Confidence Measures for Error Discrimination in an Interactive Predictive Parsing Framework[1]

Ricardo Sánchez-Sáez, Joan Andreu Sánchez and José Miguel Bened
Instituto Tecnológico de Informática
Universidad Politécnica de Valencia
{rsanchez,jandreu,jbenedi}@dsic.upv.es

Abstract

We study the use of Confidence Measures (CM) for erroneous constituent discrimination in an Interactive Predictive Parsing (IPP) framework. The IPP framework allows to build interactive tree annotation systems that can help human correctors in constructing error-free parse trees with little effort (compared to manually post-editing the trees obtained from an automatic parser). We show that CMs can help in detecting erroneous constituents more quickly through all the IPP process. We present two methods for precalculating the confidence threshold (globally and per-interaction), and observe that CMs remain highly discriminant as the IPP process advances.

1 Introduction

Within the Natural Language Processing (NLP) field, we can tell apart two different usage scenarios for automatic systems that output or work with natural language. On one hand, 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 amount of error (measured with metrics like Word Error Rate, BLEU, F-Measure, etc.) present within the results that are being produced.

The second usage scenario arises when there exists the need for perfect and completely error-free results, for example, flawlessly translated sentences or correctly annotated syntactic trees. In such cases, the intervention of a human validator/corrector is unavoidable. The corrector will review and validate the results, making the suitable modifications before the system output can be employed. In these kind of tasks, the most important factor to be minimized is the human effort that has to be applied to transform the system's potentially incorrect output into validated and error-free output. Measuring user effort has an intrinsic subjectivity that makes it hard to be quantitatized. Given that the user effort is usually inversely proportional to the quality of the system output, most research about problems associated to this scenario t to minimize just the system's error rate as well.

Interactive Predictive NLP Systems

Only recently, more comparable and reproducible evaluation methods for Interactive Natural Language Systems have started to be developed, within the context of Interactive Predictive Systems (IPS). These systems formally integrate the correcting user into the loop, making him part of the system right at its theoretical framework. IPSs allow for human correctors to spare effort because the system updates its output after each individual user correction, potentially fixing several errors at each step. Interactive Predictive methods have been studied and successfully used in fields

[1]Work partially supported by the Spanish MICINN under the MIPRCV "Consolider Ingenio 2010" (CSD2007-00018), MITTRAL (TIN2009-14633-C03-01), Prometeo (PROMETEO/2009/014) research projects, and the FPU fellowship AP2006-01363.

like Handwritten Text Recognition (HTR) (Toselli et al., 2008) and Statistical Machine Translation (SMT) (Vidal et al., 2006; Barrachina et al., 2009) to ease the work of transcriptors and translators.

In IPS related research the importance of the system base 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 together with new objective effort evaluation metrics such as the Word Stroke Ratio (WSR) (Toselli et al., 2008) or the Key-Stroke and Mouse-Ratio (KSMR) (Barrachina et al., 2009) started to be used as a benchmark. These ratios reflect the amount of user effort (whole-word corrections in the case of WSR; keystrokes plus mouse actions in the case of KSMR) given a certain output. To get the amount of user effort into context they should be measured against the corresponding error ratios of comparable non-interactive systems: Word Error Rate for WSR and Character Error Rate for KSMR.

This dichotomy in evaluating either system performance or user effort applies to Syntactic Parsing as well. The objective of parsing is to precisely determine the syntactic structure of sentences written in one of the several languages that humans use. Very bright research has been carried out in this field, resulting in several top performing completely automatic parsers (Collins, 2003; Klein and Manning, 2003; McClosky et al., 2006; Huang, 2008; Petrov, 2010). However, these produce results that are erroneous to some extent, and as such unsuitable for some applications without a previous manual correction. There are many problems where error-free results consisting in perfectly annotated trees are needed, such as handwritten mathematical expression recognition (Yamamoto et al., 2006) or construction of large new gold treebanks (de la Clergerie et al., 2008).

