Chapter 9
Interactive Parsing

With Contribution Of: José Miguel Benedí, Joan Andreu Sánchez and Ricardo Sánchez-Sáez.

Contents

This chapter introduces the *Interactive Parsing* (IP) framework for obtaining the correct syntactic parse tree of a given sentence. This formal framework allows us to make the construction of interactive systems for tree annotation. These interactive systems can help to human annotators in creating error-free parse trees with little effort, when compared with manual post-editing of the trees provided by an automatic parser.

In principle, the interaction protocol defined in the IP framework differs from the *left-to-right* interaction protocol used throughout this book. Specifically, the IP protocol will be of *desultory order*; that is, in IP the user can edit any part of the parse tree and in any order.

However, in order to efficiently calculate the next best tree in IP framework, in Sect. 9.4, a *left-to-right depth-first* tree review order will be introduced. In addition, this order also introduces computational advantages into the lookout of most probable tree for interactive bottom-up parsing algorithms.

The use of Confidence Measures in IP is also presented as an efficient technique to detect erroneous parse trees. Confidence Measures can be efficiently computed in the IP framework and can help in detecting erroneous constituents within the IP process more quickly, as they provide discriminant information over all the IP process.

A.H. Toselli et al., *Multimodal Interactive Pattern Recognition and Applications*,
DOI 10.1007/978-0-85729-479-1_9,

9.1 Introduction

Probabilistic Parsing is an important problem related to Natural Language Processing and Computational Linguistics, which has proven to be useful for Language Modeling [2, 8, 24], RNA Modeling [25], and Machine Translation [7, 35], among others [19]. In probabilistic parsing, a parse tree is obtained for an input sentence that represents syntactic relations between different parts of the sentence. This parse tree is obtained by using a probabilistic model and a parsing algorithm.

Probabilistic Context Free-Grammars (PCFG) are a powerful formalism that has been widely used for Probabilistic Parsing [1, 10, 23, 24, 30]. PCFG represent an appropriate trade-off between representation capability and time complexity of the algorithms that are able to use them. The good results obtained in parsing have made PCFG the most used formalism in order to tackle this problem. Therefore, we will focus on the use of PCFG as probabilistic models.

A possible classification of the parsing algorithms can be done based on the use of lexical information. In *lexicalized parsing*, lexical information is used in the parsing process in order to disambiguate between non-lexical rules [5]. The main drawback of the probabilistic lexicalized parsing is the high time complexity of the parsing algorithms. Alternatively, *unlexicalized parsing* can be used. In the last years, unlexicalized parsing has achieved very good parsing results with algorithms of lower time complexity [18, 23]. For unlexicalized parsing, algorithms can be grouped into two main approaches: those that are based on the Earley algorithm [12], and those that are based on the Cocke–Kasami–Younger (CKY) algorithm [15]. The Earley algorithm is a classical parsing algorithm that can deal with PCFG in general format. Alternatively, the CKY algorithm has a similar performance but it requires the PCFG in Chomsky Normal Form (CNF). Several reasons have made more popular the use of the CKY algorithm in the last years: first, the CKY algorithm makes easier the use of efficient techniques for obtaining a set of n-best parse trees [16]; second, in recent years, efficient A* algorithms have been proposed for unlexicalized parsing that are able to achieve very good performance [13, 18]; third, efficient maximum-entropy techniques have been devised for CKY-style parsing [6]; and fourth, efficient CKY-style parsing algorithms have been recently proposed for Machine Translation [9, 14, 34]. Throughout this section we will focus in probabilistic parsing with PCFG in CNF with the CKY-style algorithms, but similar ideas could be applied to the Earley algorithm with PCFG in general format, and other parsing formalisms.

In probabilistic parsing, given a sentence x and a PCFG G, the problem in which we are interested is to obtain the parse tree t that best represents the relation between the words of the sentence x according to model G. From a pattern recognition point of view, the probabilistic parsing can be formulated as

$$\hat{t} = \arg\max_{t \in \mathcal{T}} p_G(t \mid x), \tag{9.1}$$

where $p_G(t \mid x)$ is the probability of the parse tree t given the input string x using a PCFG G, and $\mathcal{T}$ is the set of all possible parse trees for x. In probabilistic parsing,

a parse tree t that is associated to a string $x = x_1^n$ can be decomposed into subtrees t_{ij}^A, such that, A is the label of the root node and i, j are indexes delimiting the analyzed substring x_i^j. In this way, $t = t_{1n}^S$ where S is the axiom of the grammar. If the PCFG is in CNF, then the maximization in Eq. (9.1) can be carried out with a dynamic programming CKY-style algorithm. This CKY-style algorithm resembles the Viterbi algorithm for finite-state models, and therefore sometimes is also called a Viterbi algorithm. This algorithm fills in a $(n \times n)$ parse matrix $\mathcal{V}$ for a string of size n. Each element of $\mathcal{V}$ is a probabilistic non-terminal vector such that each component is defined as

$$\mathcal{V}_{i,j}[A] = \hat{p}\left(t_{ij}^A\right) = \hat{p}_G\left(A \stackrel{+}{\Rightarrow} x_i^j\right) \quad A \in N; 1 \leq i, j \leq n, \tag{9.2}$$

where N is the set of non-terminal symbols of the grammar, and $\hat{p}_G(A \stackrel{+}{\Rightarrow} x_i^j)$ is the probability of the most probable tree that generates the substring x_i^j from the non-terminal symbol A.

