Polytechnic University
of Valencia

Department of Information
Systems and Computation

A statistical confidence measure framework for probabilistic parsing

Final dissertation

for the

Master in Pattern Recognition, Artificial Intelligence and Digital Imaging

By:

Ricardo Sánchez-Sáez

Directed by:

José Miguel Benedí Ruiz
Joan Andreu Sánchez Peiró

Valencia, November 2008

Contents

List of Figures

List of Tables

Prologue

Since April 2007, I am enjoying an FPU scholarship, awarded by the Spanish IT minister, which allows me to carry out research for my PhD thesis. I am working towards this goal at the Pattern Recognition and Human Language Technology (PRHLT) research group, which is located at the Departamento de Sistemas Informáticos y Computación (DSIC) and at the Instituto Tecnológico de Informática (ITI), both pertaining to the Universitat Politècnica de València (UPV).

In this second year I am finishing the official Master in Artificial Intelligence, Pattern Recognition and Digital Imaging, which acts as the starting part of the PhD thesis course. This document is the Master's Thesis, and in it we will present the carried out research works and the outline of what will be my PhD Thesis.

Chapter 1

Preliminaries

1.1 Introduction

The work concerned by this thesis deals with the introduction of confidence measures in the parsing world.

Parsing, when referred to Natural Language Processing (NLP), means to recognize an input sentence and assign it a syntactic structure, or tree.

A parse tree is composed by substructures that are referred as edges, or constituents. An edge is defined by a syntactic tag (or nonterminal) that spans a substring of the input sentence. Thus, a tree is precisely determined by all its edges, and always has a root edge that spans the whole input sentence.

Several approaches exist when constructing parsing algorithms. One of them is statistical parsing, in which the most fitting parse tree for a given input string is calculated according to probabilistic criteria.

Statistical parsing usually makes use of an objective function that is maximized to obtain the best tree. Several objective functions can be chosen: one of the most commonly used is the whole tree probability, which is maximized according to probabilities of grammar production rules, as it is done in the classical Cocke-Younger-Kasami algorithm (CYK) [1]; quite different objective functions can be used, like the one considering the probability of individual edges, independently of the probability of the whole tree, as it is done by Goodman in [13]. We have pointed out the specific objective function by Goodman because it is closely related to the application of confidence measures to parsing, as both techniques share some theoretical foundations.

Confidence measures are a formalism that allow us to determine whether the individual part of a given output is correct. Confidence measures are widely used in fields like Automatic Speech Recognition (ASR) and Statistical

Machine Translation (SMT) for different tasks. Within these two areas, they give us the probability of correctness for each individual word pertaining to a given output sentence.

In this work, we introduce the adaptation of the confidence measure framework to parsing, which remains a largely unexplored field. We research the use of confidence measures to assess the probability of each tree constituent being correct. We apply confidence measures to trees obtained by the first parsing approach mentioned above (maximizing the whole tree probability).

We present expressions for calculating this statistical confidence measure for each edge, based on their posterior probability. Equations for calculating this measure using the inside and outside probabilities are given.

Once the theoretical framework for a purely statistical parsing confidence measure has been set, we introduce experimentation showing the utility of it. First we show that the confidence measure performs notably well for the Confidence Error Rate (CER) and ROC curve metrics, which are widely used for confidence measure evaluation in ASR and MT.

Then we report experiments showing the use of the confidence measure to improve POS tagging, in a method which comprises the relabeling of edges with low confidence.

Finally, we set out the basis for other confidence measure applications, which we are currently developing. These include the introduction of Computer Aided Parsing (CAP) systems that make use of confidence measures. Such systems can be of great utility in the construction of new syntactically annotated corpora, in English or other languages. Another use is the research of new confidence measure-based parsing methods, similar to the Goodman approach.

The rest of this chapter is organized as follows. In section 1.2 we review the state-of-the-art work in syntactic parsing, mention some works that present some kind of relationship between the parsing and the confidence measure worlds, and we lay out the basics for parsing relevant for this thesis, such as the CYK algorithm. Section 1.3 includes information about how confidence measures are used in other fields, and details about their use in Automatic Speech Recognition.

1.2 Parsing

In this section we will first make a quick overview of the state-of-the-art literature of the parsing field, we will also cite some parsing works relevant to confidence measures, and then we will explain in detail some theoretical fundamentals relevant to the work presented in this thesis.

1.2.1 State-of-the-art parsing overview

Parsing plays an important role in problems like Semantic Analysis, Question Answering, Language Modeling [9, 32], Machine Translation [40, 7], RNA Modeling [36], and others. In the case of Natural Language Processing, its aim is to precisely determine the syntactic structure of sentences written in one of the several languages that humans use.

Several kinds of parsing algorithms exist, and among them one can found those based on Probabilistic Context-Free Grammars (PCFGs) [1] , which form the base of relevant and high performing parsers [6, 10, 8].

However, basic PCFGs present two problematic characteristics: context-freedom and unlexicalization. The first, results in that probabilities of rules only depend on the current nonterminal, e.g., a *VP* has the same probability of being expanded as a verb followed by a noun, independently if it acts as the main verbal phrase or as a modifier. The latter causes that nonterminal are expanded regardless of the final word they affect, e.g., VPs both for transitive and intransitive verbs have the same probabilities. These problems cause performance of vanilla treebank grammars (grammars directly derived from annotated corpus) to be insufficient, as can be seen in [5, 20].

Great efforts have been undertaken to improve performance of these parsers. Lexicalization of grammars with elaborate smoothing accomplished very promising results [6, 10]. Manual tree annotation and nonterminal splitting greatly shortened the gap between unlexicalized models and their better performing lexicalized counterparts [18, 20]. And automatic tree annotation systems, with a nonterminal split-and-merge approach and a hierarchy of progressively refined grammars, improved over the best lexicalized results [25, 29, 30]. The most impressive results are achieved by reranking systems, as shown with the semi-supervised method of [26], or the forest reranking approximation of [15] in which packed parse forests, compact structures that contain many possible tree derivations, are used for the reordering.

1.2.2 Confidence measures and parsing in the literature

In this section we explain some similarities between our confidence measure framework and some existing parsing algorithms and techniques.

[1]See section 1.2.3 for a definition of PCFGs.

Currently, there exist efficient probabilistic parsing algorithms that are able to obtain very good parsing results [25, 26, 30, 15]. Some of this parsers use reranking techniques in order to achieve these results. In this work, we propose to move forward in parsing techniques by detecting individual erroneous syntactic structures. Confidence measures can be used to detect specific erroneous constituents in a similar way as it is carried out in ASR and SMT.

Other works have proposed to improve parsing results by defining parsing algorithms that try to improve alternative objective functions. Goodman in [13] derived an algorithm that maximized the chosen evaluation criterion (labeled recall), rather than maximizing the whole tree probability, as the classical CYK does. In his derivation, he used the same posterior probability expression that we employ here in order to calculate the confidence measures.

Goodman's algorithm presented the problem of producing trees that were not grammatical, and as such, unsuitable for downstream processing. However, many applications can benefit from maximizing the number of correct constituents, regardless of the grammaticality of the tree, for example, machine translation systems. The *max-rule* parser, a variation of Goodman's algorithm that solves the ungrammaticality issue, is used in very recent top performing parsing systems [25, 30].

Smith in [37] used confidence to bootstrap feature-rich dependency parsers. In that work, confidence is measured by Rényi entropy.

Finally, confidence measures for parsing were introduced in [3]. In that work, confidence measures are computed from lists of n-best parse trees, which is the main difference from the purely statistical approach presented in this thesis.

1.2.3 Parsing fundamentals

Now that the best performing parsing works have been briefly cited, we will explain some parsing basics necessary to have a deeper understanding the work that is being presented here.

Syntactic parsing can be seen as the operation of chunking: that is, dividing input data into syntactic units of a higher level, allowing us to organize such data within a desired structure.

Parsing is a fundamental problem related to Natural Language Processing (NLP). Within this field, it refers to the the process of recognizing an input sentence and assigning it a syntactic structure or parse tree.

Figure 1.1 shows a typical parse tree. The figure displays a syntactically annotated sentence, extracted from the Penn Treebank corpus, in which the syntactics labels of its different parts can be seen. In this simple example we

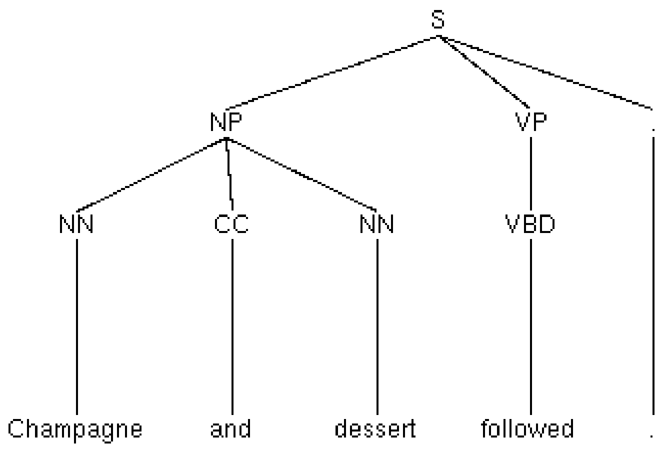

Figure 1.1: Example of a parse tree

observe that *Champagne and dessert* is the *noun phrase* (NP) that acts as the sentence subject. The subject is in turn composed by two *singular nouns* (NN) and a *coordinating conjunction* (CC). Only the *past tense verb* (VBD) *followed* forms the *verb phrase* (VP) that acts as the predicate. For more information on the syntactic labels and annotation conventions, see section 3.3 where the Penn Treebank corpus is explained.

