language, which is usually approximated by a training corpus. The inside-outside algorithm originally proposed by (Baker, 1979), generalized the forward-backward algorithm for the regular HMMs to probabilistic context-free grammars for speech recognition. It is an instance of the EM algorithm (Dempster et al., 1977) applied in the context of PCFG estimation and is used to find the frequency counts of the individual productions in the derivation of sentences in a corpus C.

The parametric search for natural language grammar, starts by assuming a fixed model structure, consisting of X non-terminals. It further assumes the productions to be in Chomsky Normal Form (CNF) shown below, resulting in fully binary branching derivations.

$$b_{ijk} : X_i \rightarrow X_j X_k \tag{6}$$

$$u_{im} : X_i \rightarrow q(m) \tag{7}$$

where X_i (for $i = 1, \dots, |X|$) are unique non-terminal entries and $q(m)$ is the m^{th} terminal symbol in the observation sequence. The first rule rewrites a non-terminal into two other non-terminals and the second rule generates a terminal symbol from a parent non-terminal. Thus, given a set of Σ terminal symbols and X non-terminals, it enumerates a total of $X^3 + X \cdot \Sigma$ production rules of the specified form and the goal of the inside-outside algorithm is to find the parameter estimates or rule probabilities for the productions in the model. The parameter estimates are constrained by the fact that the sum of the probabilities of all the rules generated by a given non-terminal should sum to 1 as in Equation (8).

$$\sum_{j,k} b_{ijk} + \sum_{m} u_{im} = 1 \text{ for each } X_i \in X \tag{8}$$

Given a list of productions, the model for generating a tree T yielding a sequence q can be written as in Equation (9), where $X_i(s,t)$ refers to the span (s,t) being rewritten by non-terminal X_i. The probability of q can then be obtained by marginalizing over all trees that yield the sentence q. Alternately, $P(q|\Theta)$ can be written in terms of *inside* and *outside* probabilities as in Equation (10).

$$P(T,q|\Theta) = \prod_{X_i(s,t)\rightarrow\alpha\in T} P(\alpha|X_i) \tag{9}$$

$$P(q|\Theta) = \prod_{i} I_i^q(1,n) O_i^q(1,n) \tag{10}$$

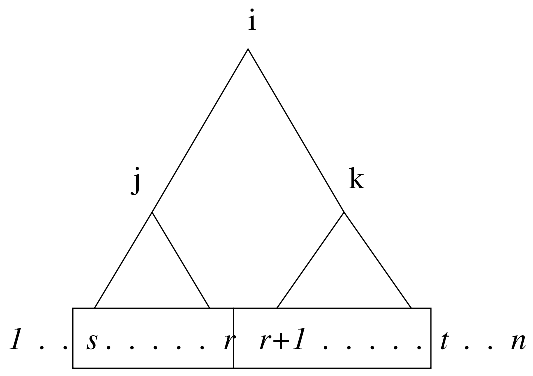


Figure 2: Computing *inside* probabilities - Figure from Lari and Young (1990)

The inside probability $I_i^q(s,t)$ defines the probability of generating a sub-tree rooted at a node X_i and deriving the observation sequence under it as in Figure (2). Formally, it is denoted as Equations (11) and (12) and is the probability of the non-terminal symbol X_i generating the fragment of observation sequence $q(s)\ldots q(t)$.

$$I_i^q(s,s) = P(u_{is}), \quad \text{when } s = t \tag{11}$$

$$I_i^q(s,t) = \sum_{j,k}\sum_{r=s}^{t-1} P(b_{ijk}) I_j^q(s,r) I_k^q(r+1,t) \tag{12}$$

The outside probabilities $O_i^q(s,t)$ are computed during the re-write process once the inner proba-

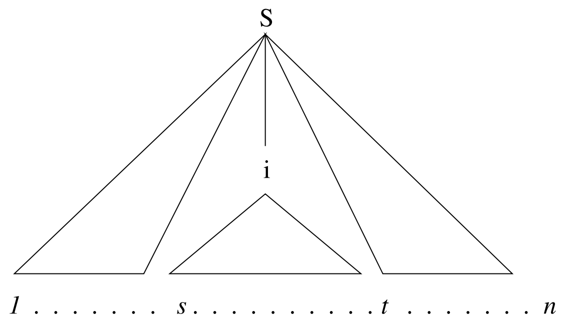
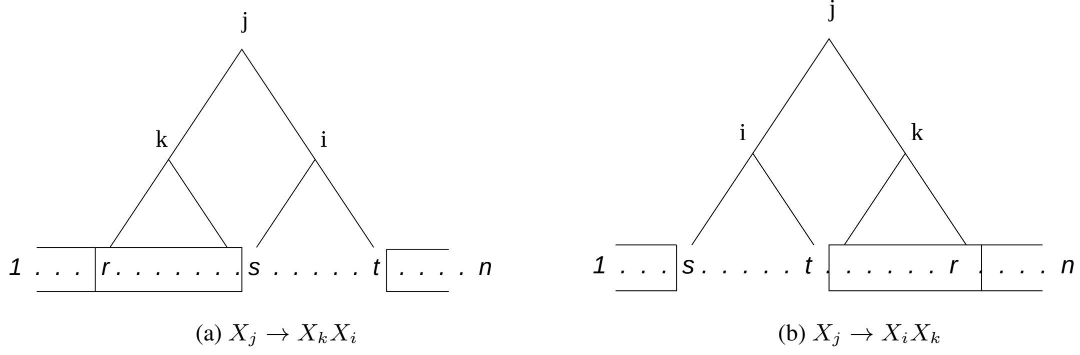


Figure 3: Definition of *outside* probabilities - Figure from Lari and Young (1990)


j
k
i
1 . . . r . . . . s . . . . . t . . . . n
j
i
k
1 . . . s . . . . . t . . . . . . r . . . . n

(a) $X_j \rightarrow X_k X_i$ (b) $X_j \rightarrow X_i X_k$

Figure 4: Computing *outside* probabilities - Figure from Lari and Young (1990)

bilities are found. As shown in Figure (3), it rewrites starting from the start symbol S and generates a non-terminal X_i that spans (s,t); and is defined as the probability of generating the sequence *outside* of X_i, i.e. $q(1)\dots q(s-1)\ X_i\ q(t+1)\dots q(n)$. The X_i non-terminal can then be part of either b_{jki} or b_{jik} and the respective computations are illustrated in Figures (4a) and (4b), which can be written as Equations (13) and (14).

$$O_i^q(1,n) = \begin{cases} 1 & \text{if } X_i = S, \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

$$O_i^q(s,t) = \sum_{j,k} \Big(O_j^q(r,t)P(b_{jki})I_k^q(r,s-1) + O_j^q(s,r)P(b_{jik})I_k^q(t+1,r) \Big) \tag{14}$$

The inside-outside re-estimation starts by assuming random parameter estimates for the rules in the grammar satisfying the above constraint in Equation (8) and uses these initial estimates to generate all possible parses for each sentence q in C. It then computes the expected fraction of parses of q having a node X_i generating the words in the span (s,t) using Equations (15) and (16).

$$w^q_{ijk}(s,t) = \frac{1}{P_q}\sum_{r=s}^{t-1} P(b_{ijk}) I^q_j(s,r) I^q_k(r+1,t) O^q_i(s,t) \tag{15}$$

$$v^q_{im}(s,t) = \frac{1}{P_q} I^q_i(s,t) O^q_i(s,t) \tag{16}$$

These counts are then aggregated and normalized appropriately to re-estimate new probability estimates for the rules in entire corpus C as in Equations (17) and (18).

$$P(b_{ijk}) = \frac{\sum_{q\in C}\sum_{s=1}^{n_q-1}\sum_{t=s+1}^{n_q} w^q_{ijk}(s,t)}{\sum_{q\in C}\sum_{s=1}^{n_q}\sum_{t=s}^{n_q} v^q_{im}(s,t)} \tag{17}$$

$$P(u_{im}) = \frac{\sum_{q\in C}\sum_{t\emptyset(t)=m} v^q_{im}(t,t)}{\sum_{q\in C}\sum_{s=1}^{n_q}\sum_{t=s}^{n_q} v^q_{im}(s,t)} \tag{18}$$

These steps are repeated iteratively until the convergence. The convergence is determined by a threshold variation in the entropy of the language L generated by grammar G.

While presenting a detailed training procedure for estimating the parameters Θ of a PCFG, Lari and Young (1990) used it to induce a grammar for an artificial *palindrome* language. By using the entropy minimization (equivalently maximizing the data likelihood $P(D|\Theta)$) of the generated language as an objective, Lari and Young showed the estimated PCFG to be better than a regular grammar estimated by an equivalent HMM. They also proposed pre-training methods for initializing the PCFG, so as to avoid local maxima and to enable faster convergence. While, such pre-training methods worked well for the simple palindrome language, Carroll and Charniak (1992) showed its ineffectiveness for natural language grammar.

Lari & Young further introduced some ideas to constrain the generated grammar, by explicitly assigning non-terminal symbols to each of the terminal symbols and forcing the remaining non-terminals to model the hidden process. This avoids the case, where few terminal symbols get modelled excessively by several non-terminals leaving out the rest. They also introduced a grammar minimization procedure to reduce the redundant non-terminals rewriting the frequent terminals. It works by first identifying the redundant non-terminals generating the same terminal and then chooses a non-terminal at random and fixes the corresponding production parameters. The production parameters of the other redundant non-terminals that produce same terminal are randomized, thereby making them available for uncovered terminal symbols corresponding to under-represented stochastic processes. While these pre-training technique and grammar minimization procedure facilitate faster convergence and improved grammar, the results are not impressive as the underlying language in Lari and Young is not a natural language but a simpler palindrome language.

2.1 Constraining Search

Carroll and Charniak (1992) presented a set of experiments in inducing probabilistic dependency grammar based on the approach by Baker (1979) and Lari and Young (1990). The approach uses the inside-outside algorithm but instead of assuming the full model involving all the productions for a given set of non-terminals, it begins with the set of all productions that are applicable for the sentences in the corpus and filters them further based on a threshold value on the re-estimated probabilities.

Carroll and Charniak also demonstrate that the search space of the inside-outside is riddled with local optima and in a specific experiment with 300 different initializations they show that all the induced grammars are i) different and ii) of extremely poor quality, when compared to the original grammar. The experiments also suggest the entropy minimization to be an ineffective choice as an objective for inside-outside algorithm, or in other words the entropy minimization does not necessarily lead to a better grammar.

In trying to address these issues, Carroll and Charniak used the grammar constraints to guide the search process for grammar induction. The general objective of these constraints is to eliminate a grammatically incompatible non-terminal being generated from a given non-terminal (for example an adjective generating a determiner or a verb generated from a pronoun). These linguistic constraints are specified in the form of a matrix, representing the non-terminals that are allowed on the right-side of each of the left non-terminal. The constraints are shown to be effective in avoiding syntactically untenable productions from being generated, while testing with a toy grammar. They also experiment on the efficacy of different subsets of constraints and show a *destituent* grammar - a grammar specifying the constituents that should be disallowed, to be better.

Pereira and Schabes (1992) present a different approach of incorporating linguistic constraints in the inside-outside re-estimation by exploiting the availability of annotated data. In this approach, a partial bracketing annotations as in the Penn treebank (PTB) corpus is used for inducing the PCFG and the inside-outside algorithm is modified appropriately to choose the productions that conform to the brackets in the corpus. Figure (5) shows some example bracketed sentences from PTB, specifically from the WSJ10 dataset[1].

```
( (S
    (NP-SBJ (DT The) (JJ 40-year-old) (NNP Mr.) (NNP Murakami) )
    (VP (VBZ is)
      (NP-PRD (DT a) (NN publishing) (NN sensation) )
      (PP-LOC (IN in)
        (NP (NNP Japan) )))
    (. .) ))


( (S
    (NP-SBJ (NNP Mr.) (NNP Baris) )
    (VP (VBZ is)
      (NP-PRD
        (NP (DT a) (NN lawyer) )
        (PP-LOC (IN in)
          (NP (NNP New) (NNP York) ))))
    (. .) ))


( (S
    (NP-SBJ (DT The) (JJ Japanese) (JJ industrial) (NNS companies) )
    (VP (MD should)
      (VP (VB know)
        (ADVP-CLR (JJR better) )))
    (. .) ))
```

Figure 5: Example Bracketed Sentences from Penn Treebank (WSJ10 set)

The bracketed examples are annotated with both Parts-of-Speech (POS) tags (found alongside the words inside the parentheses), such as VBZ, NN, DT and so on. It also includes the grammatical categories (marked at the top of the nodes in separate lines), such as ADVP, NP and PP. Pereira and Schabes use the bracketings to guide the learning, but ignore the grammatical categories in the annotations. Thus when the grammar is constrained to conform to the bracketings, it cannot identify a word sequence that crosses the gold bracketing as a constituent. As an example, the tag sequence

[1]Wall Street Journal (WSJ) section of PTB having 10 or fewer words after stripping off the symbols and punctuations

NNP NNP VBZ (corresponding to *Mr. Murakami is* in the first sentence) cannot be valid sequence, even though it appears frequently in the corpus.

Compatibility between the corpus and generated productions is ensured by an auxiliary function:

$$\overline{c}(i,j) = \begin{cases} 1, & \text{if } span(i,j) \text{ does not overlap any } b \in \mathcal{B} \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

The expressions for re-estimating the binary and unary counts is also modified to consider only the productions that are compatible with the bracketing. This approach effectively eliminates non-syntactic sequences in the right side of the productions and is shown to better model the hierarchical structure and also resulting in faster convergence. The evaluation shows over 90% of the identified constituents to be compatible with with the test set constituents on a simple ATIS corpus. It should however be noted that the specific requirement of annotated data might limit the applicability of this approach for languages with scarce resources.

Another simpler way of guiding the parametric search process would be to use a bootstrapping approach by specifying some seed examples for different constituent types and propogating the examples to induce the PCFG rules (Haghighi and Klein, 2006). Exploiting this idea in a *prototype-driven* approach, Haghighi and Klein (2006) start by manually specifying a small number of constituent yields for 7 major syntactic categories and then propogate these examples over the corpus to induce labels for new sequences by the notion of distributional similarity. While this is similar to other approaches based on distributional clustering (Clark, 2001; Klein and Manning, 2001a) (see Section 3.2), it uses a fixed model (with specific number of non-terminals X) and uses the bootstrapping to better guide the search process to avoid the local extremum.