When using automatic parsers as a baseline for building perfect syntactic trees, the role of the human annotator is usually 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, a basic two-stage setup was employed in the creation of the Penn Treebank annotated corpus: a rudimentary parsing system provided a skeletal syntactic representation, which then was manually corrected by human annotators (Marcus et al., 1994). Additional works within this field have presented systems that act as a computerized aid to the user in obtaining the perfect annotation (Carter, 1997; Oepen et al., 2004; Hiroshi et al., 2005). Subjective measuring of the effort needed to obtain perfect annotations was reported in some of these works, 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, the authors of (Sánchez-Sáez et al., 2009a) devised an Interactive Predictive Parsing (IPP) framework. That work embeds the human corrector into the automatic parser, and allows him to interact in real time within the system. In this manner, the system can use the readily available user feedback to make predictions about the parts of the trees that have not been validated by the corrector. The authors simulated user interaction and calculated effort evaluation metrics, establishing that an IPP system results in amounts slightly above 40% of effort reduction for a manual annotator compared to a two-step system.

Confidence Measures in NLP

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 IPS 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 NLP system.

In fields such as HTR, SMT or Automatic Speech Recognition (ASR), the output sentences have a global probability (or score) that reflects the likeness of the output sentence being correct. CMs allow precision beyond the sentence level in predicting errors: they can be used to label the individual words as either correct or incorrect. Automatic systems can use CMs to help the user in identifying the erroneous parts of the output in a faster way or to aid with the amendments by suggesting replacement words that are likely to be correct.

Previous research shows that CMs have been successfully applied within the ASR (Wessel et al., 2001), HTR (Tarazón et al., 2009; Serrano et al., 2010) and SMT (Ueffing and Ney, 2007) fields. In these works, the ability of CMs in detecting erroneous constituents is assessed by the classical confidence metrics: the Confidence Error Rate (CER) and the Receiver Operating Characteristic (ROC) (Ueffing and Ney, 2007).

However, until recent advances, the use of CMs remained largely unexplored in Parsing. Assessing the correctness of the different parts of a parsing tree can be useful in improving the efficiency and usability of an IPP system, not only by *tagging* parts with low confidence for the user to review, but also by automating part of the correction process itself by presenting constituents that yield a higher confidence when an error is confirmed by the user.

CMs for parsing in the form of combinations of features calculated from n-best lists were proposed in (Benedí et al., 2007). Later on, the authors of (Sánchez-Sáez et al., 2009b) introduced a statistical method for calculating a CM for each of the constituents in a parse tree. In that work, CMs are calculated using the posterior probability of each tree constituent, approach which is similar to the word-graph based methods in the ASR and SMT fields.

In this paper, we apply Confidence Measures to the Interactive Predictive Parsing framework to asses how CMs are increasingly more accurate as the user validates subtrees within the interactive process. We prove that after each correction performed by the user, the CMs of the remaining unvalidated constituents are more helpful to detect errors.

2 Interactive Predictive Parsing

In this section we review the IPP framework (Sánchez-Sáez et al., 2009a) and its underlying operation protocol. In parsing, a syntactic tree t, attached to a string $\boldsymbol{x} = x_1 \dots x_{|x|}$ is composed by substructures called constituents. 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).

Here follows a general formulation for the non-interactive syntactic parsing scenario, which will allow us to better introduce the IPP formulation. Assume that using a given parsing model G, the parser analyzes the input sentence $\boldsymbol{x}$ and produces the most probable parse tree

$$\hat{t} = \arg\max_{t \in \mathcal{T}} p_G(t|\boldsymbol{x}), \qquad (1)$$

where $p_G(t|\boldsymbol{x})$ is the probability of the 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 the IPP framework, the manual corrector provides feedback to the system by correcting any of the constituents c_{ij}^{A} from $\hat{t}$. The system reacts to each of the corrections performed by the human annotator by proposing a new $\hat{t}'$ that takes into account the correction.