As we introduced in Chap. 1, within the syntactic and statistical pattern recognition world, we can tell apart two different usage scenarios for automatic systems. First, we have the cases in which the output of such systems is expected to be used in a vanilla fashion, that is, without validating or correcting the results produced by the system. Within this usage scheme, the most important factor of a given automatic system is the quality of the results. Although memory and computational requirements of such systems are usually taken into account, the ultimate aim of most research that relates to this scenario is to minimize the error rate of the results that are being produced [17].

The second usage scenario arises when there exists the need for perfect and completely error-free results. In such a case, the intervention of a human user validator/corrector is unavoidable. The corrector will review the results and validate them, or make the suitable corrections before the system output can be employed. In these kind of problems, the most important factor that has to be minimized is the human effort that has to be applied to transform the potentially incorrect output of the system into validated and error-free output. Measuring user effort has an intrinsic subjectivity that makes it hard to be quantized. Given that the user output, most research about problems associated to this scenario tried to minimize just the error rate of the system as well.

Only recently, more formal work in this direction has started to be carried out, in the form of Interactive Pattern Recognition (IPR) systems (see Chap. 1). These systems formally integrate the correcting user into the loop, making him part of an interactive system (see Fig. 1.4). In such systems the importance of the vanilla error rate per-se is diminished. Instead, the intention is to measure how well the user and the system work together. For this, formal user simulation protocols started to be used as a benchmark. This dichotomy in evaluating system performance or user effort applies to probabilistic parsing as well.

There are many problems within the parsing field where error-free results consisting in perfectly annotated trees are needed. Building correct trees is needed for tasks such as recognition of handwritten mathematical expressions [36] or creating

new gold standard treebanks [11]. When using automatic parsers as a baseline for building perfect syntactic trees, the role of the human annotator is to post-edit the trees and correct the errors. This manner of operating results in the typical two-step process for error correcting, in which the system first generates the whole output and then the user verifies or amends it. This paradigm is rather inefficient and uncomfortable for the human annotator. For example, in the creation of the Penn Treebank annotated corpus, a basic two-stage setup was employed: a rudimentary parsing system provided a skeletal syntactic representation, which then was manually corrected by human annotators [20]. Additional works within this field have presented systems that act as a computerized aid to the user in obtaining the perfect annotation [4, 21]. Subjective measurements of the effort that is needed to obtain perfect annotations were reported in [4], but we feel that a more comparable metric is needed.

With the objective of reducing the user effort and making the laborious task of tree annotation easier, an IPR framework for probabilistic Interactive Parsing (IP) will be presented. In this IP framework, the user is located in the loop, embedding him as a part of the automatic parser, and allows him to interact in real time within the system. Thus, the IP system can use the readily available user feedback to make improved predictions about the parts that have not been validated by the corrector.

To reduce user effort in IPR systems in general, and in IP systems in particular, one approach that can be followed is adding information of the system that helps the user in finding the errors and so he can correct them in a hastier fashion. For the users of such systems it is important to know, not only that the output may be erroneous, but which parts of this more complex output blocks are more prone to be erroneous. Confidence Measures (CM) are a formalism that goes along this direction, allowing the system to assigning a *probability of correctness* for individual erroneous constituents of a more complex output block of a PR system.

In fields such as HTR, MT or ASR the output sentences have a global probability, or score, that reflects the likeness of the whole recognized or translated sentence of being correct. CM allow precision beyond the sentence level in predicting the errors: they allow one to label the individual generated words as either correct or incorrect. This enables systems to identify possible erroneous parts to the user, or to propose only those words that are likely to be correct. CM have been successfully applied in many completely automatic PR systems [26, 31–33]. Recently, CM have also been applied to IPR systems in the HTR [29] field.

In the following section we describe how the IPR framework can be stated for probabilistic IP. Then, we explore the use of CM to the IP framework, to asses how much is retained of their ability to detect erroneous constituents within the interactive process.

9.2 Interactive Parsing Framework

As presented in Sect. 9.1, it is necessary to consider the human user in the parsing process to achieve error-free parse trees. There are two possible approaches: by including the human user in a process of post-editing, or by incorporating the

human user in the interactive recognition process itself. Following the guidance of this book, this latter approach is what we consider in this section. The IPR formal framework introduced in Chap. 1, and more particularly the case for explicitly taking interaction history into account (Sect. 1.3.2), can be directly applied to the IP problem.