Given a terminal POS sequence α and its context c (adjacent tags on either side or boundary), the distribution of α over its contexts is defined as its *signature* and denoted by $\sigma(\alpha)$. Given prototype sequences for each constituent phrases, it compares the signature $\sigma(\alpha)$ (of a new sequence α) with that of the set of prototypes of each phrase $\sigma(X_i)$ using a variant of KL divergence metric and assigns the sequence α to the type having least divergence or to a NONE type if the divergence is above a threshold.

$$X(\alpha) = \begin{cases} \textsc{None}, & \text{if } \min_{X_i} D_{SKL}(\sigma(\alpha), \sigma(X_i)) < t \\ \operatorname{argmin}_{X_i} D_{SKL}(\sigma(\alpha), \sigma(X_i)), & \text{otherwise} \end{cases} \tag{20}$$

where, $\sigma(X_i)$ is the signature of the phrase-type X_i and is computed by averaging the signatures of its prototypical examples provided manually. Incorporating this distributional information through a *prototype feature* (p_{st}), into the PCFG model in Equation (9) results in an augmented model written as:

$$P(T, q|\Theta) = \prod_{X_i(s,t) \to \alpha \in T} P(p_{st}|X) P(\alpha|X_i) \tag{21}$$

Intuitively, the prototype feature helps in finding the appropriate label for a sequence covered by a subtree in T. In evaluating the augmented model Haghighi and Klein (2006) used both raw corpus and bracketed corpus (Pereira and Schabes, 1992) and found significant gains in the labelled and unlabelled $F1$ scores, while using the additional bracketing information over the raw corpus. Prototype-driven learning was further integrated with the constituent-context model to label the constituent sequences as explained in section 3.3.

Unlike most of the approaches discussed earlier, the prototype-driven approach offers an easier mechanism for directly controlling the quality of the PCFG rules. For example, the bootstrapping sequences of certain constituent phrases can be modified on the basis of error analysis thereby resulting in improved grammar. The authors also demonstrate this by adding specific examples for capturing the possessive nouns and infinitival verbs.

2.2 Improving Learning with Negative Examples

The EM-style likelihood based approaches covered earlier use only positive examples in a corpus, which are likely to be grammatically correct than otherwise. Intuitively, these approaches attempt to move the probability mass to a set of production rules supported by the sentences observed but without identifying the productions from which the probability mass must be moved.

Now, contrasting this with the language acquisition in children, research has indicated to the availability of *indirect* negative evidence to them (Pullum and Scholz, 2002) giving them an opportunity to improve their learning. For example, as children listen to more utterances like *a red apple* or *that black dog*, they rule out the variations such as *red a apple*, *apple red a*, *black that dog* as ungrammatical, from the fact that they never hear them. Extending this analogy to the likelihood based approaches, it has been argued that negative evidences could help the EM to move the probability mass from the rules covering the negative evidences to the rules applicable to the positive examples. In other words, while the plain EM-style algorithms attempt to move probability mass to a set of rules blindly, the negative examples explicitly identify the productions that must be penalized. *Contrastive Estimation* proposed by Smith and Eisner (2005) exploits this idea of implicit negative evidence available in a sentence for the grammar induction task.

The idea is to generate for a given sentence q, a large neighbourhood $\mathcal{N}(q)$ of ungrammatical sentences as negative evidence, by perturbing q with certain operations. For example, a delete (DEL) operator generates sentences in the neighbourhood by deleting one word in q; it thus generates $n+1$ different sentences including q, where n is the length of q. Similarly, the transpose (TRANS) operator transposes two subsequent words in a sentence to generate its neighbourhood. These operators can also be combined together to produce an even larger neighbourhood. It can be seen that the sentences generated by such operations will most likely be ungrammatical.

The question of choosing an appropriate neighbourhood is task and language dependent; for example the transpose operator may not be good neighbourhood generator for inducing grammars for configurational languages that allows different word-orders unlike English. Smith and Eisner (2005) use three different neighbourhoods, viz. LENGTH, TRANS and DELORTRANS for their experiments in unsupervised dependency learning for English.

Contrastive estimation is a generic family, of which EM can be seen a specific case where the neighbourhood $\mathcal{N}(q)$ is the entire set of observed sentences. Comparing the objective functions maximized by regular EM and CE in Equations (22) and (23) respectively, it should be noted that CE approach is constrained by the neighbourhood of a sentence, whereas for the regular EM it is the entire set of derivations possible with the terminal symbols set $\Sigma\star$. This results in an intractable normalization for the EM, which is restricted in the CE to be the sentence's neighbourhood.

$$\prod_{q \in C} P(q|\Theta) = \prod_{q \in C} \sum_{T \in \mathcal{T}: yield(T)=q} P(T, q|\Theta) \tag{22}$$

$$\prod_{q \in C} P(q|\mathcal{N}(q), \Theta) = \prod_{q \in C} \sum_{T \in \mathcal{T}: yield(T)=q} P(T, q|\mathcal{N}(q), \Theta) \tag{23}$$

The CE approach by Smith and Eisner differs from other approaches in two ways. Firstly, instead of the PCFG they use weighted CFG (WCFG), which defines the probability as a score allowing the production rules to have arbitrary weights without being constrained by the fact that they have to sum to 1. The score of a (sentence, tree) pair is then normalized by the sum of the scores of all the structures allowed by the weighted CFG. Secondly, it views the grammar induction as MATCHLINGUIST task: the task of matching the human linguistic annotations in PTB and this is implicitly specified by the choice of an objective function that guides the grammar learning to that of matching the human annotations.

The approach by Smith and Eisner uses the DMV model (Klein and Manning, 2004) (cf Section 3.5) in a log-linear framework. Briefly, the DMV model treats a dependency tree T shown in Figure (1b) as an ordered (*head*, *argument*) pairs (denoted as (h, a)) and score the entire structure as the product of the probabilities of the individual dependencies. Apart from the argument (a) for a head (h), the model includes the direction of the dependency and the number of arguments for the head (valence) as separate features in the log-linear framework. The neighbourhood likelihood can then be defined in terms of the features as:

$$\mathcal{L}_{\mathcal{N}}(\Theta) = \sum_{q \in C} log \frac{\sum\limits_{T \in \mathcal{T}(q)} exp\Bigg(\Theta \cdot f(q,T)\Bigg)}{\sum\limits_{(q,T) \in \mathcal{N}(q) \times \mathcal{T}} exp\Bigg(\Theta \cdot f(q,T)\Bigg)} \tag{24}$$

where the feature functions $f(q,T)$ refer to the probability distributions of the DMV model $P(root)$, $P(\text{END}|\cdot))$ and $P(a|\cdot)$. Using these feature functions the model learns dependency structure for the WSJ10 dataset with two different initializers: a simple uniform initialization and an initializer similar to Klein and Manning (2004). The weights are learnt by the modified EM procedure as in Klein and Manning, but an additional smoothing step was added. CE has been shown to perform better than the simple generative EM as in Carroll and Charniak (1992) by a significant margin. Further results are discussed in Section 4.

3 Structural Search Approaches

The approaches covered in the earlier section used a fixed model (by fixing the set of productions R in the grammar G) while searching for the maximum likelihood parameters. Theoretically, this approach is expected to assign a lower probability for invalid productions and thus minimal score for an ungrammatical sentence. However it chooses a locally optimal solution due to modelling and search issues detailed above.

An alternative would be to search for an appropriate structure as dictated by the evidence provided by the data, so that the resulting grammar is compact in describing the data. These models rely on some heuristics for deciding the grammar constituents and use a Minimum Description Length (MDL)-style criterion that reduces the joint encoding of the induced grammar and data. Some of the earliest approaches in this category Stolcke and Omohundro (1994); van Zaanen (2000) and Adriaans et al. (2000), are based on the simple idea of clustering the word sequences, either bottom-up or top-down, to identify constituents which can then possibly be embedded as PCFG-style rules. This section begins by explaining these simpler approaches and then proceeds to the more formal approaches that employed a rigorous framework of distributional analysis or dependency parsing.

3.1 Minimum Description Length Approaches

Stolcke and Omohundro (1994) proposed a simple model merging approach that incrementally generalizes the grammar starting from an instance-based maximum likelihood grammar. Following a Bayesian approach, it uses the posterior probability of the model given the data in corpus C to guide the search process and maximizes the objective function in Equation (25). A normalization factor in the denominator of (25) has been dropped as it is invariant to the model M.

$$P(M|C) = P(M)P(C|M) \tag{25}$$

Based on the original model merging idea proposed by (Omohundro, 1992), it starts from an large instance-based initial model prior ($P(M_0)$) maximizing the likelihood and iteratively applies two *generalization* operators, viz. *merging* and *chunking* to generalize the grammar so as to maximize the posterior. At each iteration it explores all possible merging steps and choose the model(s) that result in the largest increase in the posterior. While Stolcke and Omohundro (1994) give a detailed sketch of an algorithm with a small example grammar, they do not give any objective evaluation of the model that would allow us to compare it with others.

van Zaanen and Adriaans (2001) present a comparative study of the two early systems Alignment-based Learning (ABL) (van Zaanen, 2000) and EMILE (Adriaans et al., 2000). ABL is based on the linguistic notion of *substitutability* (Harris, 1951) and is also used by several other approaches as we will see later. Briefly, it is defined as the ability of a constituent (sequence of words) to be substituted by another constituent (having different sequence of words) of same type without violating the syntax. Consider the example in examples (1) and (2) from the ATIS corpus with the underlined segment indicating the shared context. It is now easy to deduce that the phrases *a family fare* and *coach fare for flight 1943* are of same type based on the shared context they appear in. Thus, broadly, ABL or other similar methods depend on identifying the valid constituents from a text and grouping them by the constituent types.

(1) $\underline{\text{What is}}$ a family fare?

(2) $\underline{\text{What is}}$ coach fare for flight 1943?

ABL identifies constituents by reversing the notion of substitutability: if two sentence fragments can be substituted for each other then they are constituents of same type. The approach proposed by van Zaanen involves two phases: *Alignment Learning* and *Selection Learning*. In the alignment learning phase it uses the edit-distance algorithm to align two sentences such that identical fragments in the sentence pair are identified. Given the training corpus of n sentences, the ABL compares every sentence pair (n^2 of them) and identifies all possible constituents. Every time a new constituent is found, it introduces a new non-terminal and also merges the evidence for a constituent that has been seen earlier thereby avoiding the proliferation of non-terminals.

(3) $(_1$ Book Delta 128 $)_1$ <u>from Dallas to Boston</u>

(4) $(_1$ <u>Give me</u> $(_2$ all flights $)_1$ <u>from Dallas to Boston</u> $)_2$

(5) <u>Give me</u> $(_2$ help on classes $)_2$

However, the alignment learning phase might produce overlapping alignments in a sentence by comparing it with two different sentences. For example consider the sentence (4), which is being compared with (3) and (5). The underlines represent the alignment between the sentence pairs and the parentheses indicate the constituents indexed to show the constituent correspondences. The selection learning phase attempts to choose the correct constituent from such overlapping constituents. van Zaanen describes five methods for selection, two of which are important. viz. *leaf* and *branch*. These are probabilistic methods and compute the probabilities of overlapping constituents to choose the right one. The leaf method gets the probability of a constituent α by normalizing its count to the total number of constituents $\mathcal{C}$ as in:

$$P_{leaf}(\alpha) = \frac{|\alpha' \in \mathcal{C} : yield(\alpha') = yield(\alpha)|}{|\mathcal{C}|} \tag{26}$$

The branch method additionally uses the non-terminal information for better normalization and the expression is given by:

$$P_{branch}\big(\alpha | root(\alpha) = X(\alpha)\big) = \frac{|\alpha' \in \mathcal{C} : yield(\alpha') = yield(\alpha) \wedge root(\alpha') = X(\alpha)|}{|\alpha'' \in \mathcal{C} : root(\alpha'') = X(\alpha)|} \tag{27}$$

The EMILE system (Adriaans et al., 2000) is similar to ABL in the sense that it exploits the shared contexts to group the constituents into clusters. Employing the notion of substitution classes, the idea is to iteratively find new constituent sequences from contexts and vice versa starting from a small set of bootstrapping examples. Given a large enough corpus in a very narrow domain (such as the ATIS corpus of air traffic reservation queries), one can expect to find classes of constituents appearing in similar contexts. It has been shown to work for shallow, context-free languages with characteristic contexts and expressions provided that the sample is drawn based on a simple distribution.

Initially, EMILE uses a *clustering* phase to generate clusters of contexts and expressions using some random seeds provided manually. Each expression cluster is then assigned a group-label, which can be used to rewrite sentences by substituting the expressions with the label, resulting in proto-rules. For example, Sentence (1) can result in: *[0] ⇒ What is [19]* and *[19] ⇒ a family fare*. In the subsequent *rule induction* phase, it generates recursive rules by substituting labels for smaller expressions in longer expressions. For example, using the rule *[19] ⇒ a family fare* and a longer expression *[201] ⇒ the price of a family fare to Rome*, it can create a new rule *[201] ⇒ the price of [19] to Rome*.

However, the biggest drawback of both ABL and EMILE lies in their strong assumptions about the presence of *typical* constituents and surrounding context. Naturally, while ABL and EMILE work reasonably well on the ATIS corpus pertaining to a narrow domain, they may not generalize in other corpus such as WSJ, whose sentences are more complex in a slightly broader domain.

3.2 Distributional Clustering Approaches

Clark (2000) presented Context Distribution Clustering (CDC) algorithm that induces clusters of tag sequences based on the context distributional information, which are then filtered using a Mutual Information (MI) criterion between the left and right contexts of a sequence to remove destituent (non-constituent) clusters. The algorithm is incorporated in the well-known Minimum Description Length (MDL) framework, whereby it chooses the clusters so that the resulting constituents have the shortest description length. It starts by clustering the most frequent (> 5000) tag sequences in the BNC corpus using the k-means clustering with the value of k set to 100. While, several clusters clearly correspond to syntactic constituents, this method also produces spurious clusters where the tag sequences are destituents.

Based on the traditional tests for constituency, a valid constituent is known to occur in the different contexts unlike a destituent sequence. For example, a noun phrase sequence DT NN or NN can appear as a subject starting a sentence, or as an object in the end of a sentence. In the subject position, it will typically be followed by a finite verb as in (6). On the other hand, it normally appears after a finite verb in the object position as in the example (7) and is likely followed by the sentence end marker or a prepositional phrase. In contrast, a frequent destituent sequence like IN DT (*in the*) is always followed by a noun, as shown in the same sentence.

(6) *Susan*/NN made/VBD a/DT presentation/NN to/TO the/DT committee/NN ./PUNC

(7) We/NN put/VBD *a*/DT *man*/NN in/IN the/DT moon/NN ./PUNC

This suggests a strong correlation (and thus high MI) between the contexts occurring on either side of a constituent, whereas no such correlation exists for a destituent sequence. Clark exploits this to identify the destituent clusters by measuring the MI between the left and right contexts[2] and uses a threshold - that accounts for the natural MI existing between the neighbourhood words; to determine the constituency.

The algorithm starts from a maximum likelihood grammar with a single non-terminal and individual rules for each sentence type and iteratively clusters the tag sequences and filters them according to the MI criterion. At each iteration it selects a cluster greedily, such that this results in best reduction in description length. A new non-terminal is added to the cluster if it does not have a singleton non-terminal member already; other cluster members are rewritten as rules that expand the single non-terminal. The next iteration starts from the current grammar and might create a new cluster or new rules to be merged with an existing cluster. The MDL gain in each iteration is shown to be related to the mutual information of set of rules that are found in that iteration.

In contrast to the ML and MDL objective functions, Klein and Manning (2001a) uses two linguistic criteria for constituency, viz. *External distribution* and *Substitutability* as the basis for grammar induction. While *substitutability* was defined earlier, according to *external distribution* a valid constituent (sequence of words in a given sentence) will appear in a variety of structural positions within larger constituents unlike the destituents that are restricted to limited contexts.