A parse tree t is composed by substructures that are referred as edges, or constituents. An edge t^A_{ij} is defined by the nonterminal node (or syntactic tag) A that spans the substring x between positions i and j, and represents the set of all possible fitting derivations from the nonterminal to the substring. Thus, the tree is precisely determined by all its edges, and has a root edge that always spans 1 to $|x|$.

The constituents of the tree shown in Figure 1.1 are: $t^S_{1,5}$, $t^{NP}_{1,3}$, $t^{VP}_{4,4}$, $t^{.}_{5,5}$, $t^{NN}_{1,1}$, $t^{CC}_{2,2}$, $t^{NN}_{3,3}$ and $t^{VBD}_{4,4}$.

In statistical parsing, the most fitting parse tree for a given input string is calculated according to probabilistic criteria. Thus, the parse tree is obtained through a chosen parsing algorithm which uses a stochastic model.

Many parsing methods exist in the literature, which includes those based on Probabilistic Context-Free Grammars (PCFGs), a specific type of formal grammar.

A formal grammar is a precise definition of a formal language, which in turn is a set of strings over some alphabet. Formally, a grammar is a quadruple

$$G = (T, N, P, \sigma)$$

where

> T is a set of terminal symbols (the symbols of the formal language)
> N is a set of nonterminal symbols
> P is a set of derivation rules such as $\alpha \rightarrow \beta$
> with α and β being sequences of nonterminal and terminal symbols,
> α with the obligation of having at least one nonterminal,
> and β with the possibility of including the empty string ϵ
> $\sigma \in N$ is the starting symbol.

The formal language generated by G is the set

$$L(G) = \{w \in T^* | \, \sigma \Rightarrow^* w\}$$

where $\Rightarrow^*$ denotes the use of zero or more derivation rules.

Another way of defining a parse tree, apart from listing its constituents, is by the exact sequence of production rules applied. The sequence of production rules describing the tree shown in Figure 1.1, with a left-to-right depth-first order, is: $(S- > NPVP.)$, $(NP- > NNCCNN)$, $(NN- > Champagne)$, $(CC- > and)$, $(NN- > dessert)$, $(VP- > VBD)$, $(VBD- > followed)$ and $(.- > .)$.

This general definition of formal grammar is classified by the Chomsky Hierarchy in four groups of progressively stricter grammars:

Type-0 Or *unrestricted grammars*, which is the less restrictive type and include all formal grammars.

Type-1 Or *context-sensitive grammars*, which generate context-sensitive languages, in which a context of terminal and nonterminal symbols can be added to the production rules.

Type-2 Or *context-free grammars*, which generate context-free languages, in which the production rules take the form $A \rightarrow \beta$ with $A \in N$ and $\beta \in (N \cup T)^*$.

Type-3 Or *regular grammars*, which generate regular languages, in which the production rules can take the forms $A \rightarrow aB$ (or alternatively $A \rightarrow Ba$), $A \rightarrow a$ or $S \rightarrow \epsilon$, with B being a nonterminal, a a terminal, S the starting symbol (which cannot appear in the right side of rules), and ϵ the empty string.

Although type-2 and type-3 grammars are less powerful than their higher level counterparts, their simplicity allows the construction of efficient parsing algorithms, and as such they are the most commonly used types.

Related to Context-Free Grammars, the class of our interest, is the notion of Chomsky Normal Form (CNF) grammars. A CNF grammar has all its production rules in the form $A \to BC$, $A \to x$, or $S \to \epsilon$, where A, B and C are nonterminals, x is a terminal, S the starting symbol, and $B, C \neq S$.

Every context-free grammar can be transformed in an equivalent CNF grammar, and every grammar in CNF is context-free. Several ways of binarizing a grammar into a CNF equivalent are discussed in section 3.4 Quite obviously, CNF grammars can only produce binary trees, fact which makes possible simpler parsing algorithms, and which allows the introduction of powerful expressions like the *inside* and *outside probabilities*, which we will review in just a moment.

Probabilistic Context-Free Grammars are just context-free grammars in which each rule has a probability associated to it, with $pr(A \to \alpha) \in]0, 1]$ and the following added restriction

$$\sum_{\forall \alpha_j} pr(A \to \alpha_j) = 1 \quad \forall A \tag{1.1}$$

by which the probability of all the rules having the same left-hand nonterminal must sum one. More information on PCFG can be found on [23, p. 382].

The probability of a derivation t (or parse tree) for a given string x produced by the grammar G is the product of the probabilities of the applied derivation rules.

$$p_G(t, \boldsymbol{x}) = \prod pr(A \to \alpha) \in t \tag{1.2}$$

The probability of a string $\boldsymbol{x}$ being generated by the grammar is the sum of the probabilities of all possible parses of that string

$$p_G(\boldsymbol{x}) = \sum_{\forall t \in \mathcal{T}} p_G(t, \boldsymbol{x}) \tag{1.3}$$

where $\mathcal{T}$ is the set of all possible parse trees for $\boldsymbol{x}$ by G.

Expression (1.3) can in fact be easily calculated by the use of the *inside* probability, which we will now introduce. Alongside, the *outside* probability is presented, which will come in handy later on. The inside β and outside α probabilities are well known expressions, widely used in parsing for several tasks, for example, learning the stochastic information of PCFGs by maximum log-likelihood.

The inside probability

$$\beta_A(i,\, j) = p_G(A \Rightarrow^* x_i \dots x_j) \tag{1.4}$$

is the total probability of the nonterminal A generating the string $x_i \dots x_j$.

The outside probability

$$\alpha_A(i,\, j) = p_G(S \Rightarrow^* x_1 \dots x_{i-1}\, A\, x_{j+1} \dots x_{|\boldsymbol{x}|}) \tag{1.5}$$

is the total probability of, beginning with the start symbol S, generating the nonterminal A and all the words other than $x_i \dots x_j$. For more information on these, see Figure 1.2, [1], and [23, p. 392].

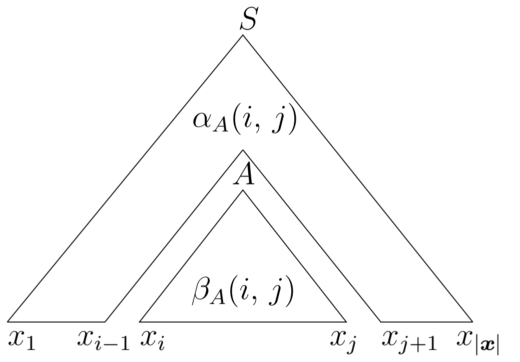

Figure 1.2: Parse diagram showing the inside and outside probability.

Here we present recursive expressions used for calculating these probabilities with grammars in the CNF

$$\beta_A(s,\, t) \begin{cases} p(A \to x_s) & s = t \\ \sum_{BC} \Big(p(A \to BC) \\ \quad \sum_{r=s}^{t-1} \Big(\beta_B(s,\, r)\beta_C(r+1, t) \Big) \Big) & s < t \end{cases} \tag{1.6}$$

$$\alpha_A(s,\, t) \begin{cases} 1 & s = 1 \wedge t = I \wedge A = S \\ \sum_{BC} \Big(p(B \to CA) \sum_{r=1}^{s-1} \Big(\alpha_B(r,\, t)\beta_C(r,\, s-1) \Big) \Big) \\ + \sum_{BC} \Big(p(B \to AC) \sum_{r=t+1}^{I} \Big(\alpha_B(s,\, r)\beta_C(t+1,\, r) \Big) \Big) \end{cases} \tag{1.7}$$

where $1 <= s, t <= |\boldsymbol{x}|$.

The probability of $\boldsymbol{x}$ being generated is the inside probability of the grammar's starting symbol S over the whole string $\boldsymbol{x}$

$$p_G(\boldsymbol{x}) = \sum_{\forall t \in \mathcal{T}} p_G(t, \boldsymbol{x}) = \beta_S(1, |\boldsymbol{x}|) \tag{1.8}$$

With all this stochastic apparatus defined, the expression for obtaining the most probable parse tree $\hat{t}$ is easily defined as follows

$$\hat{t} = \arg\max_{t \in \mathcal{T}} p_G(t, \boldsymbol{x}) = \arg\max_{t \in \mathcal{T}} p_G(t|\boldsymbol{x}) p(\boldsymbol{x}) = \arg\max_{t \in \mathcal{T}} p_G(t|\boldsymbol{x}) \tag{1.9}$$

where $p_G(t|\boldsymbol{x})$ is the conditional probability of the tree t given the string $\boldsymbol{x}$ using model G, and $\mathcal{T}$ is the set of all possible parse trees for $\boldsymbol{x}$.

1.2.4 Cocke-Younger-Kasami algorithm

In this section we will explain and present the the stochastic version of the Cocke-Younger-Kasami parsing algorithm (CYK) [14], and present its pseudocode. The classical version of the CYK algorithm just checks if a given string is generable by a grammar. A stochastic version of the CYK can be inferred which, by storing the probabilities of the partial subtrees, can find, for a given string, the most probable tree and its probability.

From the most probable tree equation in (1.9) one can derive the following recursive equations which are the basis for the CYK algorithm

$$\hat{t}(A_{i:j}) = \begin{cases} \arg\min_{t \in \mathcal{T}^1(A_{i:j})} pr(t) & j > i \\ < A_{i:i} > & \text{if } j = i \text{ and } A \rightarrow x_i \in P \end{cases} \tag{1.10}$$

where $\hat{t}(A_{i:j})$ is the most probable subtree starting from the nonterminal A and spanning $x_i \ldots x_j$ and

$$\mathcal{T}^1(A_{i:j}) = \{< A_{i:j}, T^1(B_{i:k}), T^1(C_{k+1:j}) >: A \rightarrow BC \in P, i \leq k < j\} \tag{1.11}$$

is the set containing the best possible subtrees for all rules that apply to the current nonterminal.