According to IPR paradigm, in each interaction, the system should propose a new hypothesis (parse tree) compatible with the constraints imposed by the amendments made by the user in previous interactions. Initially, for a given input string, the system proposes a parse tree. Next, the user corrects a possible error of the proposed parse tree. Then, the system must provide a new parse tree compatible with the user correction. Obviously, this user correction restricts the set of solutions (parse trees) possible. This interactive process continues until a correct parse tree is achieved.

In principle, this interaction protocol differs from the *left-to-right* interaction protocol, raised throughout this book. Following the interaction protocol taxonomy described in Sect. 1.4.1, the IP protocol would be of *desultory order*. That is, in IP the user can edit any part of the parse tree and in any order. Then we will formally characterize the IP problem, and we will also analyze the implicit interaction protocols. In Sect. 9.4 we will return to analyze the interaction protocols for IP and we will propose some restrictions.

From a linguistic point of view, user annotation can be stated in terms of *constituents*. A constituent is a word sequence that functions as a single linguistic unit. More formally, from a parsing point of view, a constituent C_{ij}^{A} is defined by the nonterminal symbol (either a *syntactic label* or a *POS tag*) A and its span ij (the starting and ending indexes which delimit the part of the input sentence encompassed by the constituent). Notice that a parse subtree t_{ij}^{A} defines a constituent set $C(t_{ij}^{A})$. This constituent set is composed of several constituents C_{su}^{B}, with $C_{su}^{B} \in C(t_{ij}^{A})$ and $i \geq s \geq j$ and $i \geq u \geq j$ for all constituents in the set. However, note that a given constituent set C can be the result of different subtrees, if we consider cycles and unit rules.

In an IP approach, the user amends a particular constituent in every interaction for some parse tree t. More precisely, he points out a particular node of the tree and he amends the node label and/or its span. As in Sect. 1.3.2 (Eq. (1.13)), the interactive formal framework for probabilistic parsing can be defined as

$$\hat{t} = \underset{t \in \mathcal{T}}{\arg\max}\, p_G\big(t \mid x, C, C_{ij}^{A}\big), \tag{9.3}$$

where C_{ij}^{A} is the (feedback) constituent validated by the user in the last interaction; C is the set of constituents validated by the user (history); $p_G(t \mid x, C, C_{ij}^{A})$ is the probability (using a model G) of a parse tree t given the input string x, the user feedback C_{ij}^{A}, and the history C; and finally $\mathcal{T}$ is the set of all possible parse trees for x.

At this point, we consider that the user feedback is deterministic (mouse and/or keyboard). That is, the decoding process of user feedback does not introduce new errors. In Sect. 1.3.5 we will see how to include a non-deterministic multimodal feedback using interaction information to help decoding non-deterministic feedback.

Note that the definition of C is general, so if we were to define a certain order in the user review process, then the set C would not only include the constituents directly amended by the user but also the components implicitly validated by each interaction with the IP system. As seen in previous chapters, this also happens in applications where the order of analysis is from left to right. In that case, when the user edits a word, this means that this implicitly validates the previous prefix.

In Eq. (9.3), the search algorithm should take into account the restrictions introduced by C and C_{ij}^{A}, that is, the search space defined by the possible parse trees. In both cases, the restrictions introduced by the user should limit the possible solutions represented in the parse matrix $\mathcal{V}$. Note that both C and C_{ij}^{A} produce, somehow, a partial labeled bracketing of the input sequence. In this way, and following [22], we define a *compatibility function* $c(Y, r, s)$ for all those subproblems defined on $\mathcal{V}$ (Eq. (9.2)) compatible with the constituents of C. In other words, given a subproblem $\mathcal{V}_{r,s}[Y]$, its compatibility with C can be defined using the following function:

$$\forall C_{pq}^{X} \in \left(C \cup \left\{C_{ij}^{A}\right\}\right)$$

$$c(Y, r, s) = \begin{cases} 1, & \text{if } (r, s) = (p, q) \wedge (Y = X), \\ 0, & \text{if } (r, s) = (p, q) \wedge (Y \neq X), \\ 1, & \text{if } (r, s) \neq (p, q) \text{ but are consistent}, \\ 0, & \text{otherwise.} \end{cases} \tag{9.4}$$

This function filters those derivations (or partial derivations) whose parsing is not compatible with the validated constituents. The span (r, s) is not consistent with the span (p, q) when $p \leq r \leq q < s$, or $r < p \leq s \leq q$.

The compatibility function between $\mathcal{V}_{r,s}[Y]$ and C_{pq}^{X} defined in Eq. (9.4) requires that when the spans are equal also the labels Y and X must be equal. However, if the grammar G has unary rules $(A \rightarrow B)$ then this restriction is excessive, and should be relaxed as $X \overset{*}{\Rightarrow} Y$ or $Y \overset{*}{\Rightarrow} X$.