Klein and Manning formalize these two linguistic criteria by a distributional notion of context. Given a parts-of-speech tag sequence α occurring in a context (say) $x\alpha y$, where x and y are the adjacent tags or sentence boundaries. The distribution over contexts in which α occurs is called its *signature* denoted by $\sigma(\alpha)$. They present two systems: *Greedy-Merge* and *Constituency-Parser* that considers the grammar induction problem as a clustering problem, where (distributionally) similar tag sequences are grouped in the same cluster, possibly represented by a unique non-terminal.

Greedy-merge learns symbolic CFGs for partial parsing aimed more for constituent precision than on recall. It uses agglomerative clustering, where the sequences are clustered iteratively with a stopping criterion based on heuristics (same category appears in several merges) or by monitoring for the drop in parsing accuracy using a small test set. It compares pairs of sequences α and β using the normalized divergence metric in Equation (28) and merges the pair with least divergence. It also assigns a new non-terminal category for each new merge.

$$d(\alpha, \beta) = \frac{D_{JS}(\sigma(\alpha), \sigma(\beta))}{H_s(\sigma(\alpha)) + H_s(\sigma(\beta))} \tag{28}$$

[2]Sentence boundary marks the left or right context for constituents appearing in the beginning or end of a sentence respectively.

where $H_s(\sigma(\alpha))$ is the scaled entropy of the signature $\sigma(\alpha)$ for sequence α defined by Equation (29).

$$H_s(\sigma(\alpha)) = H(\sigma(\alpha))\Big(H(\sigma_u(\alpha))/H(u)\Big) \tag{29}$$

After each merge, the sentences are partially parsed by the current grammar, attaching all the parentless nodes to a pseudo ROOT node. The clustering is now repeated on the sequences that are ordered sets of adjacent sister nodes in the parse. While merging a sequence and a single non-terminal, it creates a new rule rewriting the single non-terminal as the sequence similar to CDC. The merging of two non-terminals results in the collapse of two clusters into one and is followed by the reanalysis of the rule RHS. The current grammar state is used to parse a held out test set to track improvement in parser performance and the algorithm is stopped when the test set accuracy shows a drop.

While the idea of greedy merge has similarities with CDC (Clark, 2000) in terms of approach and high precision-low recall, there are significant differences as well. Clark starts with a maximum likelihood grammar with one rule for each sentence type and iteratively finds points for breaking sentences. Greedy merge, contrary to this uses agglomerative clustering in a bottom-up fashion to cluster the sequences. Secondly, the MI criterion of CDC is similar to the one proposed by Magerman and Marcu (1990) in that it helps to find the right points for segmenting the sentence capturing long distance dependencies, whereas the greedy-merge is plainly motivated by contextual similarities between sequences as captured by the diverge measure in equation (28).

The constituency-parser method (Klein and Manning, 2001a) is interested in identifying the constituent sequences in a given corpus. It other words, it is a clustering system that separates constituent sequences from destituents. Considering a sentence as a collection of (sequence, context) pairs (α, c), it assumes that such pairs are generated independently by a model shown in Equation (30). It first generates a binary judgement j_α of whether a sequence α is a constituent or not and then generates the sequence α followed by its context c given the judgement[3]. The joint probability of (α, c) pair tokens for the entire parse tree T can be written as.

$$P(\alpha, c) = \sum_{j_\alpha \in T} P(j_\alpha)P(\alpha|j_\alpha)P(c|j_\alpha) \tag{30}$$

It uses EM to maximize the likelihood of these pair tokens given the latent judgements j_α, constrained by the requirement that the pairs from a given sentence must form a valid binary parse

[3]This is not a generative model in strict sense, since the (sequence, context) pair tokens are considered independent of each other. As authors point out, this can be thought of like a random field over possible parses with the sequences and their contexts used as features.

(avoiding mutually incompatible parses). It uses four different methods to initialize the bracketings for EM: Random (using random initial parses), Entropy ($P(j_\alpha|\alpha)$ is weighted proportionally to $H_s(\sigma(\alpha))$), Right-branch (by forcing right-branching structures) and Greedy (using the grammar output of Greedy-merge). It finds the best tree in the E-step by scoring it with the product of the likelihood of its hidden judgements as in:

$$\underset{\mathcal{T}}{\operatorname{argmax}} \prod_{(\alpha,c)\in s} P\Big(j_\alpha(\alpha,\mathcal{T})|\alpha,c\Big)$$

The probability of a pair token (α, c) appearing in $\mathcal{T}$ being a constituent is computed by Equation (31). For the subsequent re-estimation steps, it again uses the four methods mentioned above for deciding the binary bracketing. In the M-step, it fixes the best trees and finds the parameters that maximize the likelihood of the (α, c) pair tokens given the latent judgements j_α, using a gradient ascent method.

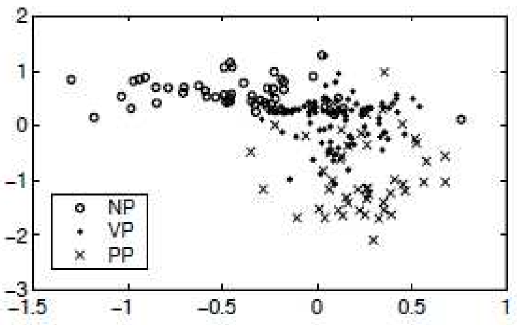
$$P(j_\alpha|\alpha,c) = P(j_\alpha|\alpha)P(j_\alpha|c)/P(j_\alpha) \tag{31}$$

As mentioned earlier, the main purpose of this method is clustering of constituent and destituent sequences so that it does not produce CFG-style rules (probabilistic or otherwise). The resulting constituents have high recall, because the model is forced to select a bracketing scheme so as to produce a full binary parse of the sentence. The complete evaluation of this method along with other approaches is presented in Section 4.

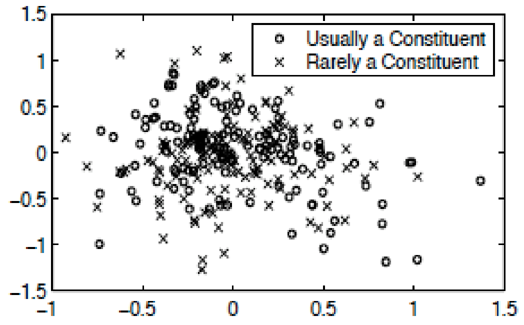
3.3 Constituent Context Approaches

Earlier models of clustering approaches for grammar induction attempted to solve the two problems of identifying the constituents and labelling them simultaneously using the distributional analysis of sequences and their contexts. Consider the Figure (6a), where the points correspond to the 50 most frequent constituent sequences of three types shown in the vector space of its contexts, projected onto their first two principal components. Clearly, the three types of constituents can be separated reasonably well by clustering. On the other hand Figure (6b) shows 150 sequences identified as constituents with probability more than 0.5 and another 150 sequences that are not, again projected onto the two principal components. This shows that the problem of separating constituents from destituents is not easily amenable to clustering (Klein and Manning, 2001b).

The Constituent Context Model (CCM) proposes a simpler probability model over trees to induce bracketing (Klein and Manning, 2001b) using a simple observation that valid constituents in a tree should be non-crossing. It uses two features defined for any parse tree t of a sentence q: $f_\alpha(t)$



(a) Three types of Constituents



(b) Constituents and Destituents

Figure 6: Frequent sequences plot illustrating the amenability of constituent labelling and constituent identity to clustering - Figure from Klein and Manning (2001b)

over the terminal sequences and $f_c(t)$ over the contexts of a sequence α. The parse tree t of q is thus viewed as a collection of sequences and contexts for every node in t and the empty spans between two terminals. It uses a conditional exponential model defined in Equation (32).

$$P(t|q,\Theta) = \frac{exp\Big(\sum_{(\alpha,c)\in t} \lambda_\alpha f_\alpha + \lambda_c f_c\Big)}{\sum_{t:yield(t)=q} exp\Big(\sum_{(\alpha,c)\in t} \lambda_\alpha f_\alpha + \lambda_c f_c\Big)} \tag{32}$$

where, λ_α and λ_c are the weights corresponding to the model features. The sequence feature f_α counts the frequency of sequence α being a constituent in a tree t, just by itself. The contextual feature f_c counts the frequency of a terminal symbol c used as a context in a tree.

The algorithm is an EM-style iterative that maximizes the conditional probability of trees over the entire corpus C and is given by: $P(\mathcal{T}|C,\Theta) = \prod_{t\in\mathcal{T},yield(t)=q} P(t|q,\Theta)$. It is initialized by λ being zero (with ties broken at random) and arbitrary set of trees[4] for $\mathcal{T}$ so that all parse trees have uniform probability.

The algorithm first finds the set of best (high probability) tree structure $t*$ for each sentence with a fixed Θ. It then fixes the trees and estimates a new set of Θ^* that maximizes $P(\mathcal{T}|C,\Theta)$ using for example, a conjugate gradient (CG) ascent method. Note that in both steps, the conditional likelihood of trees $P(\mathcal{T}|C,\Theta)$ is greater than its value in the corresponding previous step

[4]Considering arbitrary sets of trees without any constraints on the constituent sizes will lead to exponential number of trees. In order to keep it at a manageable level it assumes a binary branching structure that results in significant reduction of trees.

assuring non-decreasing conditional likelihood in every iteration until convergence. Similar to the constituency-parser method, CCM also attempts to identify the constituents and does not group them into different types. Consequently, it does not produce CFG-style rules for full parsing, but enables shallow parsing using identified constituents.

The generative version of CCM (Klein and Manning, 2002) combines the notion of distributional clustering in a generative framework thus making it possible for the system to apply a constituent sequence learnt from a given context to learn a different context and vice versa. While CDC and greedy-merge approaches included distributional notion, the constituency-parser and CCM lacked this. The generative CCM first generates a bracketing B, using a bracketing distribution $P(B)$ and then generates a sentence q for the given B as in Equation (33).

$$P(q) = P(B)P(q|B) \tag{33}$$

Thus given a sentence, it finds several possible bracketings for it, such that there are no crossing brackets in a particular bracketing scheme. After generating the possible non-crossing bracketings B, it makes two simple assumptions for $P(q|B)$: i) the spans in a given B can be filled independently and ii) the sequence for a given span and its context are independent of each other[5].

$$\begin{aligned} P(q|B) &= \prod_{\langle i,j\rangle \in spans(q)} P(\alpha_{ij}, c_{ij}|B_{ij}) \\ &= \prod_{\langle i,j\rangle} P(\alpha_{ij}|B_{ij})P(c_{ij}|B_{ij}) \end{aligned}$$

$P(\alpha_{ij}|B_{ij})$ is a pair of multinomial distributions for both constituents and destituents over the set of possible sequences and $P(c_{ij}|B_{ij})$ is the same for all possible contexts. Given this model, the generative CCM employs EM algorithm to induce structure by alternately computing the conditional bracketing likelihood $P(B|q,\Theta)$ and then finding the parameters Θ' by maximizing the likelihood of sentences, i.e $\sum_B P(B|q,\Theta)P(q,B|\Theta')$. Unlike the conventional EM procedure, here the grammar induction starts from the M step with a initial bracketing distribution $P_{split}(B)$ obtained by recursively choosing a random point and then proceeding on either side to split a sentence. This soft EM procedure has been shown to have better convergence and robustness, and is not sensitive to initialization or smoothing parameters.

[5]While these assumptions make the model simpler, it also makes it deficient either due to conflicting span indices (i and j) or due to incorrect sequence length and consequently the generated sequences and contexts may not tile to a valid sentence. This is addressed by a renormalization step (see Klein and Manning (2002)).

The generative CCM has also been extended to handle multiple constituent classes for inducing the constituent labels apart from constituent identification and it employs a complex model of generating bracketing, labels, and finally the constituents and their contexts. It uses a total of 12 constituent types including destituents and is shown to be effective for some but not all constituent types, though with a minor drop of 0.2% in overall precision and recall compared to the two-class model of identifying constituents and destituents.

Klein and Manning (2002) also show that generative CCM can also be used with automatically induced POS tags instead of the gold tags. Generative CCM uses the baseline system by Schütze (1995) which employs a weighted k-means algorithm to induce 200 word-classes clusters using the distributional information of the word contexts.

The generative CCM in Equation (33) has been shown to have higher accuracy in identifying constituents (from destituents), but does not label them. On the other hand, the prototype-driven model of Equation (21) (see Section 2.1) based on distributional similarity is good at labelling them. Haghighi and Klein (2006) integrated these two complementary models and showed it to produce a better phrase-structure grammar. The combined model runs the EM on the two approaches separately, and computes the posteriors over their hidden variables while maximizing the same objective function given in Equation (34).

$$P(q|\Theta_{ccm}, \Theta_{proto}) = \sum_{B \in \mathcal{B}(q)} P_{ccm}(B, q|\Theta_{ccm}) \sum_{T \in \mathcal{T}(B): yield(T)=q} P_{proto}(T, q|\Theta_{proto}) \tag{34}$$

The CC model first generates unlabelled binary bracketing $\mathcal{B}$ for a sentence and then the prototype model generates labels for the nodes in the tree T that is compatible with some $B \in \mathcal{B}$. This approach is similar to the one by Periera and Schabes (Pereira and Schabes, 1992), except that Klein and Manning use the bracketing generated by $P_{ccm}(\cdot)$ instead of the manual bracketing. This ensures that the probability mass is assigned to the rules that are compatible with the hidden bracketing $\mathcal{B}$, and that the grammar G is of better quality compared to the CCM. As noted above, the advantage of this approach is that it provides an easier handle for controlling the resulting PCFG, through the seed examples for different constituent types.

3.4 Inducing Latent Tree Substitution Grammar

In the case of PCFG induction, the parse tree is derived from its constituent rules that are productions of the CFG. Now, instead of using the productions, one could use a Tree Substitution Grammar (TSG), where elementary units of the derivation are the arbitrary sized subtrees. The elementary trees can then be combined at the (frontier) non-terminal nodes in the derivation process. The objective in such an approach is to induce the latent TSG. This is commonly known in the literature as Data-oriented Parsing (DOP) (Bod, 2006). Though, this approach is similar to PCFG induction using EM, it has been included here with other structural-search based methods because the TSG rules are not known before the model is trained completely.

Unlike CFG whose productions are restricted to one level rewriting, the productions of the DOP considers all possible subtrees in a parse tree and therefore can have arbitrary depth. DOP is thought to model the language better as it includes concrete language examples in its rules and not just abstract rules. As a probabilistic model, the DOP productions are assigned probabilities and the probability of a derivation is obtained by summing over the (log) probabilities of the involved productions. Due to the arbitrary depth of the TSG rules, the number of productions is combinatorially large, therefore the parameterization is difficult. However, the DOP framework uses random sampling to overcome this issue.