Within the IPP framework, the user reviews the constituents contained in the tree to assess their correctness. When the user finds an incorrect constituent he modifies it, setting the correct span and label. This action implicitly validates what it is called the *validated prefix tree* t_p.

We define the validated prefix tree to be composed by the partially corrected constituent, all of its ancestor constituents, and all constituents whose end span is lower than the start span of the corrected constituent. When the user replaces the constituent c_{ij}^{A} with the correct one $c_{ij}'^{A}$, the validated prefix tree is

$$\begin{aligned} t_p(c_{ij}'^{A}) = \{c_{mn}^{B} &: m \leq i,\ n \geq j\ , \\ & d(c_{mn}^{B}) \leq d(c_{ij}'^{A})\} \cup \\ \{c_{pq}^{D} &: q < i\ \} \end{aligned} \qquad (2)$$

with $d(c_{ab}^{Z})$ being the depth (distance from root) of constituent c_{ab}^{Z}.

The validated prefix tree is parallel to the validated sentence prefix commonly used in Interactive Machine Translation or Interactive Handwritten Recognition, and is established after each user action.

This particular definition of the prefix tree determines the fact that the user is expected to review the parse tree in a preorder fashion (left-to-right depth-first). Note that this specific exploration order allows us to simulate the user interaction for the experimentation, as we will explain below. Also note that other types of prefixes could be defined, allowing for different tree review orders.

Within the IPP formulation, when a constituent correction is performed, the prefix tree $t_p(c_{ij}^{\prime A})$ is validated and a new tree $\hat{t}'$ that takes into account the prefix is proposed. Incorporating this new evidence into expression (1) yields the following equation

$$\hat{t}' = \arg\max_{t \in \mathcal{T}} p_G(t|\boldsymbol{x}, t_p(c_{ij}^{\prime A})). \qquad (3)$$

Given the properties of Probabilistic Context-Free Grammars (PCFG) the only subtree that effectively needs to be recalculated is the one starting from the parent of the corrected constituent. This way, just the descendants of the newly introduced constituent, as well as its right hand siblings (along with their descendants) are calculated.

2.1 User Interaction Operation

The IPP formulation allows for a very straightforward operation protocol that is performed by the manual corrector, in which he validates or corrects the successive output parse trees:

1. The IPP system proposes a full parse tree t for the input sentence.
2. Then, the user finds the first incorrect constituent exploring the tree in a certain ordered manner (preorder in our case, given by the tree prefix definition) and amends it, by modifying its span and/or label (implicitly validating the prefix tree t_p).
3. The IPP system produces the most probable tree that is compatible with the validated prefix tree t_p as shown in expression (3).
4. These steps are iterated until a final, perfect parse tree is produced by the system and validated by the user.

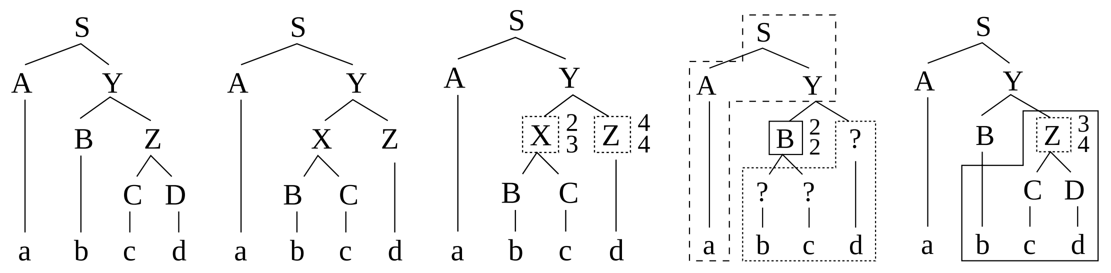
It is worth noting that within this protocol, constituents can be automatically deleted or inserted at the end of any subtree in the syntactic structure by adequately modifying the span of the left-neighbouring constituent.