9.3 Confidence Measures in IP

Annotating trees syntactically, even with the aid of automatic systems, generally requires human intervention with a high degree of specialization. This fact partially justifies the shortage in large manually annotated treebanks. Endeavors directed at easing the burden for the experts performing this task could be of great help. One approach that can be followed in reducing user effort within an IPR paradigm is adding information that helps the user to locate the individual errors in a sentence, so he can correct them in a hastier fashion. The use of the Confidence Measure (CM) formalism goes in this direction, allowing us to assign a probability of correctness for individual erroneous constituents of a more complex output block of a Pattern Recognition system.

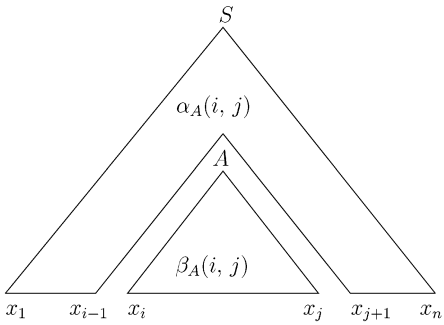

Fig. 9.1 The product of the inside and outside probabilities for each constituent comprises the upper part of Eq. (9.6)

It is interesting to see this issue from the perspective of interaction protocols introduced in Sect. 1.4.1. That section discussed two main types of interactive protocols: *passive*, when the user decides which parse tree (hypothesis) elements need supervision; and *active*, when the system decides which parse tree elements undergo user supervision. The IP paradigm framework seen so far is clearly based on a passive strategy. The introduction of CM associated to the parse tree elements provided by the system can be seen as a first step toward an active approach, where the system suggests/helps the user in the correction process.

However, until recent advances, the use of CM remained largely unexplored in probabilistic parsing despite having several applications of great interest within this field. Assessing the correctness of the different parts of a parsing tree can be useful in improving the efficiency and usability of an IP systems, not only by *coloring* parts with low confidence for the user to spot on, but also the automatic part of the interactive process by forcing the user to correct constituents with low confidence. Additionally, CM could also help improving the parsing process itself, either by being used as a component of an n-best reranker, or by being directly employed by a parsing system for recalculating parts with low confidence.

CM for parsing in the form of combination of characteristics calculated from n-best lists were explored in [3]. Computation of CM from n-best list makes sense mainly when the parsing algorithm prunes the search space by using some A* strategy [6]. However, when no pruning strategy is used in the parsing process, CM can be computed efficiently from the posterior probability of the tree constituent [27].

CM have been also explored in the IP framework. CM of each one of the parse subtrees (subproblems) can be calculated in terms of their posterior probability [27], which can be considered as a measure of the degree to which the subtree is believed to be correct for a given input sentence x. This is formulated as

$$p_G\left(t_{ij}^A \mid x\right) = \frac{p_G(t_{ij}^A, x)}{p_G(x)} = \frac{\sum_{t' \in \mathcal{T}} \delta(t_{ij}^A, t_{ij}'^A) p_G(t' \mid x)}{p_G(x)} \tag{9.5}$$

with $\delta()$ being the Kronecker delta function. Equation (9.5) is the posterior probability of the subtree t_{ij}^A given x. The numerator stands for the probability of all parse trees of x that contain the subtree t_{ij}^A (see Fig. (9.1)).

The posterior probability is computed with the well know *Inside* β and *Outside* α probabilities. The *Inside* probability is defined as $\beta_A(i,j) = p_G(A \overset{*}{\Rightarrow} x_i \dots x_j)$, and it can be computed with the *Inside* algorithm. The *Outside* probability is defined as $\alpha_A(i,j) = p_G(S \overset{*}{\Rightarrow} x_1 \dots x_{i-1} A x_{j+1} \dots x_n)$ and it can be computed with the *Outside* algorithm [1]. In this way, Eq. (9.5) can be written as follows:

$$p_G\left(t_{ij}^A \mid x\right) = \frac{p_G(t_{ij}^A, x)}{p_G(x)} = \frac{\beta_A(i,j)\,\alpha_A(i,j)}{\beta_S(1,n)}. \tag{9.6}$$

It should be noted that the calculation of CM reviewed here is generalizable for any problem that employs PCFG, and not just IP tasks. In the experiments presented in Sect. 9.5, we show that the CM can be used within IP to detect erroneous constituents.

9.4 IP in Left-to-Right Depth-First Order

Parting from the general *desultory order* framework defined in Sect. 9.2 we can instantiate the *left-to-right* tree review order, similarly to what has been done in previous chapters through this book. For that, we take a cue from the prefix/suffix paradigm in order to introduce a predefined review order for the user checking the constituents in each parse trees: a left-to-right depth-first exploration order. In addition to seeming a reasonable and ergonomic review order for the structure of a tree (the reviewer would check constituents in a hierarchical order) this order also introduces computational advantages into the most probable tree lookout for interactive bottom-up parsing algorithms. Another key benefit of adopting this order is that it facilitates the automatic simulation of user interaction, allowing one to calculate metrics that estimate the amount of effort reduction.