Starting from the leaves, the algorithm fills the trellis until the tree root field is completed. If and only if the initial symbol is there, the string is generable by the grammar, and the derivation of the most probable tree and its probability can be read from the trellis.

Here follows pseudocode for the CYK algorithm.

```
Input
    $G = (N,\ T,\ P,\ S)$ (in Chomsky Normal Form)
    $GS = (G, Pr) \qquad Pr:\ P \to ]0,1]$
                    $\sum_{1 \le j \le n_i} Pr(A_i \to \alpha_j) = 1 \qquad \forall A_i$
    $x = x_1 \dots x_n \in T^*$
Output
    Parse table $t[i,\ j] \quad (i \le i,\ j \le n)$
    Parse probability table $p[i,\ j] \quad (i \le i,\ j \le n)$
    $A \in t[i,\ j] \iff A \to^* x_i \dots x_j$
Method
    forall $i, j, A$
        $p[i,\ j][A] = MAX\_DOUBLE$
    forall $i$ in $1 \le i \le n$ do
        forall $(A \to x_i) \in P$ do
            $t[i,\ i][A] = t[i,\ i][A] = [(0,0,-)(0,0,-)]$
            $p[i,\ i][A] = p[i,\ i][A] = pr(A \to x_i)$
    forall $d$ in $1 \le d \le n-1$ do
        forall $i$ in $1 \le i \le n-d$ do
            $j = i + d$
                forall $k$ in $i \le k \le j-1$ do
                    forall $(A \to BC) \in P$ do
                        if $(B \in t[i,\ k]) \wedge (C \in t[k+1,\ j]$ then
                            $newprob = p[i,k][B] * p[k+1,j][C] * pr(A \to BC)$
                            if $newprob < p[i,\ j][A]$
                                $t[i,\ j][A] = [(i,k,B),(k+1,j,C)]$
                                $p[i,\ j][A] = newprob$
                        end if
    if $p[1,\ n][S]! = MAX\_DOUBLE$ then $x \in L(G)$ else $x \notin L(G)$
End method
```
(1.12)

The time complexity of the algorithm is is $O(|x|^3\,|P|)$ (or $\Theta((|x|^2/2)\,(|x|/3)\,|P|))$ for a tighter bound); and the the space complexity is $O(|x|^2\,|N|)$ $(\Theta((|x|^2/2)\,|N|)$.

1.3 Confidence measures

1.3.1 Introduction

Confidence measures are usually used to compute the degree of trust in some part of the output of a recognition system. In the case of ASR and SMT, confidence measures refer to the probability of individual words being correct in the output sentences. Obviously, obtaining a sentence with maximizes the global probability does not imply that all the words, individually, are the most probable ones.

Given the difficulty and importance of parsing in all of its applications [21], there exists an increasing necessity to detect erroneous syntactic structures. With the introduction of the confidence measure framework into the parsing field, this powerful formalism can be used to detect, and eventually correct, individual erroneous constituents.

Confidence measures have been successfully applied in the mentioned tasks of ASR [39, 34], SMT [38], and even Spoken Dialogue Systems [33]. However, its use remains largely unexplored in parsing, and they have several applications of great interest within this field. Assessing the correctness of the different parts of the parsing is needed for the construction of efficient Computer Assisted Parsing systems, which will be useful in the creation of gold standard treebanks for new languages. Confidence measures can also help to improve the parsing process itself, either by being used as a component of an n-best reranker, or by directly recalculating parts with low confidence.

Some confidence measures for parsing, in the form of combinations of characteristics calculated from n-best lists were proposed in [3]. Nevertheless, other alternatives can be considered, akin to graph-based methods in ASR and SMT, in which the forward and backward probabilities are used for their calculation.

Most of the cited methods for obtaining confidence measures in ASR and SMT are based on calculating the posterior probability for a specific word.

One way to estimate the posterior probability is to use lists with the n-best output sentences. In this case, intuitively, the probability of a word of being correct is calculated by counting how many times does that word appear in the same position over all the n-best sentences.

Another method of estimating the posterior probability is to use a forward-backward expression over word graphs. Word graphs can be seen as a condensation of the information contained in an n-best list. In the ideal case of a non-pruned word graph, it represents all the possible output sentences for a given input. In practice word graphs are usually pruned so they contain only information about the most probable outputs. This approach presents greater flexibility than n-best lists, as they are not limited by a predefined number of n outputs, but rather take form depending on the distribution of probability mass.

1.3.2 Confidence measures in Automatic Speech Recognition

We will now develop on the fundamentals of statistical Automatic Speech Recognition (ASR), explain word graphs acting a speech recognizer output, and describe how confidence measures are precisely calculated over them. For a deeper review on these concepts see [35].

The ASR problem can be statistically formulated [16, 17] as follows. Let $\boldsymbol{x} = \{x_1, \dots x_T\}$ be a sequence of acoustic vectors representing a spoken utterance over a lapse of time, and $\boldsymbol{w} = \{w_1, \dots w_m\}$ a sequence of m words. $P(\boldsymbol{w}|\boldsymbol{x})$ is the probability of the sequence $\boldsymbol{w}$ corresponding to the utterance represented by $\boldsymbol{x}$. The word sequence that maximizes the probability for a given acoustic vector is

$$\hat{\boldsymbol{w}} = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\boldsymbol{x}) \tag{1.13}$$

by the Bayes rule

$$= \arg\max_{\boldsymbol{w}} \frac{p(\boldsymbol{x}|\boldsymbol{w})p(\boldsymbol{w})}{p(\boldsymbol{x})} \tag{1.14}$$

where $p(\boldsymbol{x}|\boldsymbol{w})$ is the probability of observing the acoustic vectors when $\boldsymbol{w}$ is spoken, modeled by the acoustic model; $p(\boldsymbol{w})$ is the probability of the word sequence, modeled by the language model; and $p(\boldsymbol{x})$ is the prior probability of the acoustic sequence, which doesn't affect the maximization and can be omitted.

$$= \arg\max_{\boldsymbol{w}} p(\boldsymbol{x}|\boldsymbol{w})p(\boldsymbol{w}) \tag{1.15}$$

The acoustic modelling is usually approached by the use of Hidden Markov Models (HMMs) [16, 31], and the language model frequently corresponds to an n-gram model (usually 2-grams or 3-grams) [27]. As we will not develop on these concepts, refer to the cited work for further information.

Like already mentioned, a word graph can be seen as a condensation of a n-best list of recognized word sequences. A word graph can be easily produced from a HMM model, and it is a very convenient tool for calculating the probabilities of the different word sequences and, for the part most concerning us, the posterior probability of the individual words in each of the sequences [28].

Formally, a word graph $R = (Q, A, q_i, q_f, \mathcal{F})$ is a directed and weighted graph, without cycles, in which:

Q is a set of states, each state corresponds to one language model state, and was activated in the instant $t \in \{1, \dots, T\}$ of the recognition process. Each state is denoted by u^t, where u is the language model state, and t the instant.

A is a set of edges, where each edge is composed of $[w, u^\tau, v^t]$, where w is a word and $u^\tau, v^t \in Q$ are the states delimiting w in the recognition process.

$q_i = u^1 \in Q$ is the initial state, $q_f = v^T \in Q$ is the final state.

And $\mathcal{F} : QxQ \to \mathbb{R}$ is a function that assigns to each edge $[w, u^\tau, v^t]$ the probability (or, in the more general case, a score) to the word w between the states u and v, and between the instants τ and t.

Every path in the word graph starting from q_i and ending in q_f corresponds to a suggested hypothesis $h = \{(w_i, u_1^{\tau_1}, v_1^{t_1}), \dots, (w_k, u_k^{\tau_k}, v_k^T)\}$, where $t_{i-1} = \tau_i - 1 \; \forall i = 2, \dots, k$.

The probability of the hypothesized word sequence is the product of the probabilities of the involved edges. The confidence measure of a word w is its posterior probability given the acoustic sequence $\boldsymbol{x}$

$$\mathcal{C}(w|\boldsymbol{x}) = p(w|\boldsymbol{x}) \tag{1.16}$$

The posterior probability of a word that occurs between the states u^τ and v^t can be easily calculated over a word graph by summing the probabilities of all hypothesis that contain the edge, which is done using the forward and backward probabilities $[w, u^\tau, v^t]$

$$p([w, u^\tau, v^t]|\boldsymbol{x}) = \frac{p([w, u^\tau, v^t],\, \boldsymbol{x})}{p(\boldsymbol{x})} \tag{1.17}$$

$$= \frac{1}{p(\boldsymbol{x})} \sum_{h \in R : \exists [w', u', v'] : w' = w, u' = u^\tau, v' = v^t} p([w, u^\tau, v^t],\, \boldsymbol{x}) \tag{1.18}$$

The normalization term, the probability of the acoustic sequence is calculated by summing the probability of all hypothesis contained in the word graph, which can be also calculated by the use of either the forward or backward probability

$$p(\boldsymbol{x}) = \sum_h p(j, \boldsymbol{x}) \tag{1.19}$$

A problem of this approach is that the posterior probability of a word only is accumulated if the states delimiting the word u, v, and its time interval $[\tau, t]$ are exactly the same. In practice the same word usually occurs in similar but slightly different time intervals; and in different states. Hence, the posterior probability scatters among the different intervals $[\tau', t']$, and states u', v'. There are several techniques that are usually employed to avoid this problem when calculating the confidence measure, so the probability of the same words with similar intervals and different states is accumulated.