The IPP interaction process is similar to the ones already established in HTR and SMT. In these fields, the user reads the output sentence from left to right. When the user finds and corrects an erroneous word, he is implicitly validating the prefix sentence up to that word. The remaining suffix sentence is recalculated by the system taking into account the validated prefix sentence.

Fig. 1 shows an example that intends to clarify the Interactive Predictive process. First, the system provides a tentative parse tree (Fig. 1.b). Then the user, which has the correct reference tree (Fig. 1.a) in mind, notices that it has two wrong constituents (c_{23}^{X} and c_{44}^{Z}) (Fig. 1.c), and chooses to replace c_{23}^{X} by c_{22}^{B} (Fig. 1.d). Here, c_{22}^{B} corresponds to $c_{ij}^{\prime A}$ of expression (3). As the user does this correction, the system automatically validates the prefix (dashed line in Fig. 1.d, $t_p(c_{ij}^{\prime A})$ of expression (2)). The system also invalidates the subtrees outside the prefix (dotted line line in Fig. 1.d). Finally, the system automatically predicts a new subtree (Fig. 1.e). Notice how c_{34}^{Z} changes its span and c_{44}^{D} is introduced which provides the correct reference parse.

For further exemplification, Sánchez-Sáez et al. (2010) demonstrate an IPP based annotation tool that can be accessed at `http://cat.iti.upv.es/ipp/`.

Within the IPP scenario, the user has to manually review all the system output and correct or validate it, which is still a considerable amount of effort. CMs can ease this work by helping to spot the erroneous constituents.



(a) Reference tree (b) Iteration 0: *Proposed output tree 1* (c) Iteration 0: *Erroneous constituents* (d) Iteration 1: *User corrected constituent* (e) Iteration 1: *Proposed output tree 2*

Figure 1: Synthetic example of user interaction with the IPP system.

3 Confidence Measures

Probabilistic calculation of Confidence Measures (Sánchez-Sáez et al., 2009b) for all tree constituents can be introduced within the IPP process.

The CM of each constituent is its posterior probability, which can be considered as a measure of the degree to which the constituent is believed to be correct for a given input sentence $\boldsymbol{x}$. This is formulated as follows

$$p_G(c_{ij}^A|\boldsymbol{x}) = \frac{p_G(c_{ij}^A,\boldsymbol{x})}{p_G(\boldsymbol{x})} = \frac{\sum_{t'\in\mathcal{T};\, c_{ij}'^A\in t'} \delta(c_{ij}^A, c_{ij}'^A)\ p_G(t'|\boldsymbol{x})}{p_G(\boldsymbol{x})} \tag{4}$$

with $\delta()$ being the Kronecker delta function. Numerator in expression (4) stands for the probability of all parse trees for $\boldsymbol{x}$ that contain the constituent c_{ij}^A (see Fig. 2).

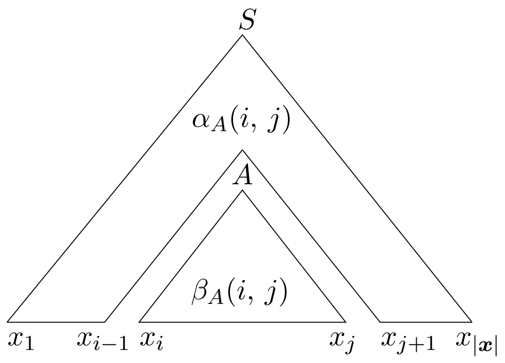


Figure 2: The product of the inside and outside probabilities for each constituent comprises the upper part of expression (5)

The posterior probability is computed with the inside β and outside α probabilities (Baker, 1979)

$$\begin{aligned}\mathcal{C}(t_{ij}^A) = p_G(c_{ij}^A|\boldsymbol{x}) &= \frac{p_G(c_{ij}^A,\boldsymbol{x})}{p_G(\boldsymbol{x})} \\ &= \frac{\beta_A(i,\, j)\ \alpha_A(i,\, j)}{\beta_S(1,\, |\boldsymbol{x}|)}\ .\end{aligned} \tag{5}$$

It should be clear that the calculation of confidence measures reviewed here is generalizable for any problem that employs PCFGs, and not just NLP tasks. In the experiments presented in the following section we show that CMs are increasingly discriminant when used within the IPP framework to detect erroneous constituents.