This order can be formalized by defining a *prefix tree* $t_p(t, C_{ij}^A)$ that is defined for each correction performed by the user over a constituent C_{ij}^A. The *prefix tree* is comprised of all the ancestors of the corrected constituent and all constituents whose end span is lower than the start span of the corrected constituent. In terms of subtrees, the prefix tree can be represented with the following expression:

$$t_p\left(t, C_{ij}^A\right) = \left(t - t_{ij}^A\right) - \left\{t_{pq}^B \in t : p > j\right\} \tag{9.7}$$

with $t - t'$ meaning to remove the subtree t' from the tree t.

In terms of constituents, Eq. (9.7) is equivalent to

$$\begin{aligned} t_p\left(t, C_{ij}^A\right) = &\left\{C_{mn}^B \in C(t) : m \le i, n \ge j, d\left(C_{mn}^B\right) \le d\left(C_{ij}^A\right)\right\} \\ &\cup \left\{C_{pq}^D \in C(t) : q < i\right\} \end{aligned} \tag{9.8}$$

with $d(C_{ab}^Z)$ being the depth (distance from root) of constituent C_{ab}^Z. In Fig. 9.2d one can see that the slash-outlined part becomes the *prefix tree* and the dot-outlined subtree part becomes recalculated.

In this way, the history from Eq. (9.3) in Sect. 9.2 becomes the prefix $C = t_p(C_{ij}^A)$. Because of this, the compatibility function becomes further restricted:

$$\forall C_{pq}^X \in \left(C \cup \left\{C_{ij}^A\right\}\right)$$

$$c(Y, r, s) = \begin{cases} 1, & \text{if } (r,s) = (p,q) \wedge Y = X, \\ 0, & \text{if } (r,s) = (p,q) \wedge Y \neq X, \\ 1, & \text{if } (r,s) \neq (p,q) \wedge r \geq p \text{ and are consistent}, \\ 0, & \text{otherwise.} \end{cases} \tag{9.9}$$

Notice that the restriction $r \geq p$ comes from the converse of Eq. (9.7).

9.4.1 *Efficient Calculation of the Next Best Tree*

If we employ PCFGs as the underlying parse model for IP, then computing the next best tree as expressed in Eq. (9.3) becomes simpler when one uses a *left-to-right depth-first* tree review order.

The following calculation takes advantage of the fact that there are two types of subtrees that do not have to be recalculated: subtrees that are part of the prefix because they have already been implicitly validated, and subtrees that are not part of the prefix but do not descend from the parent of the corrected constituent. This follows from the fact that under the *left-to-right* review order we know that the parent of an amended constituent is already correct so, owing to the context-freeness of PCFG, a change in a constituent only affects its descendants and its right sibling subtrees at most.

In the following expressions we show how $\hat{t}$, which is the next best tree produced by the IP framework can be calculated in an efficient manner by reusing parts of $\hat{t}'$, the best tree obtained in the previous user iteration.

Let the corrected constituent be C_{ij}^A and its parent C_{st}^D then the next tree $\hat{t}$ can be calculated in the following manner:

$$\hat{t} = \arg\max_{t \in \mathcal{T}} p_G\left(t \mid x, C, C_{ij}^A\right) = \left(\hat{t}' - \hat{t}_{st}'^D\right) + \hat{t}_{st}^D \tag{9.10}$$

with

$$\hat{t}_{st}^D = \arg\max_{t_{st}^D \in \mathcal{T}_{st}} p_G\left(t_{st}^D \mid x_s^t, C_{ij}^A\right) \tag{9.11}$$

where t_{mn}^A is the subtree of t that has constituent C_{mn}^A as the root.

Equation (9.10) calculates the newly proposed tree $\hat{t}$ by subtracting the subtree rooted at the parent of the corrected constituent $(\hat{t}' - \hat{t}_{st}'^D)$ and appending a newly calculated subtree $\hat{t}_{st}'^D$, whose root is the parent of the corrected constituent and is calculated taking into account just the corrected constituent as shown in Eq. (9.11).

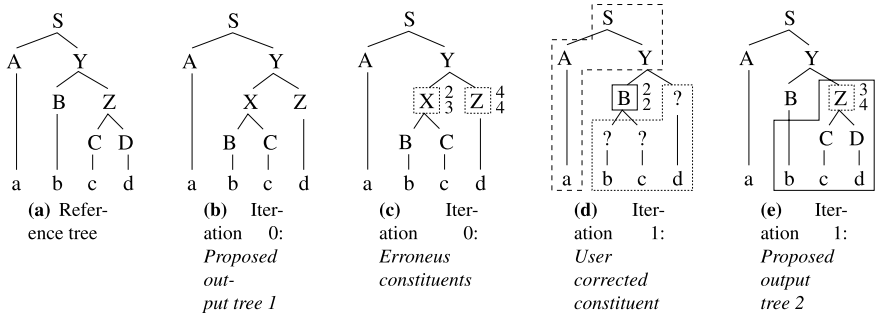

Fig. 9.2 Synthetic example of user interaction with the IP system. (**a**) The reference tree. (**b**) The system proposes an initial tree. (**c**) The user simulation subsystem looks at the tree and detects two incorrect constituents. (**d**) The first error is corrected by the user simulation subsystem. Note the implicitly validated prefix (*slashed outline*) and the recalculated part (*dotted outline*). (**e**) The system produces a new tree which equals the reference

This new maximization is more limited in extent and easier to perform because we only consider the last corrected constituent rather than the whole history of corrected constituents.