The DOP can be used in an unsupervised setting by considering all possible binary parses and using their subtrees to train a probabilistic model. In fact two versions of unsupervised DOP model using a relative frequency based and maximum likelihood (ML) training has been proposed and the ML-based approach is shown to perform better (Bod, 2006). The unsupervised ML-DOP starts by generating all possible (unlabelled) binary trees for the sentences in the training corpus and then randomly sampling a large subset of binary trees. It then extracts a fixed number of subtrees, which is then trained iteratively with EM and using a re-estimation procedure similar to inside-outside algorithm. The overfitting can be avoided by *cross-training* procedure - dividing the training set into two parts and training the subtrees extracted from one part on the other and vice versa. However, the random sampling of subtrees might render the replication of this approach to get the same performance difficult.

3.5 Dependency Models

The example dependency structure given earlier in Section 1.1 is repeated in Figure (7) to illustrate the dependency structure. As noted earlier, a projective dependency tree captures the head-dependent relationship among the words in a sentence (ex. the auxiliary verb *will* has *rat*, *eat*, and . as dependents; determiner *this* is the dependent of the head noun *winter*) such that no dependency edges in the tree cross each other.

Similar to the case of PCFG that assigns a score to a parse tree by aggregating the probability

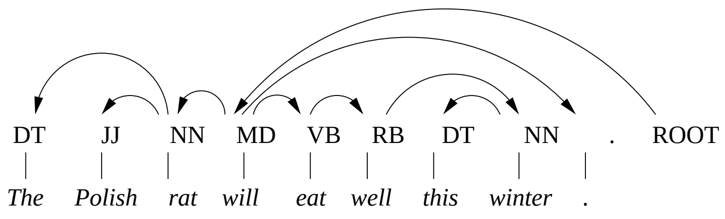


Figure 7: Dependency Structure

of the rules involved in the derivation, the dependency models treat the dependency structure D as a set of ordered (*head*, *argument*) pairs denoted as (h, a), and score the entire structure as the product of the probabilities of the individual dependencies. Optionally, the direction of the dependency and the number of arguments for the head (valence) are also included in the model as separate features.

Earlier work on unsupervised dependency induction used a generative model (Paskin, 2001) that first generates a plain dependency graph $\mathcal{G}$ having empty nodes connected by directed arcs, uniformly at random and then generates the words in the sentence q starting from a pseudo ROOT node to complete an entire dependency structure D.

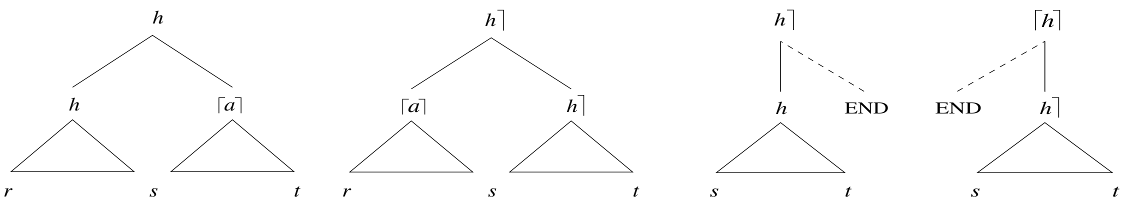
$$\begin{aligned} P(D) &= P(\mathcal{G})P(q|\mathcal{G}) \\ &= P(\mathcal{G}) \prod_{(s,t,dir)\in\mathcal{G}} P(q(s)|q(t), dir) \end{aligned} \tag{35}$$

It exploits the property of dependency structures that, every word in a sentence has *exactly* one head and links a pair of *grammatical bigrams* such that this property is satisfied. The term $P(a|h, dir)$ is a multinomial distribution for generating an argument conditioned on the head and the direction (specifying whether the argument occurs on right or left of the head). The parameters of $P(a|h, dir)$ are determined by running EM over the raw data. However, the model is deficient because it captures more of the semantical relatedness between different words rather than the syntactic dependencies and hence links two words that have high mutual information. This deficiency occurs because, the model in Equation (35), first fixes the structure instead of letting the structure to evolve on the basis of the heads and their valencies (number of arguments they can take).

Head outward process (Eisner, 1996; Collins, 1999) used in supervised dependency models captures this linguistic valency where each head takes a specific number of arguments. For example, transitive verbs such as *paint*, *touch* take two arguments (a subject and an object), while some other verbs (*ask*, *give*) has a valency of 3 and take both direct and indirect objects in addition to a subject. The supervised approaches by Eisner and Collins include such valency information along with the identity of a head and the direction in generating its set of arguments. Specifically, the valency is factored in the model by including a END process, which is generated for each side of a head after its arguments are generated.

The Dependency Model with Valence (DMV) is a generative model that combines the graph generation with that of the terminal symbols (Klein and Manning, 2004), while including the head-outward and END processes. For each head, it generates the dependents on right and left sides separately in the same order, and in each direction the generation is continued until a final END argument is generated for that side indicating that no further arguments need to be generated in that side. A single dependent for ROOT is chosen uniformly over all the words in the sentence.

The DMV model can be seen as dealing with four types of distinct lexicalized tree structures as shown in Figure (8). Starting from the head h, it iteratively generates the dependents on the right-side Figure (8a) until an END argument is generated as in Figure (8c). Then, it repeats the process on the left side to generate the configurations in Figures (8b) and (8d). The right and left ceiling ($\rceil$ and $\lceil$) in a node signifies that END argument has been generated for the side.



(a) Right Dependent (b) Left Dependent (c) Right END (d) Left END

Figure 8: Dependency Structures handled by the DMV model - Figure from Klein and Manning (2004)

At every step, the model first decides if an END is to be generated for the head in the current direction given its binary *adjacency*, defining whether or not an argument has already been generated for it; $P(\text{END}|h, dir, adj)$. If this binary END decision turns out to be false, the model generates an argument for the head in the current direction $P(a|h, dir)$; the adjacency is not considered in generating the argument itself. Klein and Manning (2004) use the word-classes and not the actual words as the terminal symbols. The probability of a fragment rooted at a head h in the D is given by Equation (36):

$$P(D(h)) = \prod_{dir \in \{r,l\}} \Big(\prod_{a \in A_D(h,dir)} P(\neg\text{END}|h, dir, adj) P(a|h, dir) P(D(a)) \Big) P(\text{END}|h, dir, adj) \tag{36}$$

where, $A_D(h, r)$ refers to the right-side r dependents (arguments) of a head h in D.

The parameters of END ($P(\text{END}|\cdot)$) and choosing argument $P(a|\cdot)$ are re-estimated by the inside-outside algorithm (as explained in Section 2) and uses the expected frequencies of a node labelled h in the span (s, t) in the parses of observed sequence o ($c_o(h : s, t)$). The probability of the entire tree T can then be written as Equation (37), with the root node q_r chosen uniformly from all the words in q.

$$p(T) = p_{\text{ROOT}}(q_r) \cdot P(D(r)) \tag{37}$$

In their approach, Klein and Manning, begin with the M-step of EM and use a new initialization scheme called *harmonic initializer* that gives an initial guess of the posterior distributions. Given a sentence q of length n words, i) it chooses a root node q_r with uniform probability $1/n$ and ii) each non-ROOT node takes the same number of arguments, with each node at some position i taking other nodes at position j as arguments, in inverse proportion to the distance $|i - j|$ between them plus a constant. These constraints for the initializer can be summarized by the Equations (38) to (40).

$$P_{\text{ROOT}}(q_i) = \frac{1}{n} \tag{38}$$

$$P(q_i|\text{ROOT}, r) = 0 \tag{39}$$

$$P(q_j|q_i, dir) = \frac{1}{c + |i - j|};\ q_i \neq q_j \neq \text{ROOT} \tag{40}$$

The other unspecified cases can be assumed to be either uniform or straightforward as suggested by Spitkovsky et al. (2010a) and are shown by Equations (41) to (43).

$$P(\text{END}) = \frac{1}{2} \tag{41}$$

$$P(\text{END}|\text{ROOT}, l, 1) = 0 \tag{42}$$

$$P(\neg\text{END}|\text{ROOT}, l, 0) = P(\text{END}|\text{ROOT}, r, 0) = 1 \tag{43}$$

This initializer ensures that the model is not locked in a local optimum as it happens in the EM with random initialization. The DMV model has been evaluated for the accuracy in directed and undirected dependency attachments and has been shown to perform reasonably well for three different languages, namely English, German, and Chinese as later covered in the Section 4. It was further combined with the CCM so that the resulting joint model can extract both constituency and dependency structures.

The combined model uses the four structures illustrated in Figure (8), each of which are scored by the product of the probabilities from individual models. For CCM, the configurations are viewed as a sequence of terminal symbols appearing in its context. The inside-outside algorithm is run over this product model and the parameters of the two models are separately re-estimated using the overall counts obtained from the inside-outside procedure. Interestingly, the combined model scores better in both constituency and dependency evaluation than the standalone models for English, German and Chinese. It has to be noted however that, both DMV and the combined model, produce constituent sequences and dependency attachments, but not a probabilistic grammar.

A typical approach in learning is to change the training regime by bootstrapping with a simpler subset and adding more data in stages, iteratively increasing the learning complexity, which mimics the idea of gradual learning. Employing this idea in combination with the DMV approach Spitkovsky et. al (2010a), propose three different algorithms: *Babysteps*, *Less is More* and *Leapfrog*. Babysteps is similar to Carroll and Charniak (1992) in that it starts from a simpler training set consisting of single-word sentences in WSJ and iteratively adds more complex[6] sentences from WSJk ($k = 1 \dots 45$) to the training. At each iteration the algorithm is initialized by the model learnt from the previous iteration. Less is More is similar to the original DMV and uses a fixed, low-complexity subset of the training data. The third variant, Leapfrog, combines the first two approaches by making smaller increments initially up to $WSJ15$ and then jumping large steps of $WSJ15, 30, 45$.

The inference method used offers yet another handle for experimentation. Majority of the approaches above use the classic EM with inside-outside re-estimation, which is an exact inference method. The inside-outside algorithm in EM can be replaced by a computationally simpler alternative of best parse, resulting in Viterbi-EM (Spitkovsky et al., 2010b). Despite being an approximate inference method, Viterbi-EM gains significant accuracy improvements as it uses the same inference method for both training and evaluation.

The DMV model of probabilistic grammar was subsequently used in several other works. Cohen et al. (2008) used the DMV in a Bayesian setting through a variational EM algorithm. This model extends the *correlated topic model* (CTM) (Blei and Lafferty, 2006) originally proposed for topic modelling in documents for grammar induction. The CTM employs a logistic normal prior instead of the Dirichlet prior used by the *Latent Dirichlet Allocation* (LDA) model (Blei et al., 2003) as the former is better in capturing the correlations between the topics by relaxing the strong independence assumed between the different topics by the latter. This can be illustrated by comparing the process involved in generating the multinomial θ in Dirichlet logistic-normal distributions.

The process for generating a sentence s and its derivation tree t using a hierarchical generative

[6]The complexity of the sentence is approximated by its length.

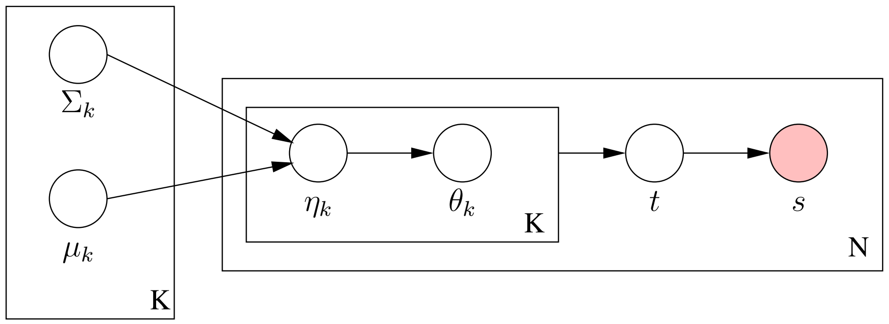


Figure 9: Graphical Model for Probabilistic Grammar with Logistic-normal prior - Figure from Cohen et al. (2008)

model with a logistic-normal distribution shown in Figure (9) is as follows:

- Draw a logistic normal distribution $\eta \sim \mathcal{N}(\mu_k, \Sigma_k)$ for $k = 1, \ldots, K$
- Map to simplex $\theta_{k,i} = \exp(\eta_{k,i}) / \sum_{i'=1}^{N_k} \exp(\eta_{k,i'})$ for $k = 1, \ldots, K$ and $i = 1, \ldots, N_k$
- Sample from θ to generate (s, t)

The covariance matrix Σ of the Gaussian $\mathcal{N}$ can be used for specifying such prior information; this allows the CTM to assume some correlation between the topics, incorporated through a logistic normal distribution of topics. In contrast, for the case of a Dirichlet distribution, the correlation among different multinomials in θ (for example between θ_i and θ_j for $i \neq j$) cannot be captured directly, but only through the normalization step in the Dirichlet generation process given below.

- Draw $\eta_j \sim \Gamma(\alpha_j, 1)$ independently for $j \in 1, \ldots, d$
- $\theta_j = \eta_j / \sum_i \eta_i$ for $j \in 1, \ldots, d$

In the context of the grammar induction, Cohen et al. (2008) analogously propose to use the correlations between similar POS tags to hypothesize the argument types taken by a specific word after observing a similar tag as its argument. For example, having observed a singular noun (NN) as an argument for a verb (VBD) with some probability p', the CTM can assume the probability of having a plural noun (NNS) as an argument for verb (VBD) to be correlated to p', thereby improving the induction of dependency structures.

For inference, Cohen et al. give a Variational Bayesian algorithm that approximates a posterior function $q(\eta, t)$ such that it maximizes the lower bound of the likelihood $p(s, t|\mu, \Sigma)$ given by the Jensen's inequality as in Equation (44).

$$\log p(s,t|\mu,\Sigma) \geq \sum_{k=1}^{K} \mathbb{E}_q\big(\log p(\eta_k|\mu_k,\Sigma_k)\big) + \mathbb{E}_q\big(\log p(s,t|\eta)\big) + H(q) \tag{44}$$

Using a mean-field assumption factorizes the posterior $q(\eta, t)$ as

$$q(\eta,t) = q(\eta)q(t) \tag{45}$$

Now, simplifying the likelihood by using the factorized posterior and by introducing K new variational parameters $\tilde{\zeta}_k$ yields the second term of Equation (44) as:

$$\mathbb{E}_q\big(\log p(s,t|\eta)\big) = \sum_{k=1}^{K}\sum_{i=1}^{N_k} \tilde{f}_{k,i}\left(\tilde{\mu}_{k,i} - \log\tilde{\zeta}_k + 1 - \frac{1}{\tilde{\zeta}_k}\sum_{j=1}^{N_k}\exp\left(\tilde{\mu}_{k,j} + \frac{\tilde{\sigma}^2_{k,j}}{2}\right)\right)$$

Employing a short-hand notation of $\tilde{\psi}_{k,i}$ for the complex expression involving $\tilde{\mu}$, $\tilde{\sigma}$ and $\tilde{\zeta}$ in the second term expression, the lower bound of Equation (44) can be written as Equation (46).

$$\log p(s,t|\mu,\Sigma) \geq \sum_{k=1}^{K} \mathbb{E}_q\big(\log p(\eta_k|\mu_k,\Sigma_k)\big) + \sum_{k=1}^{K}\sum_{i=1}^{N_k} \tilde{f}_{k,i}\tilde{\psi}_{k,i} + H(q) \tag{46}$$

Now, the grammar parameters θ can be estimated with the EM algorithm, where the E-step maximizes the lower bound with respect to the variational parameters $\tilde{\mu}$, $\tilde{\sigma}$, $\tilde{\zeta}$ and $\tilde{f}$ using co-ordinate ascent. Each of these parameters are optimized separately by different optimization techniques: $\tilde{\mu}$ through conjugate gradient, $\tilde{\sigma}$ using Newton's method and $\tilde{\zeta}$ by a closed form solution involving $\tilde{\mu}$ and $\tilde{\sigma}$. $\tilde{f}$ are the counts of the individual dependencies observed in the derivations of s and are estimate through dynamic programming. The M-step estimates the values of μ and Σ using a closed-form solution and the parameters estimated in E-step. The process is repeated until convergence of the likelihood $p(q|\mu,\Sigma)$ of held-out data.