Chapter 2

A statistical confidence measure for parsing

2.1 Introduction

In this chapter we introduce our main contribution, the statistical framework for the calculation of confidence measures for each edge, using its posterior probability. For it we employ the inside and outside probabilities.

2.2 Posterior probability of an edge

As already mentioned, a tree t is composed by edges , or constituents. Given a tree t, an edge t_{ij}^{A} is defined by a nonterminal node A that spans the substring between positions i and j.

While in [3] confidence measures for parsing were calculated using n-best parse lists, here we set a framework for probabilistic calculation of confidence measures for edges consisting in their posterior probability. This purely statistical measure is similar to the posterior probability calculation over word graphs and its use as a confidence measure presented in the above commented works [39, 38].

Assume that using a chosen probabilistic grammar G as the model, the parser analyzes the input sentence $\boldsymbol{x} = x_1 \ldots x_{|x|}$ and produces the most probable parse tree

$$\hat{t} = \arg\max_{t \in \mathcal{T}} p_G(t|\boldsymbol{x}), \tag{2.1}$$

where $p_G(t|\boldsymbol{x})$ is the probability of the tree t given the string $\boldsymbol{x}$ using model G, and $\mathcal{T}$ is the set of all possible parse trees for $\boldsymbol{x}$.

The posterior probability of an edge can be considered as a measure of the degree to which the edge is believed to be correct. The posterior probability of an edge given the string $\boldsymbol{x}$ is

$$p_G(t_{ij}^A|\boldsymbol{x}) = \frac{p_G(t_{ij}^A, \boldsymbol{x})}{p_G(\boldsymbol{x})} = \frac{\sum_{t' \in \mathcal{T} : t_{ij}^A \text{ is in } t'} p_G(t'|\boldsymbol{x})}{p_G(\boldsymbol{x})} \ . \tag{2.2}$$

Expression (2.2) is the normalized probability of the edge t_{ij}^A being placed on the tree in the exact position that spans the $x_i \ldots x_j$ substring. The upper part is the sum of probabilities of all possible parse trees for $\boldsymbol{x}$ containing the nonterminal A with the same exact start and end points i and j.

Expression (2.2) can be efficiently computed with the inside and outside probabilities, which were introduced in section 1.2.3. Following [2], the equation can be rewritten as

$$p_G(t_{ij}^A|\boldsymbol{x}) = \frac{p_G(t_{ij}^A, \boldsymbol{x})}{p_G(\boldsymbol{x})} = \frac{\beta_A(i,\, j)\ \alpha_A(i,\, j)}{\beta_S(1,\, |\boldsymbol{x}|)} \ . \tag{2.3}$$

The posterior probability can now directly be used as a measure of the confidence in each individual edge

$$\mathcal{C}(t_{ij}^A) = p_G(t_{ij}^A|\boldsymbol{x}) \ . \tag{2.4}$$

Expression (2.3) is the same that is maximized in [13] for the labeled recall parsing algorithm, which can indeed be seen as a confidence measure based parsing algorithm.

Figure 2.1 shows a synthetic example in order to clarify these definitions. The figure shows the only four possible parse trees for the string *abc*: *(a)*, *(b)*, *(c)* and *(d)*. Let all productions in the example's grammar carry the same probability, and suppose that the parser returns the *(a)* tree. Then the following confidence measure values are obtained for the edges in the *(a)* tree: $\mathcal{C}(t_{13}^S) = 1$, $\mathcal{C}(t_{12}^Z) = 2/4$, $\mathcal{C}(t_{11}^A) = 1$, $\mathcal{C}(t_{22}^B) = 1$ and $\mathcal{C}(t_{33}^D) = 1/4$. If the correct parse tree is *(e)*, which is unobtainable by the example grammar, the use of a confidence threshold would allow us to know that the t_{33}^D edge is incorrect in the *(a)* tree.

It should be clear that the calculation of confidence measures for parsing can be useful in any kind of problem which uses probabilistic parsing, and not just NLP tasks. In the experiments presented in the following chapters we show that our confidence measure can help parsing through the detection of erroneous edges.

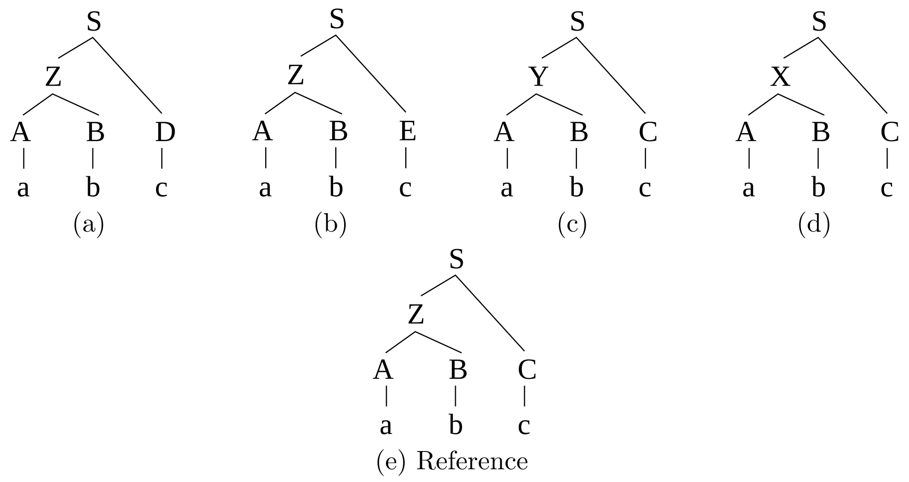

Figure 2.1: Synthetic example of confidence measure calculation. All productions in the example's grammar have the same probability. The grammar can only generate the *(a), (b), (c)* and *(d)* parse trees for the *abc* input string. The reference parse tree is unobtainable. Confidence measures for the edges in the (a) tree are $\mathcal{C}(t_{13}^{S}) = 1$, $\mathcal{C}(t_{12}^{Z}) = 2/4$, $\mathcal{C}(t_{11}^{A}) = 1$, $\mathcal{C}(t_{22}^{B}) = 1$ and $\mathcal{C}(t_{33}^{D}) = 1/4$.

Chapter 3

Confidence measure benchmarking

3.1 Introduction

In the previous chapter we introduced a purely statistical framework for computing confidence measures in parsing. In this chapter introduce it to basic and reproducible CYK parsing setup, similar to the one presented in [20].

This chapter is divided as follows: section 3.2 describes how confidence measures can be used to discern incorrect edges, and presents several evaluation metrics that are usually used to assert their performance; the Penn Treebank corpus is described in section 3.3; the experimental setup is explained in section 3.4; and the obtained results are presented and discussed in sections 3.5 and 3.6.

3.2 Evaluation Metrics

Given a parse tree that is obtained for an input string, the tree contains a number N of edges which can then be checked against a reference or gold tree, to find that a number N_c of them have a correct label and span, and a number N_i of them have not. Optimally, confidence measures allow us to precisely determine which of these constituents are correct and which are not.

Once incorrect edges are detected, some action to correct them can be carried out. In this work we introduce experimentation showing the recalculation of incorrect POS tags in chapter 4.

A threshold τ ($\tau \in [0, 1]$) can be set in the use of confidence measures: tags with a confidence value lower than the threshold will be deemed incorrect, and vice versa. After applying this threshold to a confidence measure, we obtain a number $N_f(\tau)$ ($N_f(\tau) \in [0, N_c]$) of edges correctly labeled by the parser, but deemed incorrect by the confidence measure (false rejection); as well as a number $N_t(\tau)$ ($N_t(\tau) \in [0, N_i]$) of edges incorrectly labeled, and determined incorrect by their confidence (true rejection). In the ideal case of a perfect confidence measure, $N_f(\tau) = 0$ and $N_t(\tau) = N_i$ for the best threshold τ. Obviously, if we set $\tau = 0$, all edges are considered to be correct, so $N_f(0) = 0$ but $N_t(0) = 0$ too (likewise, $N_f(1) = N_c$ and $N_t(1) = N_i$).

For our evaluation we employed some commonly used metrics that determine whether confidence measures successfully discern correct labels from incorrect ones. One measure is the Confidence Error Rate (CER), which is the number of errors (false rejections plus false acceptances) performed by the confidence measures divided by the total number of edges. The CER is calculated for a given threshold, and it is computed as:

$$CER(\tau) = \frac{N_f(\tau) + (N_i - N_t(\tau))}{N_c + N_i} \tag{3.1}$$

The baseline CER is the one obtained assuming that all syntactic edges are correct (the only possible assumption when confidence measures are not available) and is:

$$CER(0) = \frac{N_i}{N_c + N_i} \tag{3.2}$$

Two rates are directly derived from the true rejection and the false rejection values presented above: the False Rejection Rate (FRR): $R_f(\tau) = \frac{N_f(\tau)}{N_c}$ and the True Rejection Rate (TRR): $R_t(\tau) = \frac{N_t(\tau)}{N_i}$.

Another measure which determines the correctness of confidence measures globally over all possible thresholds is the Receiver Operating Characteristic [1] (ROC) curve [11] which is the visual representation of the false rejection rate against the true rejection rate for all possible values of $\tau \in [0, 1]$.

The worst case ROC is a diagonal line, and a good ROC is the one that, for most thresholds τ, describes a curve with values near 0 for N_f and near 1 for N_i. The AROC (area under a ROC curve divided by the area of the worst-case diagonal ROC) varies between 1.0 and 2.0, and provides and adequate overall estimation of the confidence measure's accuracy. See [39, 38] for examples of evaluating confidence measures with ROC curves.