9.5 IP Experimentation

Based on the theoretical framework instantiated in Sect. 9.4, we devised an experimental setup to obtain an automatic assessment of user effort savings when using an IP system compared to a traditional system. The experimental setup is based on a user simulation subsystem that uses the gold reference trees to imitate system interaction by a human corrector and provides a comparable benchmark.

9.5.1 User Simulation Subsystem

Again, we devised an automatic evaluation protocol following the aforementioned left-to-right depth-first review order. The protocol is quite simple, and an example can be seen in Fig. 9.2a.

1. The IP system proposes a full parse tree $\hat{t}$ for the input sentence.
2. The user simulation subsystem finds the first incorrect constituent by exploring the tree in the order defined by the prefix tree definition (left to right, depth-first) and comparing it with the *reference tree*. When the first erroneous constituent is found, it is amended by being replaced by the correct one C_{ij}^{A}, operation which implicitly validates the prefix tree $t_p(C_{ij}^{A})$.
3. The IP system produces the most probable tree that is compatible with the validated prefix tree.

4. These steps are iterated until a final, perfect parse tree is produced by the IP system and validated against the *reference* by the user simulation subsystem.

9.5.2 Evaluation Metrics

With the user simulation in place, we need some metrics to measure the effort reduction. Parsing quality is generally assessed by the classical evaluation metrics, *Precision*, *Recall*, and *F-measure*:

- *Precision*: number of correct constituents divided by the number of constituents in the gold reference parse tree.
- *Recall*: number of correct constituents divided by the number of constituents in the proposed parse.
- *F-measure*:

$$2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

However, for the assessment of an IP process, we need two comparable metrics: one that reports the amount of human correcting work needed to obtain the gold tree in a classical two-step process (i.e. the number of operations needed to post-edit the proposed tree in order to obtain the gold one); and a second one that measures the amount of effort needed to obtain the gold tree with the human interacting within the presented IP system.

We defined the following metric that measures the amount of effort needed in order to post-edit a proposed tree and obtain the gold reference parse tree, akin to the Word Error Rate used in Statistical Machine Translation and related fields:

- *Tree Constituent Error Rate (TCER)*: Minimum number of constituent substitution, deletion and insertion operations needed to convert the proposed parse tree into the corresponding gold reference tree, divided by the total number of constituents in the reference tree.

The TCER is in fact strongly related to the F-measure: the higher the F-measure is, the lower TCER will be.

Finally, the relevant evaluation metric that assesses the IP system performance represents the amount of effort that the operator would have to spend using the system in order to obtain the gold tree, and is directly comparable to the TCER:

- *Tree Constituent Action Rate (TCAC)*: Number of constituent corrections performed using the IP system to obtain the reference tree, divided by the total number of constituents in the reference tree.

9.5.3 Experimental Results

An IP system was implemented over the classical CKY-Viterbi algorithm. Experimentation was run over the Penn Treebank (English language) and the UAM Treebank (Spanish language). Within the Penn Treebank, sections 2 to 21 were used to obtain a vanilla Penn Treebank Grammar; test set was the section 23. We divided the UAM Treebank into two sets as well: the set used to obtain the grammars comprised was of the first 1 400 sentences of the corpus; and the test set consisted of the remaining 100 sentences.

Parsers based on CKY work with grammars in CNF, which is a subtype of CFG in which all the production rules must be in the form:

$$A \to BC \quad \text{or} \quad A \to a \quad \text{or} \quad S \to \epsilon. \tag{9.12}$$

Notice that for every CFG (or PCFG) there is an equivalent version of this grammar in CNF.

In order to obtain the CNF grammars from the vanilla PCFG for each of the languages we used the CNF transformation method from the toolkit NLTK.[1] In addition to obtaining a plain right-factored CNF transformation, the CNF method from the NLTK allows to augment the performance of the obtained unlexicalized binary grammars by introducing additional context information in the non-terminals names. This process is called grammar markovization and is controlled by the vertical (v) and horizontal (h) markovization parameters [18]. A plain right-factored CNF transformed grammar corresponds to a markovization with both v and h set to 0.

A basic schema was introduced for parsing sentences with out-of-vocabulary words: when an input word could not be derived by any of the preterminals in the vanilla treebank grammar, a very small probability for that word was uniformly added to all of the preterminals.