The LN prior allows the probabilities in a single multinomial to covary; thus it correlates the probabilities of a plural noun (NNS) and a singular noun (NN) separately being an argument of a head verb (VBD). The LN prior is realized in practice by combining the singular and plural nouns into a coarse group and allowing the probabilities within this group to covary. However, this could be further generalized to enable covariance across multinomials throughout the model, thus permitting correlations between the probabilities of different heads (such as VBD, VBZ and VBN) independently taking NN as an argument. In other words, instead of only letting the probabilities of arguments from the same group to covary, a generalized version can allow covariance of the probabilities of both heads and arguments independently in their respective coarse groups. This resulted in a shared LN prior model (Cohen and Smith, 2009a), which again used Variational Bayes for inference algorithm explained above.

Another alternative to inject prior information in the Variational Bayes framework is to use a mixture model for posterior distribution, where the mixture components are linguistically motivated (Cohen and Smith, 2009b). Unlike the mean-field assumption (Equation (45)) used in logistic-normal prior model that assumes a factorized approximate posterior, mixture models employ relaxed restriction on the posterior by introducing a family of mixture distributions $\Delta_r = \{\langle \lambda_1 \dots \lambda_r \rangle \in \mathbb{R}^r : \lambda_i \geq 0, \sum_i \lambda_i = 1\}$. The posterior can now be written as a linear combination of the mixture components.

$$q(\eta, t|\lambda) = \sum_{i=1}^{r} \lambda_i q_i(\eta) q_i(t) \tag{47}$$

The variational EM can now be modified to maximize the lower bound of the likelihood with respect to $q(\eta)$, $q(t)$ and λ. Cohen and Smith (2009b) achieve this by independently optimizing the bound to update $q_i(\eta)$ and $q_i(t)$ (where, $i = 1, \dots, r$), respectively by running the E and M steps for r times and finally optimizing for mixture coefficients λ in a separate C step. In order to avoid a stronger bias towards a particular mixture component, λ is additionally constrained within Δ_r initially, which are then gradually relaxed using the idea of annealing in the later iterations of EM so as to avoid local maxima. Using the mixture components helps to pull the attachment counts during the E-step, towards the prior expectations as encoded by the mixture beliefs. Cohen and Smith (2009b) experiment with different linguistically motivated mixture components RIGHTATTACH (RA - each word is attached its parent in the right), VERBASROOT (VAR - only verbs can be the root of the sentence), SHORTDEP (SD - allowing only shorter dependencies) and so on.

The DMV models explained thus far are simple ones and do not use sophisticated features employed by the supervised approaches. The valence frame information and lexicalization are two of the important linguistic features that can enrich the DMV as proposed in Extended Valence Grammar (EVG) and Lexicalized-EVG (L-EVG) (Headden-III et al., 2009). The DMV uses the valence information only for determining the number of arguments for a head. The EVG introduces the notion of valence frames by which it allows different distributions over the arguments of a head at different valence slots. This difference in the distribution makes it possible to distinguish between the different roles played by the arguments; for example give a sequence DT JJ NN as in *A green apple*, the dependency attachments NN-JJ and NN-DT are distinguished from each other unlike the case of DMV, which treats them to be equivalent.

In addition to the valence frames, Headden et al. (2009) incorporates lexical information which adds to the sophistication at the cost of introducing data sparsity issues. The lexical information is incorporated for both head and argument resulting in lexicalized EVG (L-EVG). The argument POS tag is now conditioned on the head POS tag (h), head word (h_w), direction (dir) and valence slot (v). The argument word (a_w) is conditioned only on the argument POS tag (a). A third key difference between this and the earlier DMV-based approaches is that, Headden et al. uses the split-head algorithm for converting the DMV into a CFG representation by splitting the heads for its left and right derivations. Consequently, EVG and L-EVG produces a PCFG style productions unlike the other DMV variants that are characterized by probabilistic models for generating arguments and END on either directions.

However the sophistication of the model leads to increased sparsity and Headden et al. resort to smoothing to address the issue. For the EVG, the smoothing is achieved either by dropping the valence information and backing off to the original DMV model, or by dropping the head POS tag. Headden et al. show smoothing with dropped head to perform better than dropped valence frame. In the case of L-EVG, the unlexicalized EVG with dropped head is used as the backoff distribution. In order to facilitate smoothing in EVG and L-EVG, Headden et al. use a modified class of PCFGs called tied PCFGs that gives the same probability for analogous rules rewriting two different non-terminals. This equal probability gives rise to equivalence classes over rules ($\mathcal{R}$) and non-terminals ($\mathcal{N}$) that are tied. They use a Dirichlet prior over the tied PCFGs and combine it with linear interpolation for smoothing. In order to effectively use the Dirichlet prior they employ Variational Bayes technique to estimate a distribution over θ, instead of a point estimate directly returned by EM.

Using the idea of Variational Bayesian learning of PCFG (Kurihara and Sato, 2004), Headden et al. add the prior information through the Dirichlet distribution having hyperparameters α over tied PCFGs. Following the VB, they maximize the approximate posterior distribution $Q(D, \theta)$ over dependency trees D and parameters θ, which maximizes the lower bound $\mathcal{F}$ of the marginal likelihood $\log P(q|\alpha)$, which has an intractable integral as given in Equation (48).

$$\log P(q|\alpha) \geq \sum_{D \in \mathcal{D}(q)} \int_{\theta} Q(D, \theta) \log \frac{P(q, D, \theta|\alpha)}{Q(D, \theta)} \tag{48}$$

The approximate posterior distribution is further simplified through an independence assumption and is given in Equation (49), where $\bar{N}$ is a non-terminal equivalence class. $Q(D)$ and $Q(\theta)$ are maximized alternately until convergence and at each iteration the lower-bound $\mathcal{F}$ in Equation (48) increases monotonically.

$$\begin{aligned} Q(D,\theta) &= Q(D)Q(\theta) \\ &= \prod_{D\in\mathcal{D}(q)} Q(D) \prod_{\bar{N}\in\bar{\mathcal{N}}} Q(\theta_{\bar{N}}) \end{aligned} \tag{49}$$

In their experiments Headden et al. use a random initializer and create independent sets with each having several initial settings and run Variational Bayes EM for a fixed number of iterations. The model with best likelihood estimate on a held-out data in each set is then run until convergence. The variational posteriors are then averaged to yield a point estimate of θ. In the case of L-EVG, the model is bootstrapped by running sets of smoothed EVGs as explained above and using the best model from these as initial distribution for corresponding L-EVG. Headden et al. show this random initializer to be better than the harmonic initializer (Klein and Manning, 2004) for both original (DMV) and the extended (EVG, and L-EVG) models. The improvement in the DMV for the random initializer can partly be attributed to the Dirichlet prior employed, but further experiments involving the two initializers are required for a complete understanding of the strengths of the initializers. In the case of the extended models, the additional contexts added to the model might prefer the random initializer because it does not depend on the DMV unlike the harmonic initializer.

Most of the DMV variants model shorter dependencies better than longer ones and this is possibly because the argument generation is conditioned only on valence positions 0 and 1 indicating the generation of first and subsequent arguments. While, modelling larger dependencies through independent valence positions might be better, it might also affect the tractability of the model. One alternative could be to use large dependency fragments (Blunsom and Cohn, 2010) without using the valence.

Adapting the CFG-DMV model further with a Tree Substitution Grammar (TSG), Blunsom and Cohn (2010) defined a hierarchical Pitman-Yor Process (PYP) prior that biases towards a compact grammar. The frontier non-terminal nodes in TSG allows other TSG elementary trees to be added, leading to the generation of the tree until all the non-terminal frontiers are exhausted. The probability of a derivation, tree and sentences for probabilistic TSG (PTSG) are thus defined in a way similar to PCFG.

Continuing experiments with different types of priors in the earlier works, the TSG model use a hierarchical Pitman-Yor Process prior, which allows multiple levels of backoff (including unlexicalized and skip-head) models to be used. It can be seen that the lexicalized, longer TSG rules and multi-level PYP function to compensate for each other. Blunsom and Cohn use Markov Chain

Monte Carlo (MCMC) with Metropolis-Hastings (MS) sampler, so as to efficiently sample full trees by exploiting the factorization of derivation probabilities.

3.6 Summary

Table (1) gives a summary of the major approaches covered in this report including the objective function used and the type of search employed. Just to briefly recap the notations used in the table: q represents a sentence in corpus C. B (bracketing), T (phrase-structure tree) and D (dependency tree) denote the specific instances of different types of structures induced with $\mathcal{B}$, $\mathcal{T}$ and $\mathcal{D}$ refer to the corresponding set of instances pertaining to q. The parameters learned by the model are represented by Θ with the subscripted Θ indicating specific features in the corresponding model.

Algorithm	**Objective**	**Search**	**Output**
EM	$\prod_{q \in C} \sum_{T \in \mathcal{T}(q)} P(T, q \mid \Theta)$	Parametric	PCFG
CCM	$\prod_{q \in C} \sum_{B \in \mathcal{B}(q)} P(B \mid q, \Theta) P(B, q \mid \Theta)$	Structural	Constituent Sequences
DMV	$\prod_{q \in C} \sum_{D \in \mathcal{D}(q)} P(D, q \mid \Theta)$	Structural	Dependency Structure
Prototype (w/ CCM)	$\prod_{q \in C} \sum_{B \in \mathcal{B}(q)} P(B, q \mid \Theta_{ccm}) \sum_{T \in \mathcal{T}(B,q)} P(T, q \mid \Theta_{proto})$	Parametric	PCFG
CE	$\prod_{q \in C} \sum_{T \in \mathcal{T}(q)} P(T, q \mid \mathcal{N}(q), \Theta)$	Parametric	Weighted Dependency structure
CTM	$\prod_{q \in C} P(q \mid \mu, \Sigma)$	Structural	Dependency Structure
EVG	$\prod_{D \in \mathcal{D}(q)} Q(D) \prod_{\bar{N} \in \bar{\mathcal{N}}} Q(\theta_{\bar{N}})$	Structural	Dependency Structure

Table 1: Summary of Different Unsupervised Grammar Induction Approaches - Table idea from Smith and Eisner (2005)

4 Experiments and Results

4.1 Experimental Setup

Most of the work in unsupervised induction has focussed on inducing grammar for English. However, some of them have also experimented with inducing grammars for other languages such as Chinese and German. This section first presents the experiments and discusses the results for grammar induction for English. The results on the other languages are presented towards the end of the section.

ATIS and WSJ are the two datasets that are typically used in the experiments for evaluating different unsupervised induction approaches. ATIS is a corpus of air traffic information system having short sentences concerning the same domain and includes topics such as reservation. The sentences in WSJ corpus come from the Wall Street Journal corpus pertaining to news domain mostly about business and politics. Considering the computational complexity[7] involved in training unsupervised induction algorithms on longer sentences, majority of them restrict the training to a subset of WSJ consisting of 7422 sentences having at most 10 words after removing the punctuation and this set is referred as WSJ10. While the earlier works focussing on this problem used ATIS for evaluation, the recent approaches were usually evaluated on WSJ10.

Both ATIS and WSJ have the phrase structure annotations as part of Penn treebank (PTB) and consequently the approaches inducing the constituent sequences or phrase-structure grammar directly use these annotations for evaluation. In contrast, most of the recent research beginning from DMV (Klein and Manning, 2004) has focussed on inducing dependency structures and thus need to map the phrase-structure annotations to dependency structures using some head-finding techniques. Methods of Collins (1999), Hwa and Lopez (2004) and Yamada and Matsumoto (2003) satisfy this requirement and have been employed in different approaches.

Though different metrics have been proposed for evaluating unsupervised grammar induction (see Section 5), PARSEVAL remains the most widely used metric. PARSEVAL computes a family of measures by matching the brackets between a parser derivation and the corresponding gold parse. The set of measures include *precision* (percentage of total brackets proposed that are correct), *recall* (percentage of total gold brackets that are found) and *F-score* (harmonic mean of precision and recall), with the term *unlabelled* indicating that the constituent identity is ignored if available. Along

[7]The inside-outside dynamic programming algorithm typically used in the unsupervised induction has a cubic running time.

with these, it also defines, average number of crossing brackets (CB) and percentage of constituents having zero or at most 2 crossing brackets (0 CB and $\leq$ 2 CB). The research on dependency induction use the directed dependency attachment accuracy giving the percent of directed attachments proposed by the approach matching with the gold attachments. Some of the works additionally use a weaker version of this by ignoring the direction of attachment as undirected accuracy.

4.2 Results and Discussion

Beginning with a discussion on approaches evaluated on the ATIS corpus, Table (2) presents the results for ABL, EMILE, CDC and CCM using the PARSEVAL metric. First it shows ABL to have significantly higher recall but with lesser precision (overall with higher F-score) than EMILE. This is because EMILE requires enough evidence before incorporating an expression in the grammar, whereas ABL is a greedy approach of including constituents on sparse evidence. This also means that as the corpus size increases, ABL will face scalability issues as it has to now deal with increased number of constituents having n-way overlap.

The last three rows of Table (2) give the results for CDC (after 40 iterations) and CCM approaches. As can be seen CDC has higher precision than ABL and EMILE and has a recall that is comparable to ABL. However CDC scores significantly low on recall when compared with the CCM family of approaches despite being trained on a much larger, complex BNC corpus and using tagged data. Partly the low recall for CDC is because it requires large number of samples in a specific left and right contexts of x and y together so that the corresponding sequence can be considered as a constituent. In other words the Mutual Information (MI) criterion of CDC imposes a strong requirement on the combined left and right contexts leading to low recall. Note that the CCM and generative CCM numbers are also better than the Right-branching baseline[8] and hence are naturally considered as baseline approach in the subsequent works.