[1] A variant of the ROC curve is the Detection Error Tradeoff (DET) curve which plots the False Rejection Rate versus the False Acceptance Rate

3.3 The corpus

The creation of large and high quality syntactically annotated corpora is a difficult task. The manual annotation of large amount of text not only is a laborious and time-consuming job, but usually there are several linguists involved, so a list of predefined annotation conventions and guidelines have to be enforced.

ADJP	Adjective phrase
ADVP	Adverb phrase
NP	Noun phrase
PP	Prepositional phrase
S	Simple declarative clause
SBAR	Clause introduced by subordinating conjunction or 0
SBARQ	Direct question introduced by wh-word or wh-phrase
SINV	Declarative sentence with subject-aux inversion
SQ	Subconstituent of SBARQ excluding wh-word or wh-phrase
VP	Verb phrase
WHADVP	wh-adverb phrase
WHNP	wh-noun phrase
WHPP	wh-prepositional phrase
X	Constituent of unknown or uncertain category

Table 3.1: Penn Treebank labels employed for skeletal annotation.

The Penn Treebank (PTB) is one of such high quality annotated corpora, and has become the reference corpus to experiment with when benchmarking parsing systems, widely used in the most relevant parsing literature. It is a project [1] of the LINC Laboratory of the Computer and Information Science Department at the University of Pennsylvania. All data produced by the Treebank is released through the Linguistic Data Consortium [2]. It consists in one million words of 1989 Wall Street Journal material annotated. More details about how the sentences are annotated can be found in [24].

Basically, there are two sets of nonterminal symbols used in the annotations: the preterminals, or POS tags, which are the labels immediately preceding the terminal words; and the syntactic tags, which are the nonterminals which form the skeletal structure of the sentences.

The 14 employed skeletal labels are shown in table 3.1, and 48 annotated POS tags are shown in table 3.2.

[1]http://www.cis.upenn.edu/ treebank/
[2]http://www.ldc.upenn.edu/

CC	Coordinating conjunction	TO	to
CD	Cardinal number	UH	Interjection
DT	Determiner	VB	Verb, base form
EX	Existential there	VBD	Verb, past tense
FW	Foreign word	VBG	Verb, gerund/pres. participle
IN	Preposition/subordinating conj.	VBN	Verb, past participle
JJ	Adjective	VBP	Verb, non-3rd ps. sing. pres.
JJR	Adjective, comparative	VBZ	Verb, 3rd ps. sing. present
JJS	Adjective, superlative	WDT	wh-determiner
LS	List item marker	WP	wh-pronoun
MD	Modal	WP$	Possessive wh-pronoun
NN	Noun, singular or mass	WRB	wh-adverb
NNS	Noun, plural	#	Pound sign
NNP	Proper noun, singular	$	Dollar sign
NNPS	Proper noun, plural	.	Sentence-final punctuation
PDT	Predeterminer	,	Comma
POS	Possessive ending	:	Colon, semi-colon
PRP	Personal pronoun	(	Left bracket character
PP$	Possessive pronoun	)	Right bracket character
RB	Adverb	”	Straight double quote
RBR	Adverb, comparative	’	Left open single quote
RBS	Adverb, superlative	”	Left open double quote
RP	Particle	’	Right close single quote
SYM	Symbol (mathematics or science)	”	Right close double quote

Table 3.2: Penn Treebank POS tagset.

3.4 Experimental setup

For our experiments three sets were defined, using several sections of the PTB corpus. For the training set we chose sections 2 to 21 of the PTB, which were directly used to obtain a vanilla Penn Treebank Grammar.

The development set comprised the first 346 sentences of section 24, and the test set was the whole section 23 of the PTB. The chosen test and train sets facilitate comparison with previous work.

Before obtaining the grammar, we carried out the *NoEmpties* transformation in all sets. As described in [19], it comprises the removal of functional tags and cross-referencing annotations, plus the pruning of empty branches (represented by the -NONE- tag on the corpus). Additionally, we followed the common practice of substituting all cardinals appearing in the sentences (leafs below the *CD* label) by a newly introduced common terminal.

The CYK algorithm needs grammars to be in the Chomsky Normal Form (CNF), so we obtained several binarized versions of the train grammar. Binarization splits the non-binary rules (those with three or more siblings) introducing dummy nonterminals between the parent and the children. This transformation does not alter the probabilistic correctness of the parsing process.

For calculating the binarized grammars, we used the Chomsky Normal Form transformation method from the open source Natural Language Toolkit (NLTK) [22] to obtain several right-factored binary grammars of different sizes. This method implements the vertical (v value) and horizontal (h value) markovizations discussed in [20]. Setting these two parameters we can control how many ancestors and siblings are annotated in the newly introduced nonterminals. A vertical value of $v = 1$ means no ancestor information, $v = 2$ adds parent information, $v = 3$ grandparent, and so on. With $h = \text{inf}$ all siblings are annotated, $h = 0$ annotates none of them, $h = 1$ one, and so on. The binarization equivalent to a standard treebank PCFG grammar corresponds to $v = 1$ and $h = \infty$ [20].

At parsing time we performed the confidence measure calculation using equations (2.2) and (2.3) as described in chapter 2.

The obtained parse trees were binary, as they were produced by PCFGs in CNF, so in order to compare them to the reference trees a trivial unbinarization process was performed. In this process newly introduced nonterminals are removed and their children go up with their original parents.

The edges of each solution tree were then automatically compared to the edges in the reference ones. For each edge in the solution tree, it was labeled as correct if an edge with the same tag and span was present in the corresponding reference tree, otherwise it was labeled as incorrect.

With the edges labeled as correct or not, we calculated the baseline CER and the confidence measure CER. As the CER depends on the selected threshold, the separate development set was used to obtain the best threshold, and this value was used to calculate the CER within the test set. Additionally, ROC curves for the development and test sets are presented, with their corresponding AROC values.

3.5 CER and ROC results

In this section we present the results of the proposed confidence measure for several CNF PCFGs, which were obtained modifying the vertical and horizontal order of grammar markovization. Goodness of confidence measures can be evaluated measuring the improvements of the best CER over the baseline CER, and additionally with the ROC curve and its AROC.

		Edges	
PCFG	(size)	Dev	Test
h=0, v=1	(561)	9,385	60,927
h=0, v=2	(2,034)	9,384	61,107
h=0, v=3	(5,058)	9,381	61,123
h=1, v=1	(2,174)	9,336	60,630
h=1, v=2	(5,434)	9,359	60,821
h=1, v=3	(11,420)	9,367	60,955

Table 3.3: Number of edges obtained for the development and test sets (Dev and Test columns) for each PCFG. Parameters h and v are respectively the horizontal and vertical markovization order, as discussed in [20]. Grammar size represents the number of nonterminals in each PCFG after binarization.

In these first experiments we did not use smoothing, so not all sentences were successfully parsed by our PTB grammar. For the development set 219 out of 346 sentences were parsed; and for the test set they were 1581 out of 2416 sentences. Despite only the parsed sentences were taken into account, this did not cause a significant bias (see section 4.4).

		Baseline	Confidence M.	
PCFG	(size)	CER	CER	AROC
h=0, v=1	(561)	16.4	12.3	1.78
h=0, v=2	(2,034)	15.0	12.6	1.74
h=0, v=3	(5,058)	15.1	13.1	1.72
h=1, v=1	(2,174)	12.6	10.4	1.74
h=1, v=2	(5,434)	11.9	10.8	1.72
h=1, v=3	(11,420)	12.1	11.1	1.70

Table 3.4: Results for the test set: Baseline CER, CER obtained using the best development threshold and AROC, for each PCFG. Parameters h and v are the vertical and horizontal markovization order. Grammar size is the number of nonterminals.

The use of the different CNF transformations of the grammar, generated by different markovization parameters, did not alter the number of successfully parsed phrases, given that the PCFG were all derived from the same original one. However, the different markovizations caused changes in the obtained parse trees, which could vary in the number of generated edges. Table 3.3 shows the total amount of edges obtained in each experiment.

The resulting CER and AROC values for the test set are shown in Table 3.4. Confidence measures allowed us to clearly improve the baseline CER for all PCFG. Notable improvements were obtained for PCFGs with worse baseline CERs. The obtained relative reductions range from 8.2% to 25%.

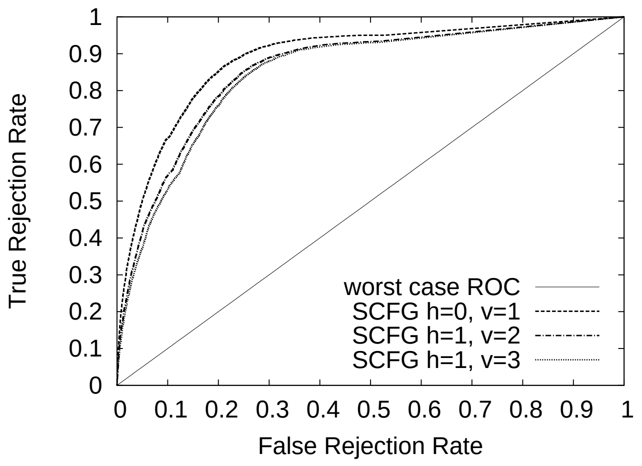

Figure 3.1: ROC curves of the test set for the most representative PCFGs.

Even for the PCFG with the best baseline CER ($h = 1$, $v = 2$), confidence measures allowed us to detect that 1.1% of the edges could be erroneously labeled, a relative reduction of 9.2%. We see that the best performing grammar corresponded to markovization orders $h = 1$ and $v = 2$, which is along the line of the results reported in [20, table on page 3].