For our experiments we obtained several CNF grammars of different sizes through the use of different v and h values. Results for the metrics discussed on Sect. 9.5.2 for the different markovizations of the obtained grammars can be seen in Tables 9.1 and 9.2. We observe that the percentage of corrections needed using the IP system is much lower than the rate of corrections needed on just post-editing the proposed trees: from 42% to 47% effort reduction by the human supervisor. These results clearly show that an IP system can relieve manual annotators of a lot of burden in their task.

Additionally, we performed experimentation with CM used over an IP process, to assess their power to detect incorrect constituents. We assessed that CM retain all of their error detection capabilities during the IP process: they are able to discern between 18% and 25% of incorrect constituents at most stages of the IP process, with a bump up to 27% after about seven user interactions. The complete details can be found at [28].

[1] http://nltk.sourceforge.net/.

Table 9.1 Results for the test set of the Penn Treebank: F_1 and TCER for the baseline system; TCAC for the IP system; relative reduction between TCER and TCAC

PCFG	Baseline		IP	RelRed
	F_1	TCER	TCAC	
$h=0, v=1$	0.67	0.40	0.22	45%
$h=0, v=2$	0.68	0.39	0.21	46%
$h=0, v=3$	0.70	0.38	0.22	42%

Table 9.2 Results for the test set of the UAM Treebank: F_1 and TCER for the baseline system; TCAC for the IP system; relative reduction between TCER and TCAC

PCFG	Baseline		IP	RelRed
	F_1	TCER	TCAC	
$h=0, v=0$	0.57	0.48	0.26	46%
$h=0, v=1$	0.59	0.47	0.25	47%
$h=0, v=2$	0.62	0.44	0.24	46%
$h=0, v=3$	0.61	0.45	0.24	47%

Note that the presented experiments were done using parsing models that perform far from the latest F_1 results; their intention was to assess the utility of the IP schema. However, we expect that relative reductions with IP systems incorporating state-of-the-art parsers would be relevant as well.

9.6 Conclusions

In this chapter, we have introduced a novel Interactive Parsing framework which can be operated by a user to obtain error-free syntactic parse trees. This compares to the classical two-step schema of manually post-editing the erroneous constituents produced by the parsing system. In the general IP framework presented, the interaction protocol that we have initially defined is desultory type: the user can edit any part of the parse tree and in any order. However, to efficiently increase the computation of next best parse tree, a left-to-right depth-first tree review order has been introduced.

To make an automatic experimental assessment, we have simulated the user interaction with the system. Since we have the reference parser, this experimental feature has been possible. We have also defined and calculated some evaluation metrics. In general, the achieved results showed that in an IP system is produced a high amount of effort reduction for a manual annotator compared to a two-step system.

In addition, we have proved that using confidence measures to discriminate incorrect from correct constituents helps to some extent in the IP process. In this point, a purely statistical confidence measure (based on inside-outside estimated posterior probability of constituents) for probabilistic parsing has been introduced.

Finally, is important to note that, in addition to the automatic experimental evaluation reported in previous section, a complete IP prototype has been implemented (see Chap. 12) and made available to potential real users.

References

1. Baker, J. K. (1979). Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, *65*, 31–35.
2. Benedí, J. M., & Sánchez, J. A. (2005). Estimation of stochastic context-free grammars and their use as language models. *Computer Speech & Language*, *19*(3), 249–274.
3. Benedí, J. M., Sánchez, J. A., & Sanchis, A. (2007). Confidence measures for stochastic parsing. In *Proceedings of the international conference recent advances in natural language processing* (pp. 58–63), Borovets, Bulgaria.
4. Carter, D. (1997). The TreeBanker. A tool for supervised training of parsed corpora. In *Proceedings of the workshop on computational environments for grammar development and linguistic engineering* (pp. 9–15), Madrid, Spain.
5. Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the national conference on artificial intelligence* (pp. 598–603), Providence, Rhode Island, USA.
6. Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of the first conference on North American chapter of the association for computational linguistics* (pp. 132–139), Seattle, Washington, USA.
7. Charniak, E., Knight, K., & Yamada, K. (2003). Syntax-based language models for statistical machine translation. In *Machine translation summit, IX international association for machine translation*, New Orleans, Louisiana, USA.
8. Chelba, F., & Jelinek, C. (2000). Structured language modeling. *Computer Speech and Language*, *14*(4), 283–332.
9. Chiang, D. (2007). Hierarchical phrase-based translation. *Computational Linguistics*, *33*(2), 201–228.
10. Collins, M. (2003). Head-driven statistical models for natural language parsing. *Computational Linguistics*, *29*(4), 589–637.
11. de la Clergerie, E. V., Hamon, O., Mostefa, D., Ayache, C., Paroubek, P., & Vilnat, A. (2008). PASSAGE: from French parser evaluation to large sized treebank. In *Proceedings of the sixth international language resources and evaluation* (pp. 3570–3577), Marrakech, Morocco.
12. Earley, J. (1970). An efficient context-free parsing algorithm. *Communications of the ACM*, *8*(6), 451–455.
13. Gascó, G., & Sánchez, J. A. (2007). A* parsing with large vocabularies. In *Proceedings of the international conference recent advances in natural language processing* (pp. 215–219), Borovets, Bulgaria.
14. Gascó, G., Sánchez, J. A., & Benedí, J. M. (2010). Enlarged search space for sitg parsing. In *Proceedings of the North American chapter of the association for computational linguistics—human language technologies conference* (pp. 653–656), Los Angeles, California.
15. Hopcroft, J. E., & Ullman, J. D. (1979). *Introduction to automata theory, languages and computation*. Reading: Addison-Wesley.
16. Huang, L., & Chiang, D. (2005). Better k-best parsing. In *Proceedings of the ninth international workshop on parsing technology* (pp. 53–64), Vancouver, British Columbia. Menlo Park: Association for Computational Linguistics.
17. Jain, A. K., Duin, R. P., & Mao, J. (2000). Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*, 4–37.
18. Klein, D., & Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st annual meeting on association for computational linguistics* (Vol. 1, pp. 423–430), Association for Computational Linguistics Morristown, NJ, USA.
19. Lease, M., Charniak, E., Johnson, M., & McClosky, D. (2006). A look at parsing and its applications. In *Proceedings of the twenty-first national conference on artificial intelligence*, Boston, Massachusetts, USA.
20. Marcus, M. P., Santorini, B., & Marcinkiewicz, M. A. (1994). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, *19*(2), 313–330.