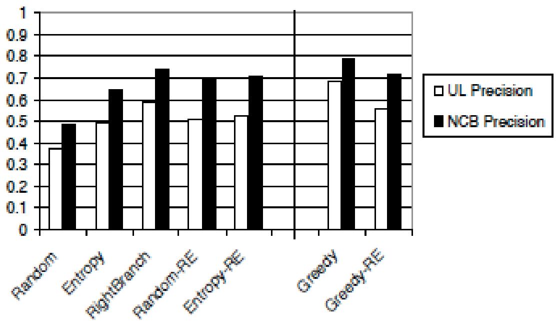
The performance of *greedy-merge* and *constituency-parser* approaches are reproduced from Klein and Manning (Klein and Manning, 2001a) in Figure (10) to better illustrate the performances in different initialization and re-estimation settings. As noted in Section 3.2 greedy-merge (labelled 'Greedy') is a precision oriented system to learn symbolic CFG rules, whereas constituency-parser is forced to generate a complete parse of the sentences and this is evident in Figures (10a) and (10b) respectively showing precision and recall. The different initialization and re-estimation settings used

[8]Right-branching structures works well for English and hence is frequently used as a natural baseline for English

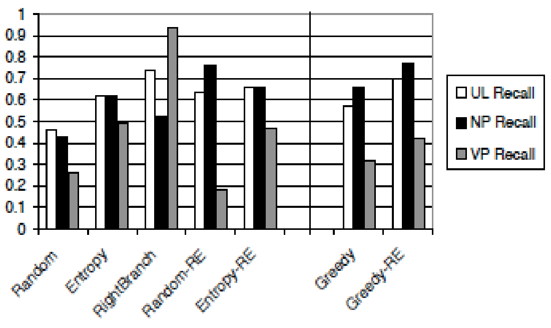
Model	**UR**	**UP**	**F-score**	**CB**	0 **CB**	≤ 2 **CB**
RBRANCH	46.4	39.9	42.9	2.18	-	-
EMILE	16.8	51.6	25.4	**0.84**	**47.4**	**93.4**
ABL	35.6	43.6	39.2	2.12	29.1	65.0
CDC (40 itr)	34.6	53.4	42.0	1.46	45.3	78.2
CCM	46.8	54.4	50.3	1.61	-	-
Gen CCM	**47.6**	**55.4**	**51.2**	1.45	-	-

Table 2: Distributional Analysis Approaches and CCM: Evaluation on ATIS Corpus

by the constituency-parser are shown separately, with the latter settings distinguished by the suffix '-RE'. The Right-branching initialization works well as expected in both precision and recall, closely followed by the Greedy-merge output used in the re-estimation step of EM.



(a) Unlabelled Precision



(b) Unlabelled Recall

Figure 10: Greedy-merge and Constituency Parser: Evaluation on ATIS corpus - Figure from Klein and Manning (2001a)

Turning specifically to the recall, the constituency parser consistently makes mistakes with VP as it frequently combines subject with the verb below the object nouns thus leading to low recall in all the settings except for right-branching initializer. For right-branching initializer, the linguistic bias of placing the dependents after head words works well for English and this ensures that the verb is first combined with the object before it is attached with the subject improving the VP recall significantly.

Table (3) gives the results for several models that followed the period dominated by the approaches employing distributional analysis over linear contexts. To facilitate coherent discussion, the results are grouped logically and show the results for both constituency and dependency parsing results using PARSEVAL and attachment accuracy metrics respectively. The baseline results in the first group confirms the strength of right-branching baseline for constituency parsing. The undirected accuracies of left and right-branching baselines are comparable, while the directed accuracy is better for left branching. Because the sentences are smaller, many sentences do not have dependents following the verb and this improves the directed accuracy for left-branching, which places the head to the right of the dependents; for example the sentence head (verb) will take dependents (subject noun) in the left and none in the right.

The grammatical bigrams (Paskin, 2001) uses a larger set of WSJ corpus (consisting of more than 3.3 mn sentences from volumes 1 & 2 of TREC dataset) for training. Evaluation has been done on a subset of WSJ corpus from the PTB showing undirected accuracy of 39.7% even below the random baseline. The method links two words exhibiting higher mutual information between them, without considering syntactic aspects, their relative positions and so on.

The Constituent-Context Model (CCM) discussed earlier is a generative model over trees focussing on producing a bracketing for the given sentences without labelling them. It outperforms the right-branching baseline in the constituency parsing results for the WSJ10 dataset, as was seen for the ATIS corpus. The DMV on the other hand is a complementary approach and works better for dependency parsing. The joint model of DMV and CCM combines their strengths naturally producing superior results for both types of parsing. The joint model was also evaluated on the induced POS tags instead of the gold annotations for training to result slightly degraded performance. Despite the slight reduction in the results, the experiments by Klein and Manning (2004) are seen positive, because this is the first work demonstrating results for fully unsupervised grammar induction.

Prototype-driven learning (Haghighi and Klein, 2006) uses three different experiments in evaluating the approach. Firstly, it evaluates the strength of the proposed distributional prototype features in inducing labels for the sequences. While the unlabelled F-score is better than the corresponding baselines, it is still lesser than the CCM and the DMV. The weakness of the prototype-driven model lies in its inability to identify the constituent sequences. In order to address this issue, the other two experiments combines this approach with two different models for producing bracketing, viz. PTB gold bracketing and CCM bracketing model.

The joint model with PTB gold bracketing gives oracular performance of 88.1 F-score figures achievable when perfect bracketing information is available. The second joint model uses CCM to bracket the corpus, instead of using gold bracketing and it results in significantly better performance than the simple model using prototype features, and its F-score is also comparable with the CCM

Model	**Recall**	**Precision**	**F-score**	**Dir**	**Undir**
LBRANCH	32.6	25.6	28.7	**33.6**	**56.7**
RANDOM	39.4	31.0	34.7	30.1	45.6
RBRANCH	**70.0**	**55.1**	**61.7**	24.0	55.9
Grammatical Bigrams*	-	-	-	-	39.7
CCM	81.6	64.2	71.9	23.8	43.3
DMV	59.2	46.6	52.1	43.2	62.7
CCM + DMV (Gold POS)	**88.0**	**69.3**	**77.6**	**47.5**	**64.5**
CCM + DMV (Ind POS)	82.8	65.2	72.9	42.3	60.4
Proto	76.2	59.6	66.9	-	-
Proto-Gold	100.0	78.8	88.1	-	-
Proto + CCM	**86.9**	**68.4**	**76.5**	-	-
Proto (Lab)	62.9	51.8	56.8	-	-
Proto-Gold (Lab)	78.7	64.8	71.1	-	-
Proto + CCM (Lab)	**68.5**	**56.9**	**62.2**	-	-
UML-DOP	-	-	**82.9**	-	-

Table 3: Results for PCFG and Dependency Attachment: Evaluation on WSJ10 Corpus

and joint CCM-DMV models. Haghighi and Klein also evaluate their model on labelled sequences and the results are shown in Table (3). The Data-oriented parsing in the unsupervised setting results in a significant 5% improvement in the F-score.

Tables (4) and (5) illustrate the results of several recent variations of the DMV. The former includes the evaluation on WSJ10 dataset focussing on Contrastive Estimation (Smith and Eisner, 2005); EVG and L-EVG (Headden-III et al., 2009), while the latter table gives the results of the DMV variations evaluated on different sets of WSJk covering logistic-normal (Cohen et al., 2008) and shared LN (Cohen and Smith, 2009a) priors and the training regime variation (Spitkovsky et al., 2010a). However, there are minor differences in the test sets or in the experimental setup, so these may not be completely comparable. As an example, CE evaluates using the entire section 23 of the WSJ corpus, whereas some models use WSJ10 dataset for evaluation. And, Cohen et al. (2008; 2009a) introduced the idea of separating the training (sections 1-21 of WSJ) and test (section 23) phases, each using different datasets as in the parentheses, with an optional tuning phase (section 22); this was later adopted by others in experiments.

The results of the CE are shown for different neighbourhoods; with transposition (TRANS)

Model	**Smoothing/ *Prior***	*Harmonic init*		*Uniform/Random init*	
		Dir. Acc.	**Iterations**	**Dir. Acc.**	**Iterations**
EM (generative)	no smoothing	35.2	64.1	23.6	63.3
CE (log-linear) Nhood: LENGTH	$\sigma^2 = 0.1$	**42.9**	150.5	32.4	101.1
	$\sigma^2 = 1$	**42.9**	260.5	33.6	177.0
	no smoothing	42.3	195.2	33.7	173.1
CE (log-linear) Nhood: TRANS	$\sigma^2 = 0.1$	32.4	54.9	41.5	33.8
	$\sigma^2 = 1$	31.5	113.7	48.5	82.5
	no smoothing	37.4	271.3	**48.8**	286.6
CE (log-linear) Nhood: DELORTRANS	$\sigma^2 = 0.1$	32.0	56.2	41.1	38.6
	$\sigma^2 = 1$	**47.1**	132.2	46.7	87.0
	no smoothing	36.4	287.9	46.2	212.8
DMV	no smoothing	46.9	-	55.7*	-
DMV	drop-head	-	-	**61.2***	-
EVG	no smoothing	-	-	53.3*	-
EVG	drop-valence	-	-	62.1*	-
EVG	drop-head	-	-	**65.0***	-
L-EVG	EVG drop-head	-	-	**68.8***	-

Table 4: Results for Dependency Attachment: Evaluation on WSJ10 Corpus

Model	**Prior**	**Directed Accuracy**		
		WSJ10	*WSJ20*	*WSJ∞*
DMV (VB-EM)*	Log-Normal	59.4	45.9	40.5
DMV (VB-EM)*	Shared LN	**61.3**	47.4	41.4
Babysteps (@15)	-	55.5	44.3	39.2
Less is More (@15)	-	56.2	48.2	44.1
Leapfrog (@15)	-	**57.1**	**48.7**	**45.0**
Viterbi EM (smoothed, @45)	-	**65.3**	**53.8**	**47.9**
TSG-DMV	PYP	**67.7**	-	**55.7**

Table 5: Results for Dependency Attachment: Evaluation on WSJ$\{10,20,\infty\}$ of Section 23

neighbourhood having the best performance. The Variational Bayesian models employing log-normal and shared log-normal priors has accuracies (Table (5)) that are better than the CE, using the harmonic initializer. In the case of shared log-normal priors, the result is shown for soft tying of verbs and nouns. It would be interesting to see if tying more categories will be helpful, even though this might be computationally expensive.

Varying the training regime involving the three approaches of Babysteps, Less is More, and Leapfrog (Spitkovsky et al., 2010a) help the longer sentences more than the shorter ones (Table (5)). This could be due to the separated training and test steps with the test step using the models trained on larger WSJ set for evaluation. The TSG-DMV (Blunsom and Cohn, 2010) drastically improves the attachment accuracy on the entire set of WSJ23 to 67.7% and is close to the 68.8% accuracy of the L-EVG for WSJ10.

The effect of different initializers, viz. *harmonic* (a smart initializer that prefers closer attachments than distant ones), *uniform* (initializes all the attachment weights to 0) and *random* (samples from random distribution and then aggregates the best performing ones) is also shown in Table (4). The findings are not conclusive as can be seen from the numbers favouring both harmonic (Klein and Manning, 2004; Cohen et al., 2008; Spitkovsky et al., 2010a) and random[9] (Headden-III et al., 2009; Spitkovsky et al., 2010b) initializers. The lack of clarity about the strength of different initializers has also been confirmed by Smith and Eisner (2005), who demonstrate mixed results across different neighbourhoods as shown in Table (4).

Interestingly, the EVG and L-EVG models by Headden-III et al. (2009), make a mean-field assumption and employ several sets of initializers before averaging the best initializers from each set. Clearly, a single random initializer may not perform to the same level as the averaged point estimate and this probably explains the higher accuracy obtained by these models.

Smoothing the models is found to be helpful in some cases, while in others it affects the performance negatively. For the case of Contrastive Estimation, significant variation in the accuracy (5.9% and 15.1%) was observed for harmonic initializer with different smoothing levels - between stronger smoothing (indicated by a a lower variance of $\sigma^2 = 0.1$) to unsmoothed ($\sigma^2 = \infty$). The unsmoothed models suffered drop in the attachment accuracies to the tune of 15% apart from taking longer time to converge. EVG and L-EVG employed different smoothing methods for its models leading to improvements in the accuracy. In the case of EVG, valence slot information was found to be more important than the head for predicting the arguments. Additionally, smoothing also helps

[9]indicated by * in Table (4)

the original DMV model, with *drop-head* smoothing improving the accuracy by a significant margin of 5.5%. Spitkovsky et al. (2010b) also demonstrate smoothing to be helpful for an approximate Viterbi inference, achieving attachment accuracy better than the EVG model. And, it further shows the smoothing to be harmful with an exact inference method using inside-outside re-estimation for EM.

Yet another point of variation among these methods is the question of the parsing method. Viterbi and Minimum Bayes Risk (MBR) are by far the two important methods used in the unsupervised grammar induction. The results again vary in support of both Viterbi (Spitkovsky et al., 2010a; Spitkovsky et al., 2010b) and MBR (Cohen et al., 2008) decoding, with Viterbi decoding presently having the highest accuracy for WSJ10 for the L-EVG (Headden-III et al., 2009) model. Specifically, the significant gain in the performance obtained by Spitkovsky et al. (2010b) can be attributed to that fact that it uses same method (Viterbi) consistently for both training and prediction, unlike the other methods that use exact inference (EM with inside-outside re-estimation) for training and approximate decoding (Viterbi) during evaluation.

4.3 Results on Chinese and German

Table (6) illustrates the results obtained for Chinese and German, primarily by Klein and Manning (2004), Haghighi and Klein (2006) and Cohen and Smith (2009a; 2009b). Experiments on Chinese use the CTB10 dataset (consisting of 2437 sentences) from the Penn Chinese Treebank and using the head finding rules similar to the head-percolation approach used for English. For the case of German, NEGRA10 dataset from the larger NEGRA corpus, consisting also of the dependency annotations has been used for the experiments.

Without getting into deeper analysis of these results, the most striking aspect can be noted by comparing the performance of Chinese and German relative to that of English. Interestingly, these approaches perform significantly worse for both Chinese and German grammar induction, than English. As an example, best directed dependency accuracies of 55.2% and 50.6%, respectively for Chinese and German is considerably lesser than 68.6% for English. This indicates that the present approaches must be augmented with different feature sets that are more appropriate for the language in question.

Model; ***prior***	**Recall**	**Precision**	**F-score**	**Dir**	**Undir**
			Chinese		
LBRANCH	48.8	26.3	34.2	30.2	43.9
RANDOM	50.7	27.3	35.5	**35.9**	**47.3**
RBRANCH	**53.9**	**29.0**	**37.8**	14.2	41.5
CCM	64.3	34.6	45.0	23.8	40.5
DMV	**66.7**	**35.9**	**46.7**	42.5	54.2
CCM + DMV	62.0	33.3	43.3	**55.2**	**60.3**
UML-DOP	-	-	**47.2**	-	-
Proto + CCM	-	-	**53.2**	-	-
Proto + CCM (Lab)	-	-	**39.0**	-	-
DMV; *LN*	-	-	-	50.1	-
DMV; *Shared LN*	-	-	-	**51.9**	-
DMV- Mixtures (VERBASROOT)	-	-	-	50.5	-
DMV- Mixtures (RA+VAR+SD)	-	-	-	**50.6**	-
			German		
LBRANCH	48.8	27.4	35.1	**32.6**	**51.2**
RANDOM	49.6	27.9	35.7	21.8	41.5
RBRANCH	**60.1**	**33.8**	**43.3**	21.0	49.9
CCM	85.5	48.1	61.6	25.5	44.9
DMV	69.5	38.4	49.5	40.0	57.8
CCM + DMV	**89.7**	**49.6**	**63.9**	**50.6**	**64.7**
UML-DOP	-	-	**67.0**	-	-

Table 6: Unsupervised Induction Results for Chinese and German

4.4 Issues in Unsupervised Grammar Induction

Research on unsupervised natural language grammar induction has typically focussed on either phrase structure representation or dependency grammars. While phrase structure grammars dominated the research in the earlier stages, majority of the recent approaches use dependency grammars for modelling the language. This subsection gives a brief, yet comprehensive summary of the issues common to them, specifically relating to EM and also highlights the disparate issues.