ROC curves for the most representative PCFGs are presented in Figure 3.1. This figure shows the grammar with the highest relative gain ($h = 0$, $v = 1$), the one with the lowest relative gain ($h = 1$, $v = 3$) [1], and the one with the lowest baseline CER. We can see that the ROC curves and its corresponding AROC values were reasonably good.

Our results can be compared by the ones presented in [3], in which confidence measures were calculated from n-best lists obtained by the Charniak parser. In this work we carried out unlexicalized parsing, therefore the obtained baseline CERs are worse than the ones reported in the cited paper. This higher baseline error rate could justify the notably higher CER relative improvements and the better AROC values.

[1]All other ROC curves, and their AROCs, lie between those two.

3.6 Posterior probability accumulation for overlapping edges

As already mentioned in section 1.3.2, when calculating confidence measures in the ASR world, the posterior probability of a word is accumulated for similar (bot not exactly alike) edges to avoid dispersion. In ASR, word graph edges referring exactly to the same word can have slightly different start and end instants, so some techniques are used to detect this and sum the probability mass of the different edges.

Borrowing from this idea and applying it to parsing, we implemented and performed some experimentation in which the confidence probabilities of closely overlapping tree edges were accumulated.

We modified the confidence measure expression as follows

$$\begin{aligned}\mathcal{C}(t_{ij}^{A}) = \\ &\Big(\beta_A(i,\, j)\alpha_A(i,\, j) \\ &\sum_{r=1}^{k} \beta_A(i-r,\, j)\alpha_A(i-r,\, j) \\ &\sum_{s=1}^{k} \beta_A(i,\, j+s)\alpha_A(i,\, j+s) \\ &\sum_{r=1}^{k}\sum_{s=1}^{k} \beta_A(i-r,\, j+s)\alpha_A(i-r,\, j+s)\Big) \\ &/\beta_S(1,\, n)\end{aligned} \tag{3.3}$$

which just accumulates the probabilities of edges with slightly different start and end points. The proximity factor k control how far are the accumulated edges.

For example, setting $k = 1$ produces the following expression

$$\begin{aligned}\mathcal{C}(A_i^j) = \\ &\Big(\beta_A(i,\, j)\alpha_A(i,\, j) + \\ &\quad \beta_A(i,\, j+1)\alpha_A(i,\, j+1) + \\ &\quad \beta_A(i-1,\, j)\alpha_A(i-1,\, j) + \\ &\quad \beta_A(i-1,\, j+1)\alpha_A(i-1,\, j+1)\Big) \\ &/\beta_S(1,\, n)\end{aligned} \tag{3.4}$$

The obtained results, shown in table 3.5, were disappointing. Almost in all cases, the CER and ROC worsen when probability from overlapping edges is accumulated.

		Baseline	Confidence M.		Overl. k=1.		Overl. k=2	
PCFG	(size)	CER	CER	AROC	CER	AROC	CER	AROC
h=0, v=1	(561)	17.1	12.1	1.81	12.0	1.80	12.4	1.76
h=0, v=2	(2,034)	14.7	12.4	1.76	12.5	1.75	12.4	1.75

Table 3.5: Results with overlapping for the dev set: Baseline CER, CER obtained using the best development threshold and AROC, for each PCFG and overlapping parameter.

3.7 Concluding remarks

In this chapter we have introduced evaluation metrics to assess the goodness of confidence measures, and have explained our experimentation setup. Results show promising improvement in CER rates, and good ROC curves. The idea of accumulating confidence probability for overlapping edges was introduced, but did not produce improvements.

Chapter 4

Confidence measures for POS tag relabeling

4.1 Introduction

Once the evaluation metrics of the previous chapter have assessed that the presented confidence measure has potential, it is necessary to test it in a real parsing specific problem.

In this chapter we illustrate the use of our confidence measure in such an application. We employed confidence measures to improve the accuracy of POS tagging, using some of the grammars discussed in section 3.4. The chapter organization is detailed here. Section 4.2 presents the experimental setup and its results. In our experiments we relabeled POS tags when there existed a candidate with a higher confidence value. In section 4.3 experimentation with thresholds to avoid relabeling of tags with high confidence is introduced. Finally, section 4.4 presents the rerun some experiments from section 4.2 after implementing a basic smoothing technique, to assess the significance of the bias introduced by the lack of smoothing in previous experiments.

4.2 POS tag relabeling

The experiment consisted in the relabeling of POS tags: substitution of POS tags with low confidence by another POS tag which yielded a higher confidence when placed at the same position.

We modified our parser so the following process was carried out after the best parse tree was obtained. For each POS tag (pre-terminal nods) in the parse tree, the confidence of all other nonterminals placed in the same exact position was calculated. The nonterminal yielding the maximum confidence was introduced as the new POS tag.

The proposed method did not produce a big number of label substitutions, as usually labels with high confidence already have the maximum confidence value among all nonterminals. Approximately 1% of the 30,057 POS tags in the test set were replaced in the relabeling process.

It should be noted that the proposed relabeling procedure generated trees that were not compatible with the grammar employed in the parsing process, and thus are not suitable for downstream processing ([13] also showed this issue). Likewise, our goal was not obtaining grammatical trees, but maximizing the number of correct constituents.

		Baseline	Relabeling	
PCFG	(size)	tag acc.	tag acc.	improvement
h=0, v=1	(561)	95.62	95.78	0.16 ± 0.14
h=0, v=2	(2,034)	95.83	96.11	0.29 ± 0.11
h=0, v=3	(5,058)	95.73	96.10	0.34 ± 0.11

Table 4.1: POS relabeling results for the test set: baseline system tag accuracy, relabeling system tag accuracy, and increment in accuracy. Accuracy values are bootstrap estimates with $B = 10^4$, and improvement interval is a 95% confidence interval based on the standard error estimate [4].

Table 4.1 shows obtained results for the test set. The accuracy of the POS tagging process already was around 95% in the baseline system, so the improvements provided by the relabeling system are small. Because of this, the bootstrapping techniques presented in [4] were used to ensure statistical significance: bootstrap estimates using $B = 10^4$ were obtained for the tag accuracy on both systems and for their differential. The 95% confidence interval for the differential between the two systems was also calculated based on the standard error estimate.

We carried out some additional experimentation trying to replace all the tree edges, instead of just the POS tags: we performed some precision and recall calculations but these metrics did not improve. We believe this was caused because our relabeling approach only dealt with erroneous tags, but not with the possibly more severe structural errors within the bracketing end and start points.

4.3 Relabeling thresholds

Rather then allowing the substitution of all POS tags, in this section we explore limiting the relabeling process by introducing several kinds of thresholds, so only labels whose confidence does not fulfill the chosen criterion are considered for substitution.

We implemented and performed additional experiments using three different kinds of thresholds, that avoid the relabeling of POS tags with already high confidence values. Let the edge candidate being considered for substitution be t', the thresholds work as follows

o, a *fixed threshold.* Using this setting, only tags with a confidence value $\mathcal{C}(t') < o$ were considered for substitution.

p, a *relative proportional threshold.* This threshold takes into account the confidence score of the second highest candidate $\mathcal{C}(t'')$ for that position. If $\frac{\mathcal{C}(t')}{\mathcal{C}(t'')} < p$, then t' is considered for substitution.

q, a *relative absolute threshold.* This threshold is also relative to the score of the second highest candidate. If $\mathcal{C}(t') - \mathcal{C}(t'') < q$, then t' is considered for substitution.

Unfortunately, these experiments did not improve the obtained results. We believe that this is due to the already small number of effective substitutions without threshold, as usually candidates with a high confidence value are not substituted in the first place. The results can be seen on tables 4.2, 4.3 and 4.4. We observe in all cases that the results worsen as the threshold increments.

	o threshold		
baseline	**95.75**		
value	0.75	0.50	0.25
accuracy	95.75	96.19	95.84

Table 4.2: Tag accuracy for the development set, for each threshold, with PCFG h=0, v=1: baseline system tag accuracy, fixed threshold o, relative proportional threshold p, and relative absolute threshold q.

4.4 Smoothing

Given the high number of rejected sentences due to words not being found in the trained grammar, we evaluated whether some kind of bias could be interfering with our results. In order to reran some of the experiments with the full test corpus, we implemented a very basic smoothing method: when an input word could not be derived by any of the preterminals in the grammar,

	p threshold						
baseline	**95.75**						
value	1.1	1.2	1.3	1.4	1.5	1.6	1.7
accuracy	95.75	95.64	95.62	95.58	95.54	95.54	95.58
value	1.8	1.9	2.0	2.1	2.2	2.3	2.4
accuracy	95.54	95.54	95.54	95.49	95.52	95.47	95.47
value	2.5	2.6	2.7	2.8	2.9	3.0	3.1
accuracy	95.47	95.45	95.43	95.39	95.37	95.37	95.37
value	3.2	3.3	3.4	3.5	3.6	3.7	3.8
accuracy	95.37	95.37	95.35	95.35	95.35	95.32	95.32
value	3.9	4.0	4.1	4.2	4.3	4.4	4.5
accuracy	95.32	95.32	95.32	95.32	95.32	95.32	95.32
value	4.6	4.7	4.8	4.9	5.0	5.1	5.2
accuracy	95.32	95.32	95.32	95.32	95.32	95.30	95.30

Table 4.3: Tag accuracy for the development set, for each threshold, with PCFG h=0, v=1: baseline system tag accuracy, fixed threshold o, relative proportional threshold p, and relative absolute threshold q.

a very small probability for that word was uniformly added to all of the preterminals.