21. Oepen, S., Flickinger, D., Toutanova, K., & Manning, C. D. (2004). LinGO redwoods. *Research on Language and Computation*, *2*(4), 575–596.
22. Pereira, F., & Schabes, Y. (1992). Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th annual meeting of the association for computational linguistics* (pp. 128–135). Newark: University of Delaware.
23. Petrov, S., & Klein, D. (2007). Improved inference for unlexicalized parsing. In *Conference of the North American chapter of the association for computational linguistics; proceedings of the main conference* (pp. 404–411), Rochester, New York.
24. Roark, B. (2001). Probabilistic top-down parsing and language modeling. *Computational Linguistics*, *27*(2), 249–276.
25. Salvador, I., & Benedí, J. M. (2002). RNA modeling by combining stochastic context-free grammars and n-gram models. *International Journal of Pattern Recognition and Artificial Intelligence*, *16*(3), 309–315.
26. San-Segundo, R., Pellom, B., Hacioglu, K., Ward, W., & Pardo, J. M. (2001). Confidence measures for spoken dialogue systems. In *IEEE international conference on acoustic speech and signal processing* (Vol. 1), Salt Lake City, Utah, USA.
27. Sánchez-Sáez, R., Sánchez, J. A., & Benedí, J. M. (2009). Statistical confidence measures for probabilistic parsing. In *Proceedings of the international conference on recent advances in natural language processing* (pp. 388–392), Borovets, Bulgaria.
28. Sánchez-Sáez, R., Leiva, L., Sánchez, J. A., & Benedí, J. M. (2010). Confidence measures for error discrimination in an interactive predictive parsing framework. In *23rd International conference on computational linguistics* (pp. 1220–1228), Beijing, China.
29. Serrano, N., Sanchis, A., & Juan, A. (2010). Balancing error and supervision effort in interactive-predictive handwriting recognition. In *Proceeding of the 14th international conference on intelligent user interfaces* (pp. 373–376), Hong Kong, China.
30. Stolcke, A. (1995). An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, *21*(2), 165–200.
31. Tarazón, L., Pérez, D., Serrano, N., Alabau, V., Terrades, O. R., Sanchis, A., & Juan, A. (2009). Confidence measures for error correction in interactive transcription of handwritten text. In *LNCS: Vol. 5716. Proceedings of the 15th international conference on image analysis and processing* (pp. 567–574), Salerno, Italy.
32. Ueffing, N., & Ney, H. (2007). Word-level confidence estimation for machine translation. *Computational Linguistics*, *33*(1), 9–40.
33. Wessel, F., Schluter, R., Macherey, K., & Ney, H. (2001). Confidence measures for large vocabulary continuous speech recognition. *IEEE Transactions on Speech and Audio Processing*, *9*(3), 288–298.
34. Wu, D. (1997). Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, *23*(3), 377–404.
35. Yamada, K., & Knight, K. (2002). A decoder for syntax-based statistical MT. In *Meeting of the association for computational linguistics*, Philadelphia, Pensilvania, USA.
36. Yamamoto, R., Sako, S., Nishimoto, T., & Sagayama, S. (2006). On-line recognition of handwritten mathematical expressions based on stroke-based stochastic context-free grammar. In *10th international workshop on frontiers in handwriting recognition* (pp. 249–254), La Baule, France.