4.4.1 Phrase-structure Grammars

Linguistically, the phrase structure grammar can be thought of as a model of language explaining various constituents in a sentence, along with how these constituents are arranged hierarchically to yield the sentence. The statistical approaches approximate this by using the word sequences - that occur frequently more often than chance, to model the constituents, which are then combined by the production rules of a grammar, maximizing the likelihood (or minimizing the cross-entropy) of the observed corpus. Thus, statistical phrase structure modelling is aimed towards finding a grammar that *predicts* the data well rather than explaining. Unsupervised grammar induction experiments illustrated the lack of correlation between minimum entropy and an optimal grammar (Pereira and Schabes, 1992), as also showing the weakness of the EM search space containing numerous local optima (Carroll and Charniak, 1992; Charniak, 1994).

In a different experiment, de Marcken (1995) further demonstrated issues in the phrase-structure search process using an augmented head-driven model. Analyzing the appropriateness of the inside-outside algorithm for *predicting* the data, de Marcken experimented with a relatively small corpus of 1000 sentences made up of three unique POS tags. By dynamically plotting the changes in the parameters between successive iterations, he showed the inside-outside algorithm to be incapable of performing actual search, because it simply prefers a grammar that concentrates the probability mass on fewer rules.

de Marcken further showed the trade-off in handling the multiple ordered adjunction either by having multiple rules rewriting the same non-terminal or by introducing additional non-terminals separating the rules. While the former approach captures the syntactic relation between the words better (the chain rules cover from the head to all its dependents), it suffers because the rule probabilities are fragmented leading to lower overall sentence probability. On the other hand, the latter approach with additional non-terminals assigns higher probability to the sentence at the cost of ignoring the relations between the head and its far-lying dependents.

These findings illustrate the problems in unsupervised induction of phrase-structure grammars, specifically when using EM. However, some of these also apply to approaches that use some heuristic for identifying constituents by grouping common but unusual word sequences.

A key insight in understanding the shortcomings of the phrase-structure grammar is the discontinuous search space and closely coupled sentence derivation by an ordered series of rules (de Marcken, 1995). de Marcken proposed flattening of the derivation using a simpler grammar that independently captures the head-dependent relations, mostly ignoring the recursive relations in the text. Interestingly, the later interest in the dependency grammars are in tune with these results.

4.4.2 Dependency Structures

Majority of the approaches in unsupervised dependency induction use projective dependency structures and also without any underlying grammar such as PCFG. The projective dependency captures the relationship among the heads and their dependents in a sentence, such that there are no crossing dependencies. While, the flat dependency structure does not provide useful syntactic information, they are helpful in keeping the model simpler and induction easier. However, these dependency models also suffer by the inherent issues of the EM algorithm used for inference.

Spitkovsky et al. (2010b) showed that the likelihood objectives and accuracy are detached from each other for dependency induction. Using a toy corpus, they showed the per-token attachment accuracies to be 40% and 50% for an MLE and a pseudo model[10] respectively, thus highlighting the disconnect.

Spitkovsky et al. further demonstrated - as one might expect from other learning settings, that the cross-entropy and derivation probability based likelihood measures are ineffective in distinguishing a supervised model θ_{sup} from a poor unsupervised model $\tilde{\theta}$, as the latter model can score better on likelihood objectives than the supervised one. Separately, Liang and Klein (2008) also showed the issues with the objective functions in the EM setting for both PCFG and DMV, apart from studying the data sparsity issues.

5 Evaluation Metrics

The widely used evaluation metrics of PARSEVAL and dependency attachment accuracy were explained in Section 4, while discussing the results of the different approaches. This section explains

[10]Psuedo model deterministically attaches a word to a head in the right or left

some of the other metrics that were proposed for evaluating unsupervised grammar induction.

One of the simpler metric is to find the percentage of sentences that can be analyzed fully (with at least one parse) in a given corpus, indicating its *coverage* for the specified corpus (Briscoe and Carroll, 1995). The issue is that, *coverage* as a measure by itself, does not indicate the correctness of the analysis for any of the sentences nor does it give any information about problematic constituents so as to be able to improve the approach.

Structural consistency evaluates the goodness of a corpus as percentage of sentences analyzed and additionally attempts to factor in the quality of the analyses in evaluation. It computes the percentage of sentences in the parser output having one or more of its analyses structurally consistent with the gold parse (Black et al., 1993), where the consistency is defined by the absence of crossing brackets. This was later extended by Briscoe and Carroll (1995) to consider top-n analyses instead of the Viterbi parse. It fails to give credit to almost correct analysis with minor issues and consequently does not help in fine-grained (say constituent-level) error analysis.

The *Parse Base* (PB) metric introduced by Black et al. (1993) measures the average number of ambiguous parses in the corpus after normalizing it by the sentence length. It assumes linear correspondence between the sentence length and the number of parses (longer the sentences, more ambiguous they are, leading to more parses). This was improved later leading to *Average Parse Base* (APB), where the correspondence is assumed to be exponential (Briscoe and Carroll, 1995). Though, this allows the comparison of two or more competing approaches using the same corpus, it fails to distinguish approaches based on coverage. Thus an approach resulting in a low-coverage grammar might score better in this metric than a high-coverage grammar.

Entropy or Perplexity of a corpus is a well known measure to study the unpredictability and ambiguity in the language, with lower entropy/perplexity indicating higher regularity. Thus the probabilistic grammar can be used to generate a number of sentences whose entropy can then be compared with that of a corpus to provide a model independent measure of the regularity of the resulting grammar. While this captures the regularity of the language generated by the grammar, this does not measure the goodness of the parses produced by the grammar.

Moving away from the string-based approaches, several types of tree-based similarity measures have been proposed to allow a finer degree if evaluation (Sampson et al., 1990). The percentage of rules correctly used in a parser derivation (that are found in the gold parse) from the total rules used in the derivation is a well known tree similarity measure. Thus, the tree similarity is tolerant to the errors in the gold annotations. An important drawback is that the numbers do not distinguish between the degree of errors in unmatched rules.

The well-known PARSEVAL metric also known as GEIG (Grammar Evaluation Interest Group) for evaluating parsers computes precision and recall measure using the matching constituents (brackets) between a parser derivation and gold parse. This only requires bracketing annotation in the test corpus and hence allows comparison of different parsers based on dissimilar grammar frameworks on the same footing. It further captures relatively fine-grained information and is also less sensitive to the errors in annotation (the errors are localized). Because of these reasons, this continues to be used widely despite several criticisms against the metric.

Some of the notable critiques against the PARSEVAL are covered in Lin (1995); Carroll and Briscoe (1996) and Manning and Carpenter (1997), which range from a single deep attachment error being heavily penalized (due to the large number of brackets affected) as opposed to an error occurring at a higher level to lack of correspondence between high constituent boundary correctness score and semantic inference. Additionally, there are systematic differences between the annotation schema used the treebanks and the induced grammar. As a result PARSEVAL penalizes the grammars for producing deeper structures than found in the treebank trees, which tend to be flat (Srinivas et al., 1995).

Lin (1995) proposed evaluation based on dependency relationships between the parse output and manual parse by computing the precision and recall for dependency links. If the parse output or the test corpus uses phrase structure representation, they are first converted to dependency analysis automatically with each dependency link capturing the relationship type. This is not without issues; the most important being the loss resulting from the phrase structure to dependency analysis conversion. However, this approach works for grammars using both phrase structure and dependency formalisms, covering majority of the parsers.

Carroll and Charniak (1998) present a detailed survey of different parser evaluation techniques and also present a new evaluation scheme using the syntactic dependency between a head and a dependent. These syntactic dependencies are called grammatical relations (GR) and different types of GR are hierarchically arranged for one or a group of similar languages. The evaluation scheme relies on a parser identifying the heads of different constituents and their relations with the dependents (arguments or modifier or further subtypes). The parser is expected to identify the specific relationship in the hierarchy as far as possible and is allowed to generalize to a higher-level relationship in the hierarchy in case of ambiguity. The evaluation is then done by computing the recall and precision scores over the GRs between the gold data and parser output. While this has commonalities with Lin (1995), there are several differences as well. Importantly, a GR analysis does not correspond to a strict dependency tree (as two or more heads can modify a single dependent) and GR also captures the semantic arguments that are syntactically realized. And finally, a lexical argument that is not realized at surface level (as in subject pro-drop) can be recorded by the GR.

6 Grammar Induction in Multilingual setting

Multilingual language modelling can take different forms when applied in different contexts such as inducing word and/or phrase alignments, synchronous parsing etc. In the case of alignments multilingual modelling capture the word or phrase level correspondences in the sentence pairs and in the case of synchronous parsing, it refers to the simultaneous generation or recognition of correlated sentences. Interestingly these tasks are also directly useful for an important application in NLP, Statistical Machine Translation (SMT). This section presents a brief summary of the works focussing on bilingual/multilingual language modelling, specifically pertaining to synchronous grammar induction mostly targeted for SMT.

Inversion Transduction Grammar (ITG) proposed by Wu (1997) generalizes the transduction grammars that describes correlated language pairs simultaneously, by allowing inverted ordering of the constituents between source and target languages. While the traditional transduction grammars allows rules of the form $A_i \rightarrow B_sC_s, B_tC_t$ (denoted by $A \rightarrow [BC]$), ITG allows the target non-terminals to be inverted resulting in $A_i \rightarrow B_sC_s, C_tB_t$ (denoted by $A \rightarrow \langle BC \rangle$). This makes the ITG bit more flexible and allows it to model structurally-related languages by imposing identical grammatical structure on both source and target sides, but fails to model linguistically divergent languages. A Stochastic ITG version can be obtained by adding probabilities to the individual productions and this stochastic framework lends itself for finding the optimal parse of a sentence-pair using dynamic programming (Wu, 1997).

Among other applications, Wu demonstrate that ITG can be used for identifying the constituents from parallel sentences. This is achieved by using a single non-terminal symbol X and rewriting recursively to generate either a pair of non-terminals or terminal symbols also allowing singleton terminals which does not have equivalents in the other language along with corresponding probabilities. The lexical translation probabilities (generating a terminal pair) is learned through an alignment model, while the probabilities for other rule types are fixed at a small value. Now, running a maximum-likelihood parser selects the parse tree whose constituents have best lexical translation probabilities. Additional post-processing is then applied to find longer constituents.

While, ITG provides a formal handle for a restrictive synchronous grammar, *hierarchical phrase-based translation* (Chiang, 2007) uses a more powerful Synchronous Context-Free Grammars (SCFG) formalism for learning hierarchical phrase pairs from aligned bitext. Beginning from the conventional phrase alignments as building blocks, it incrementally identifies hierarchical rules embedding sub-phrases, whose probabilities are then learned using relative frequency estimation. The procedure is repeated for learning longer hierarchical rules until no further rules can be learned as defined by a set of constraints. It can be efficiently parsed by using a variant of CKY parsing and further speedup can be achieved by using Cube-pruning (Huang and Chiang, 2005). In contrast to the rigid *inverse* reordering allowed by ITG, hierarchical rules in SCFG allow arbitrary reordering of the constituents having a mix of terminals and non-terminals. Because of this and due to heavy lexicalization the space of possible productions is combinatorially huge, making the hierarchical phrase-structure model to be well-suited for SMT.

ITG and hierarchical phrase-based system both have their own advantages as also their shortcomings. Merging the best elements from both approaches, a generative version of SCFG induction using MLE (Blunsom et al., 2008), proposes to generate multiple non-terminal categories like the former at the same time using a powerful SCFG model similar to the latter. The SCFG rules of this model is somewhat restricted and rewrites either a pair of non-terminals (with possibility to re-order) or emits a pair of terminal symbols. The generative process starts from selecting a non-terminal symbol and then iteratively rewriting the two child non-terminals (if available), until no non-terminals remain. It uses a hierarchical Dirichlet prior to model different rule types. Using mean-field approximation for inference, it shows an experiment to successfully recover a toy grammar by induction as also its applicability in SMT.

Exploiting different cues from bitext in improving the monolingual tasks such as POS tagging and supervised parsing, has gained attention in recent times. All these approaches are motivated by the premise that, sentence-pair fragments in a bitext can surely be expected to exhibit identical syntactic constructs, despite the syntactic divergences between the corresponding languages.

Kuhn (2004) exploited the availability of bitext word alignments to improve monolingual parsing performance. A constituent in one language will generally remain so in another language, even if it is syntactically expressed as a different type. Thus, given a contiguous sequence of words in source language aligned to a set of target words that are separated by a wide margin, the source language sequence can be hypothesized to be a destituent. Kuhn uses this alignment dissipation in the bitext to identify destituent spans in monolingual texts by defining a set of constraints for the destituents, which are then used as additional weighting factors for the EM so as to complement the positive evidence observed through rule counts during the inside-outside re-estimation. While, this drastically improves over the right-branching baseline and traditional EM (for English), it still lags behind the CCM (Klein and Manning, 2002).

Snyder et al. (Snyder et al., 2009) use a similar approach to improve monolingual constituency parsing by exploiting the bilingual cues in the parallel data in an entirely unsupervised setting. The idea is to propagate evidence from the unambiguous parallel sentence in one language to learn the structure of an ambiguous sentence in the other language. It uses a generative Bayesian model, to generate a sentence pair starting from an alignment tree whose nodes might correspond either to monolingual or aligned bilingual constituents. This alignment tree enables the model to capture the linguistic divergences and similarities at monolingual and aligned nodes respectively. The model then generates isolated or pair of POS tags for the nodes by sampling from corresponding probability distributions before generating the word alignments. The sentence pair is finally generated from the POS tags and word-alignments. Using MCMC for inference, the approach is shown to improve over the CCM (Klein and Manning, 2002) model.

The soft parameter tying (Cohen and Smith, 2009a) discussed earlier in monolingual setting is another example for bootstrapping unsupervised induction for one language by evidence from the other, even while using non-parallel corpora. It ties the parameters of the hidden grammar for two languages and learns the grammars jointly, which are then tested separately on the corresponding monolingual data. This approach of cross-tying parameters yielded 0.8% increase in the attachment accuracy for English, while marginally increasing the accuracy for Chinese.

7 Conclusion

The report presented the different approaches in unsupervised grammar induction that broadly fall into two types of search, viz. parametric and structural. Majority of the approaches use Expectation-Maximization (EM) algorithm or its variant to estimate the grammar parameters iteratively by maximizing the likelihood of the data and internally using inside-outside or another similar procedure for re-estimating the counts of the events occurring in the grammar.