Tables 4.5 and 4.6 are the smoothed equivalents to Tables 3.4 and 4.1. The smoothed results showed very similar gains to the unsmoothed ones, fact that confirm the bias did not cause a very disturbing effect within the results.

4.5 Concluding remarks

In this chapter we illustrated the use of our confidence measure in a real parsing application: improving the accuracy of POS tagging by relabeling edges with low confidence.

The results shown small but significative improvements for all tested PCFGs. We introduced experimentation with thresholds to avoid relabeling of tags with high confidence. Unfortunately, no improvements were observed by these changes. Finally, results using a basic smoothing technique were shown, which demonstrate that the bias introduced by the lack of smoothing in the previous experiments was not significant.

	q threshold						
baseline	**95.75**						
value		0.05	0.10	0.15	0.20	0.25	0.30
accuracy		95.73	95.66	95.60	95.56	95.52	95.56
value	0.35	0.40	0.45	0.50	0.55	0.60	0.65
accuracy	95.49	95.49	95.43	95.37	95.32	95.32	95.32
value	0.70	0.75	0.80	0.85	0.90	0.95	1.00
accuracy	95.30	95.30	95.30	95.30	95.30	95.28	95.28

Table 4.4: Tag accuracy for the development set, for each threshold, with PCFG h=0, v=1: baseline system tag accuracy, fixed threshold o, relative proportional threshold p, and relative absolute threshold q.

		Baseline	Confidence M.	
PCFG	(size)	CER	CER	AROC
h=0, v=1	(561)	18.1	12.8	1.79
h=0, v=2	(2,034)	16.7	12.8	1.76

Table 4.5: Smoothed results for the test set: Baseline CER, CER obtained using the best development threshold and AROC, for each PCFG. Parameters h and v are the vertical and horizontal markovization order. Grammar size is the number of nonterminals.

		Baseline	Relabeling	
PCFG	(size)	tag acc.	tag acc.	improvement
h=0, v=1	(561)	94.82	95.03	0.21 ± 0.10
h=0, v=2	(2,034)	95.03	95.37	0.34 ± 0.10

Table 4.6: Smoothed results for the test set: baseline system tag accuracy, relabeling system tag accuracy, and increment in accuracy. Accuracy values are bootstrap estimates with $B = 10^4$, and improvement interval is a 95% confidence interval based on the standard error estimate [4].

Chapter 5

Other applications

5.1 Introduction

In this chapter we lay out other applications of confidence measures in parsing which at this stage we have in an early state of development. Section 5.2 explains a parsing algorithm which uses the confidence measure equation as the objective function, and section 5.3 deals with computer aided parsing.

5.2 Confidence based parsing

In this section we present a proposal for a confidence based parsing algorithm, in which we choose the derivation rule and cut point that maximizes the posterior probability of the current constituent, from the root to the leaves. Maximizing the posterior probability of the earlier and broader edges, will lead us to the final small edges that are also correct.

Our approach is similar to the one presented by Goodman in [13], but here it has been explicitly reformulated to fit our confidence measure framework.

We need a new definition of the inside probability, with the cut point and children specified, which we will call the *kBC-inside probability.*

The kBC-inside probability

$$\beta_A(i, j, k, B, C) = p_G(A \Rightarrow BC, B \Rightarrow^* x_i \dots x_k, C \Rightarrow^* x_{k+1} \dots x_j) \qquad (5.1)$$

is the total probability of the nonterminal A generating the string $x_i \dots x_j$, using the rule $A \rightarrow BC$ and the cut point k.

The recursive equation show below relies on the classical inside formulation presented in section 1.2.3

$$\beta_A(s,\,t,\,k,\,B,\,C)\begin{cases} p(A\rightarrow x_s) & s=t=k \\ p(A\rightarrow BC)\Big(\beta_B(s,\,k) \\ \quad \beta_C(k+1,t)\Big) & s<t, s\leq k\leq t \end{cases} \tag{5.2}$$

The kBC-inside probability can then be used calculate the kBC-confidence measure, which is the posterior probability associated to a constituent, with its first used production rule and the cut point (rather than just the constituent alone).

For a given parse tree, we propose the following kBC-confidence measure for each of its subtrees and cut point, the root being the nonterminal A, and the leafs $x_i \ldots x_j$, with the cut point of the $A \rightarrow BC$ production rule being k

$$\mathcal{C}_{k,B,C}(t^A_{ij}) = \frac{\beta_A(i,\,j,\,k,\,B,\,C)\alpha_A(i,\,j)}{\beta_S(1,\,N)} \tag{5.3}$$

Finally, we can obtain the tree that maximizes the confidence by recursively using the following expression, starting from the root of the tree until the leaves are reached

$$\hat{t^A_{ij}} = \ \arg\max_{k,B,C} \mathcal{C}_{k,B,C}(t^A_{ij}) \tag{5.4}$$

5.2.1 Pseudocode for the algorithm

Here we present the pseudocode for the confidence based parsing algorithm. The first step is basically the CYK algorithm with the inside probability calculations added. The second step performs the outside probability, kBC-inside, and kBC-confidence measure calculations, and finds the tree by maximizing over the kBC-confidence trellis.

The running time is similar to the CYK algorithm, but the memory footprint requirements are much higher, as the kBC-inside and kBC-confidence tables can be quite big.

First step: Precalculating the inside tables

```
Input
    $G = (N, T, P, S)$ (in Chomsky Normal Form)
    $GS = (G, Pr) \qquad Pr : P \to ]0, 1]$
        $\sum_{1 \le j \le n_i} Pr(A_i \to \alpha_j) = 1 \qquad \forall A_i$
    $x = x_1 \ldots x_n \in T^*$
Output
    Parse probability table $p[i, j] \quad (i \le i, j \le n)$
        $p[i, j][A]! = NULL \iff A \to^* x_i \ldots x_j$
    Inside table $e[i, j][A]$
    kBC-inside table $e[i, j][A][k][B, C]$
Method
    forall $i$ in $1 \le i \le n$ do
        forall $(A \to x_i) \in P$ do
            $p[i, i][A] = p[i, i][A][0, 0][0, 0][-][0, 0][-] = pr(A \to x_i)$
            $e(A, < i, i, i, -, - >) = pr(A \to x_i)$
            $e(A < i, i >) = pr(A \to x_i)$
    forall $d$ in $1 \le d \le n - 1$ do
        forall $i$ in $1 \le i \le n - d$ do
            $j = i + d$
                forall $k$ in $i \le k \le j - 1$ do
                    forall $(A \to BC) \in P$ do
                        if $(p[i, k][B]! = NULL) \wedge (p[k + 1, j][C]! = NULL)$ then
                                Calculate $e(A, < i, j >)$
                                Calculate $e(A, < i, j, k, B, C >)$
                                $p[i, j][A] = pr(A \to BC)$
                                $p[i, j][A][k][B][C] = pr(A \to BC)$
                                $//p[i, j][A][i, k][B][k + 1, j][C] = pr(A \to BC)$
                        end if
    if $p[1, n][S]! = NULL$ then $x \in L(G)$ else $x \notin L(G)$
End method
```

(5.5)

Last step: calculating the best parse tree

```
Input
    $G = (N, T, P, S)$ (in Chomsky Normal Form)
    $GS = (G, Pr) \qquad Pr : P \to ]0,1]$
        $\sum_{1 \le j \le n_i} Pr(A_i \to \alpha_j) = 1 \qquad \forall A_i$
    $x = x_1 \dots x_n \in T^*$
Output
    Parse table $t[i, j] \quad (i \le i, j \le n)$
    Parse probability table $p[i, j] \quad (i \le i, j \le n)$
    Outside table $f[i, j][A] \quad (i \le i, j \le n)$
    kBC-Confidence table $c[i, j][k][B, C] \quad (i \le i, j \le n)$
    $A \in t[i, j] \iff A \to^* x_i \dots x_j$
Method
    Variables
        Stack unexplored
    unexplored.push_back($[(1, n, S)]$)
    while unexplored.size() $> 0$ do
        $(i, j, A)$ = unexplored.pop()
        Calculate $f(A, < i, j >)$
        if $(i! = j)$
            forall $k : i < k < j;\ B, C : A \to BC$
                $c[i, j][A][k][B, C] = e(A < i, j, k, B, C >) * f(A < i, j >)/e(S, < 1, n >)$
            $t[i, j][A] = \arg\max_{k,B,C} c[i, j][A][k][B, C]$
            unexplored.push_back($[(i, k, B)]$)
            unexplored.push_back($[(k + 1, j, C)]$)
End method
                                                                    (5.6)
```

5.3 Computer aided parsing

Tasks like the generation of newly annotated corpora, for English or other languages, can greatly benefit from Computer Aided Parsing systems.

Here we present the preliminary basis for a theoretical framework needed in the development of a Computer Aided Parsing system. The search function could be performed using the confidence based parsing algorithm presented in the previous section. Additionally, confidence measures can play the role of helping the user to visually identify edges that are likely to be incorrect.

Assume that using a chosen probabilistic grammar G as the model, the parser analyzes the input sentence $\boldsymbol{x} = x_1 \ldots x_{|x|}$ and produces the parse tree $\hat{t}$

$$\hat{t} = \arg\max_{t \in \mathcal{T}} p_G(t|\boldsymbol{x}) \tag{5.7}$$

where $p_G(t|\boldsymbol{x})$ is the probability of parse tree t given the input string $\boldsymbol{x}$ using model G, and $\mathcal{T}$ is the set of all possible parse trees for $\boldsymbol{x}$.

In an interactive scenario, after obtaining the (incorrect) best tree $\hat{t}$, then the user modifies the incorrect edge t_{ij}^{A}. When he does that, he chooses the correct label for the span ij, and at the same time is validating all the edges contained therein (t_{mn}^{B} , $m >= i$, $n <= j$). All these spans are then added to the *correct edge set* $t_{\mathcal{C}}$ because of this, the user must start working from the leafs to the root. When the user modifies one edge, the parser provides the best tree it can find for the *non-yet-accepted edge set* $t_{\mathcal{N}}$.

$$\begin{aligned}
\hat{t_{\mathcal{N}}} &= \arg\max_{t_{\mathcal{N}} \in \mathcal{T}} p_G(t_{\mathcal{N}}|\boldsymbol{x}, t_{\mathcal{C}}) \\
&= \arg\max_{t_{\mathcal{N}} \in \mathcal{T}} \frac{p_G(t_{\mathcal{N}},\, t_{\mathcal{C}}|\boldsymbol{x})}{p_G(t_{\mathcal{C}})} \\
&= \arg\max_{t_{\mathcal{N}} \in \mathcal{T}} p_G(t_{\mathcal{N}},\, t_{\mathcal{C}}|\boldsymbol{x})
\end{aligned} \tag{5.8}$$

Since $t_{\mathcal{N}} t_{\mathcal{P}} = t$, this equation is very similar to equation (1.9), with the difference that now the arg max search is performed over the *non-yet-accepted edge set* $t_{\mathcal{C}}$ instead over the complete edge set t. Thus, we can use the same models, but modify the dynamic programming search part of the parsing algorithms appropriately.

Chapter 6

Concluding remarks

In this thesis, a new formal framework for calculating a purely statistical confidence measure for probabilistic parsing has been introduced. Expressions were provided to estimate the confidence based on the posterior probability, using the inside and outside probabilities.

Results were obtained for the Penn Treebank corpus. Comparison of CER values obtained using confidence measures with the baseline showed that the proposed confidence measure discriminated correct edges from incorrect ones, confirmed by similarly good AROC values. The advantage of the confidence measure was notable in all PCFGs experimented with.

We introduced the idea of posterior probability accumulation for overlapping edges, inspired in the accumulation of similar time intervals for confidence measures in Automatic Speech Recognition. However, no improvements were obtained by this addition.

We also reported a real word experiment in which POS tagging was improved using our confidence measure for relabeling edges with low confidence. Statistical significant improvements were consistently obtained by our relabeling system. These experiments proved that the proposed method is well-suited for edge confidence estimation, and that confidence measures are useful in parsing related real tasks. We performed additional experiments using smoothing, that showed similar results to the unsmoothed experiments, fact that confirms the bias owing to not using smoothing is small.

The work presented in this thesis has been submitted to the EACL '09, and is pending notification of acceptance.

Another applications of confidence measures such as confidence based parsing, and computer aided parsing which are being worked on, were introduced.

Possible future lines of work deal with the research of methods for applying confidence measures to improve state-of-the-art parsing systems, such as reranker systems; and the use of them in the aid of other parsing related tasks, like coreference resolution.

Bibliography

[1] Baker, J. 1979. *Trainable grammars for speech recognition.* In Speech Communications for the 97th Meeting of the Acoustical Society of America, 31-35.

[2] Benedí, José-Miguel and Joan-Andreu Sánchez. 2005. *Estimation of stochastic context-free grammars and their use as language models.* In Computer Speech and Language, 19(3):249-274, 2005.

[3] Benedí, José-Miguel, Joan-Andreu Sánchez and Alberto Sanchís. 2007. *Confidence measures for stochastic parsing.* In RANLP '07.

[4] Bisani, Maximilian and Hermann Ney. 2004. *Bootstrap estimates for confidence intervals in asr performance evaluation.* In ICASSP '04, I:409-412.

[5] Charniak, Eugene. 1996. *Tree-bank grammars.* In AAA I'96, 173-180.

[6] Charniak, Eugene. 2000. *A maximum-entropy-inspired parser.* In NAACL '00, 132-139.

[7] Charniak, Eugene, Kevin Knight, and Kenji Yamada. 2003. *Syntax-based language models for statistical machine translation.* In 9th Machine Translation Summit.

[8] Charniak, Eugene and Mark Johnson. 2005. *Coarse-to-fine n-best parsing and MaxEnt discriminative reranking.* In ACL '05, 173-180.

[9] Chelba, Ciprian and Frederick Jelinek. 2000. *Structured language modeling.* Computer Speech and Language, 14:283-332.

[10] Collins, Michael. 2003. *Head-driven statistical models for natural language parsing.* In Computational Linguistics, 29(4):589-637.

[11] Duda, Richard O., Peter E. Hart, and David G. Stork. 2001. *Pattern Classification and Scene Analysis.* John Wiley and Sons, New York.

[12] Earley, Jay. 1970. *An efficient context-free parsing algorithm.* In Communications of the Association for Computing Machinery, 8(6):451-455.

[13] Goodman, Joshua. 1996. *Prasing algorithms and metrics.* In ACL '96, 177-183.

[14] Hopcroft, John E and Jeffrey Ullman. 1979. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley.

[15] Huang, Liang. 2008. *Forest reranking: discriminative parsing with non-local features.* In ACL '08.

[16] Jelinek, Frederick. 1976. *Continuous speech recognition by statistical methods.* In IEEE '76, 4(64):532-556.

[17] Jelinek, Frederick. 1997. *Statistical methods for speech recognition.* The MIT Press, Cambridge, MA.

[18] Johnson, Mark. 1998. *PCFG models of linguistic tree representation.* In Computational Linguistics, 24:613-632.

[19] Klein, Dan and Chistopher D. Manning. 2001. *Parsing with treebank grammars: Empirical bounds, theoretical models, and the structure of the Penn treebank.* In ACL '01, 338-345.

[20] Klein, Dan and Chistopher D. Manning. 2003. *Accurate Unlexicalized Parsing.* In ACL '03, 423-430.

[21] Lease, Matthew, Eugene Charniak, Mark Johnson and David McClosky. 2006. *A look at parsing and its applications.* In National Conference on Artificial Intelligence, vol. 21-II, 1642-1645.

[22] Loper, Edward and Steven Bird. 2002. *NLTK: the Natural Language Toolkit.* In ACL '02 Workshop on effective tools and methodologies for teaching natural language processing and computational linguistics, volume 1, 63-70.

[23] Manning, Christopher and Hinrich Schütze. 1999. *Foundations of statistical language processing.* The MIT Press. England.

[24] Marcus, Mitchell, Beatrice Santorini and Mary Ann Marcinkiewicz. 1994. *Building a large annotated corpus of English: the Penn Treebank.* Computational Linguistics, volume 19, number 2.

[25] Matsuzaki, Takuya, Yasuke Miyao and Jun'ichi Tsujii. 2005. *Probabilistic CFG with latent annotations.* In ACL '05, 75-82.

[26] McClosky, David, Eugene Charniak and Mark Johnson 2006. *Effective self-training for parsing.* In HLT-NAACL '06

[27] Ney, Hermann, Ute Essen and Reinhard Kneser 1994. *On structuring probabilistic dependences in stochastic language modelling.* Computer, Speech and Language, (8):1-38.

[28] Ortmanns, Stefan, Hermann Ney and Xavier Aubert 1997. *A word graph algorithm for large vocabulary continous speech recognition.* Computer, Speech and Language, 11:43-72.

[29] Petrov, Slav, Leon Barrett, Romain Thibaux and Dan Klein. 2006. *Learning Accurate, Compact and Interpretable Tree Annotation.* In ACL '06, 433-440.

[30] Petrov, Slav and Dan Klein. 2007. *Improved inference for unlexicalized parsing.* In NAACL-HLT '07, 401-411.

[31] Rabiner, Lawrence and Biing-Hwang Juang. 1993. *Fundamentals of speech recognition.* Prentice-Hall, Englewood Cliffs.

[32] Roark, Brian. 2001. *Probabilistic top-down parsing and language modeling.* Computational Linguistics, 27(2):249-276.

[33] San-Segundo, Rubén, Bryan Pellom, Kadri Hacioglu, Wayne Ward and José M. Pardo. 2001. *Confidence measures for spoken dialogue systems.* In ICASSP '01, 393-396.

[34] Sanchís, Alberto, Alfons Juan and Enrique Vidal. 2003. *Improving utterance verification using a smoothed naive bayes model.* In ICASSP '03, 592-595.

[35] Sanchís, Alberto. *Estimacin y aplicacin de medidas de confianza en reconocimiento automático del habla.* PhD thesis, Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia, Valencia (Spain), March 2004. Advisor(s): Dr. E. Vidal and Dr. A. Juan (in Spanish).

[36] Salvador, Ismael and José-Miguel Benedí. *Rna modeling by combining stochastic context-free grammars and n-gram models.* In International Journal of Pattern Recognition and Artificial Intelligence, 16(3):309–315.

[37] Smith, David A. and Jason Eisner. 2007 *Bootstrapping Feature-Rich Dependency Parsers with Entropic Priors* In Proceedings of the Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pages 667-677, 2007.

[38] Ueffing, Nicola and Hermann Ney. 2007. *Word-level confidence estimation for machine translation.* In Computational Linguistics 33(1), 9-40.

[39] Wessel, Frank, Ralf Schlüter, Klaus Macherey and Hermann Ney. 2001. *Confidence measures for large vocabulary continuous speech recognition.* In IEEE Transactions on Speech and Audio Processing, 3(9):288-298.

[40] K. Yamada and K. Knight. 2002. *A Decoder for Syntax-based Statistical MT.* Meeting of the Association for Computational Linguistics.