The earliest approaches using EM for inducing constituency structures are generally shown to be poor choices for modelling the languages, as they attempt to model both constituent identification and labelling tasks simultaneously, and is also shown to be deficient due to the natural disconnect between their maximum likelihood objective and an optimal grammar. In contrast, most of the successful approaches in recent times, beginning from the DMV (Klein and Manning, 2004) use a simplified dependency formalism involving the head-argument relationships between the words in a sentence. However, despite their apparent success the syntactic value of such dependency trees is well below that of the phrase structure trees. The most recent EVG, L-EVG and TSG-DMV approaches combine the simplicity of dependency formalism with the power of a PCFG by imposing a context-free or tree-substitution grammar over the classic DMV to yield state-of-the-art performances.

While the current models are being used for both monolingual parsing and for SMT in a bilingual setting, they are yet to match their counterparts that use some form of supervision. The difference in the parsing performance between supervised and unsupervised approaches is more pronounced in the monolingual setting than in the bilingual setting.

In terms of scalability of the different approaches, recent methods in monolingual induction are shown to be robust and are able to parse longer and complex sentences at test time. However, the complexity of the training routines in these approaches make it expensive to train on sentences of unrestricted length due to a higher level of ambiguity in them and as a result, majority of them still use the simpler WSJ10 data for training.

7.1 Future Directions

The pipeline for *unsupervised* grammar induction actually start from human-annotated Parts-of-Speech (POS) tags. Thus the applicability of these approaches to resource-poor languages that do not have annotated lexical resources including POS tagged corpus and treebanks is very weak. Though the state-of-the-art unsupervised POS induction accuracy has reached over 90% for several POS categories (Schütze, 1995), the methods employing the induced tags report a drastic drop in the F1 score by 5% and more, indicating the grammar induction approaches to be sensitive to the errors in the POS tags. One of the possible future directions could thus focus on addressing this issue.

Secondly apart from being restricted to 10 word sentences, majority of them are trained and evaluated on sentences after stripping the punctuations and symbols off, because the PARSEVAL metric does not include them in evaluation. However, the punctuations have been shown to be useful in the context of supervised dependency parsing (Collins, 1999). It would thus be interesting to see if they can help in unsupervised setting and also to study the models' sensitivity to the noise introduced by their unscrupulous usage in texts. Specifically, it might be possible to exploit non-word symbols and punctuations to identify constituent boundaries or the valence information by adding some features or well-defined heuristics.

While models based on different grammar formalisms capture certain distinct syntactic aspects well, they fail to model some other aspects as was discussed in Section 4.4. In this respect, the recent

EVG, L-EVG (Headden-III et al., 2009) and TSG-DMV (Blunsom et al., 2008) models combine both CFG and DMV to exploit their complementary strengths. A challenging goal would be to explore the possibility of additionally identifying constituent sequences and categorizing them based on syntactic types as in the phrase-structure representation.

Finally, research on unsupervised grammar induction is largely restricted to English, though there is a minor interest in other high-density languages such as Chinese and German. The characteristics of the languages bring in unique requirements for modelling. As an example the morphologically complex languages often tend to be configurable (allowing free word-order) and possibly allowing crossing structures. In such languages the morphological information expressed in the surface sentence could additionally be exploited in the modelling. It is not clear, if the current approaches will be as effective in these unexplored languages, as it is in English.

The margin of difference in the results between English and other minimally explored languages is clearly evident in the results; for example the best F1 score in phrase structure induction drops to 53.2 and 67 for Chinese and German respectively from 76.5 for English. Clearly, the current approaches need to be better adapted for other languages by using either different feature sets or different grammar formalisms.

A Notations Used

Note: Notations having more than one usage are either explicitly clarified as required in the text or will be clear in the context of their usage.

B	: Bracketing distribution
$\mathcal{B}$	: Bracketing of a sentence as observed in annotated data
b_{ijk}	: Binary rule $X_i \rightarrow X_j X_k$
C	: Corpus used
$D_{..}$	: Divergence measure with the subscript indicating its type- JS, KL etc.
D	: Also, a single Dependency structure
$\mathcal{D}$	: Set of Dependency structures
G	: Symbolic or probabilistic grammar
$\mathcal{G}$	: Graph- dependency or other
$H(.)$	: Entropy of certain sequence in the argument
$I_i^o(.)$	: Inside probability of a node rooted by X_i and spanning over a pair of indices specified in the argument of o
$I_i^o(.)$	: Outside probability of generating X_i in a span (specified in the argument) along with with rest of the terminal symbols in o
$\mathcal{N}$	: Neighbourhood in Contrastive Estimation; Also sets of non-terminal equivalence classes in EVG and L-EVG
$P(.)$	: Probability distrituion
$p(.)$	: Probability specific to a model
q_1^n	: An observed sentence of length n
R	: Productions in G
r, s, t	: Span indices in a given observation sequence$o$
S	: Total sentences in corpus C
t	: A specific tree, such that $t \in \mathcal{T}$
T	: Parts-of-Speech tags or the set of terminals in the grammar
$\mathcal{T}$	: Set of trees corresponding to a corpus

u_{im}	: Unary rule $X_i \rightarrow o(m)$
$w^q_{ijk}(.)$	: Count of the rule $X_i \rightarrow X_j X_k$ appearing in different parses of a sentence q for the specified span
$v^q_{im}(.)$	: Count of the rule $X_i \rightarrow q(m)$ appearing in different parses of a sentence q for the specified span
X	: Non-terminals set, also the number of non-terminals
X_i	: Non-terminal symbol of type i
$X(\alpha)$	: Non-terminal type of sequence α
α, β	: Terminal sequences of some span (aka yields), typically POS tags are considered as terminals
$\sigma(.)$	: Signature of a terminal sequence in the argument, typically the adjacent tags or the boundary on either sides
Σ	: Terminal symbols set, also indicates the number of terminals
Θ	: Parameters of a model

References

[Adriaans et al.2000] Pieter W. Adriaans, Marten Trautwein, and Marco Vervoort. 2000. Towards high speed grammar induction on large text corpora. In *SOFSEM '00: Proceedings of the 27th Conference on Current Trends in Theory and Practice of Informatics*, pages 173–186, London, UK.

[Baker1979] J. K. Baker. 1979. Trainable grammars for speech recognition. In D. H. Klatt and J. J. Wolff, editors, *Proc. of Speech Communication Papers for the 97th Meeting of the Accoustical Society of America*, pages 547–550.

[Black et al.1993] Ezra Black, Roger Garside, and Geoffrey Leech. 1993. *Statistically-driven Computer Grammars of English: The IBM/Lancaster Approach.* Rodopi, Amsterdam, The Netherlands.

[Blei and Lafferty2006] David M. Blei and John D. Lafferty. 2006. Correlated topic models. In *In Proceedings of the 23rd International Conference on Machine Learning*, pages 113–120.

[Blei et al.2003] David M. Blei, Andrew Y. Ng, Michael I. Jordan, and John Lafferty. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

[Blunsom and Cohn2010] Phil Blunsom and Trevor Cohn. 2010. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proceedings of EMNLP*, pages 1204–1213.

[Blunsom et al.2008] Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. Bayesian synchronous grammar induction. In *Proceedings of NIPS*.

[Bod2006] Rens Bod. 2006. An all-subtrees approach to unsupervised parsing. In *In Proceedings of ACL-CoLING*, pages 865–872.

[Briscoe and Carroll1995] E. Briscoe and J. Carroll. 1995. Developing and evaluating a probabilistic lr parser of part-of-speech and punctuation labels. In *Proc. of 4th ACL/SIGPARSE International Workshop on Parsing Technologies*, pages 48–58.

[Carroll and Briscoe1996] John Carroll and Ted Briscoe. 1996. Apportioning development effort in a probabilistic lr parsing system through evaluation. In *Proc. of the EMNLP in ACL-SIGDAT 1996*, pages 92–100.

[Carroll and Charniak1992] Glenn Carroll and Eugene Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. Technical report, Brown University.

[Carroll et al.1998] John Carroll, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proceedings of LREC98*.

[Charniak1994] Eugene Charniak. 1994. *Statistical Language Learning*. MIT Press, Cambridge, MA, USA.

[Chiang2007] David Chiang. 2007. Hierarchical phrase-based translation. *In Computational Linguistics, vol. 33*.

[Chomsky1980] Noam Chomsky. 1980. *Rules and Representations*. Columbia University Press, New York, NY.

[Clark2000] Alexander Clark. 2000. Inducing syntactic categories by context distribution clustering. In *Proc. of Conference on Computational Natural Language Learning*, pages 91–94, Lisbon, Portugal.

[Clark2001] Alexander Clark. 2001. Unsupervised induction of stochastic context-free grammars using distributional clustering. In Walter Daelemans and Remi Zajac, editors, *Proceedings of the Fifth Conference on Natural Language Learning at ACL 2001*, pages 113–120.

[Cohen and Smith2009a] Shay B. Cohen and Noah A. Smith. 2009a. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *NAACL '09: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 74–82.

[Cohen and Smith2009b] Shay B. Cohen and Noah A. Smith. 2009b. Variational inference for grammar induction with prior knowledge. In *ACL-IJCNLP '09: Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 1–4.

[Cohen et al.2008] Shay B. Cohen, Kevin Gimpel, and Noah A. Smith. 2008. Logistic normal priors for unsupervised probabilistic grammar induction. In *Proceedings of Neural Information Processing Systems (NIPS) 21*.

[Collins1999] Michael John Collins. 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA.

[de Marcken1995] Carl de Marcken. 1995. On the unsupervised induction of phrase-structure grammars. In *In Proceedings of the Third Workshop on Very Large Corpora*, pages 191–208.

[Dempster et al.1977] A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, pages 1–38.

[Eisner1996] Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *In Proceedings of the 16th International Conference on Computational Linguistics (COLING) 1996*, pages 340–345.

[Haghighi and Klein2006] Aria Haghighi and Dan Klein. 2006. Prototype-driven grammar induction. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 881–888, Morristown, NJ, USA. Association for Computational Linguistics.

[Headden-III et al.2009] William P Headden-III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of the Human Language Technology Conference of the NAACL*, Boulder, Colorado, May.

[Huang and Chiang2005] Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 53–64.

[Hwa and Lopez2004] Rebecca Hwa and Adam Lopez. 2004. On the conversion of constituent parsers to dependency parsers. Technical report, University of Pittsburgh.

[Klein and Manning2001a] Dan Klein and Christopher D. Manning. 2001a. Distributional phrase struction induction. In Walter Daelemans and Remi Zajac, editors, *Proceedings of the Fifth Conference on Natural Language Learning at ACL 2001*, pages 113–120.

[Klein and Manning2001b] Dan Klein and Christopher D. Manning. 2001b. Natural language grammar induction using a constituent-context model. In *Advances in Neural Information Processing Systems 14*. MIT Press.

[Klein and Manning2002] Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 128–135, Morristown, NJ, USA. Association for Computational Linguistics.

[Klein and Manning2004] Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: models of dependency and constituency. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 478, Morristown, NJ, USA. Association for Computational Linguistics.

[Kuhn2004] Jonas Kuhn. 2004. Experiments in parallel-text based grammar induction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*.

[Kurihara and Sato2004] Kenichi Kurihara and Taisuke Sato. 2004. An application of the variational bayesian approach to probabilistic context-free grammars. In *In International Joint Conference on Natural Language Processing Workshop Beyond Shallow Analyses*.

[Lari and Young1990] K. Lari and S. J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4(1):35 – 56.

[Liang and Klein2008] Percy Liang and Dan Klein. 2008. Analyzing the errors of unsupervised learning. In *In Proceedings of HLT-ACL*.

[Lin1995] Dekang Lin. 1995. A dependency-based method for evaluating broad-coverage parsers. In *IJCAI'95: Proceedings of the 14th international joint conference on Artificial intelligence*, pages 1420–1425.

[Manning and Carpenter1997] Christopher D. Manning and Bob Carpenter. 1997. Probabilistic parsing using left corner language models. In *In Proceedings of the 5th ACL/SIGPARSE International Workshop on Parsing Technologies. MIT*, pages 147–158. Kluwer Academic Publishers.

[Omohundro1992] Stephen Omohundro. 1992. Best-first model merging for dynamic learning and recognition. In *Advances in Neural Information Processing Systems 4*, pages 958–965.

[Paskin2001] Mark A. Paskin. 2001. Grammatical bigrams. In *NIPS*, pages 91–97.

[Pereira and Schabes1992] Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th annual meeting on Association for Computational Linguistics*, pages 128–135, Morristown, NJ, USA. Association for Computational Linguistics.

[Pullum and Scholz2002] Geoffrey K. Pullum and Barbara C. Scholz. 2002. Empirical assessment of stimulus poverty arguments. *The Linguistic Review, vol. 19*, pages 9–50.

[Sampson et al.1990] Geoffrey Sampson, Robin Haigh, and Eric Atwell. 1990. Natural language analysis by stochastic optimization: a progress report on project april. *Journal of Experimental and Theoritical Artificial Intelligence*, 1(4):271–287.

[Schütze1995] Hinrich Schütze. 1995. Distributional part-of-speech tagging. In *Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics*, pages 141–148.

[Smith and Eisner2005] Noah A. Smith and Jason Eisner. 2005. Guiding unsupervised grammar induction using contrastive estimation. In *International Joint Conference on Artificial Intelligence (IJCAI) Workshop on Grammatical Inference Applications*, pages 73–82, Edinburgh, July.

[Snyder et al.2009] Benjamin Snyder, Tahira Naseem, and Regina Barzilay. 2009. Unsupervised multilingual grammar induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 73–81, Suntec, Singapore, August. Association for Computational Linguistics.

[Spitkovsky et al.2010a] Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010a. From Baby Steps to Leapfrog: How "Less is More" in unsupervised dependency parsing. In *Proceedings of Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2010)*.

[Spitkovsky et al.2010b] Valentin I. Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D. Manning. 2010b. Viterbi training improves unsupervised dependency parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning (CoNLL-2010)*.

[Srinivas et al.1995] Bangalore Srinivas, Christine Doran, and Seth Kulick. 1995. Heuristics and parse ranking. In *Proc. of 4th ACL/SIGPARSE International Workshop on Parsing Technologies*.

[Stolcke and Omohundro1994] Andreas Stolcke and Stephen M. Omohundro. 1994. Inducing probabilistic grammars by bayesian model merging. In *ICGI '94: Proceedings of the Second International Colloquium on Grammatical Inference and Applications*, pages 106–118, London, UK. Springer-Verlag.

[van Zaanen and Adriaans2001] Menno van Zaanen and Pieter Adriaans. 2001. Comparing two unsupervised grammar induction systems: Alignment-based learning vs. emile. Technical report, University of Leeds.

[van Zaanen2000] Menno van Zaanen. 2000. Abl: Alignment-based learning. In *In Proceedings of COLING-2000*, pages 961–967.

[Wu1997] Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Comput. Linguistics, vol. 23(3)*, pages 377–403.

[Yamada and Matsumoto2003] Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *In Proceedings of International Workshop on Parsing Technologies*, pages 195–206.