4 Experiments

Evaluation of the quality of CMs within the IPP framework is done in a completely automatic fashion by simulating user interaction. Section 4.1 introduces the evaluation protocol and metrics measuring CM quality (i.e., their ability to detect incorrect constituents). The experimentation framework and the results are discussed in section 4.2.

4.1 Evaluation Methods

4.1.1 IPP Evaluation

A good measure of the performance of an Interactive Predictive System is the amount of effort saved by the users of such a system. It is subjective and expensive to test an IPS with real users, so these systems are usually evaluated using automatically calculated metrics that assess the amount of effort saved by the user.

As already mentioned, the objective of an IPP based system is to be employed by annotators to construct correct syntactic trees with less effort. Evaluation of an IPP system was previously done by comparing the IPP usage effort (the number of corrections using the IPP system) against the estimated effort required to manually post-edit the trees after obtaining them with a traditional automatic parsing system (the amount of incorrect constituents) (Sánchez-Sáez et al., 2009a).

In the case of IPP, the gold reference trees are used to simulate system interaction by a human corrector and provide a comparable benchmark. This automatic evaluation protocol is similar to the one presented in section 2.1:

1. The IPP system proposes a full parse tree 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 (preorder) and comparing it with the *reference*. When the first erroneous constituent is found, it is amended by being replaced in the output tree by the correct one, operation which implicitly validates the prefix tree t_p.

3. The IPP system produces the most probable tree that is compatible with the validated prefix tree t_p.

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

In this work, metrics assessing the quality of CM are introduced within this automatic protocol. We calculate and report them after each of the iterations in the IPP process.

4.1.2 Confidence Measure Evaluation Metrics

The CM of each tree constituent, computed as shown in expression (4) can be seen as its probability of being correct. Once all CM are calculated, a confidence threshold $\tau \in [0, 1]$ can be chosen. Constituents are then marked using τ: the ones with a confidence above this threshold are marked as correct, and the rest as incorrect. Comparing the confidence marks in the output tree with the reference, we obtain the *false rejection* $N_f(\tau) \in [0, N_c]$ (number of correct constituents in the output tree wrongly marked as incorrect by their CM) and the *true rejection* $N_t(\tau) \in [0, N_i]$ (number of incorrect constituents in the output tree that are indeed detected as incorrect by their confidence).

The amount of correct and incorrect constituents in each tree is N_c and N_i respectively. In the ideal case of perfectly error discriminant CM, using the best threshold would yield $N_f(\tau) = 0$ and $N_t(\tau) = N_i$.

A evaluation metric that assess the ability of CMs in telling apart correct constituents from incorrect ones is the *Confidence Error Rate (CER)*:

$$CER(\tau) = \frac{N_f(\tau) + (N_i - N_t(\tau))}{N_c + N_i}. \quad (6)$$

The CER is the number of errors incurred by the CMs divided by the total number of constituents.

The CER can be compared with the Absolute Constituent Error Rate (ACER), which is the CER obtained assuming that all constituents are marked as correct (the only possible assumption when CM are not available):

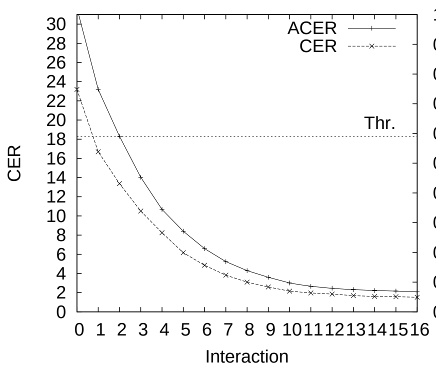
$$ACER = CER(0) = \frac{N_i}{N_c + N_i}. \quad (7)$$

4.2 Experimental Framework

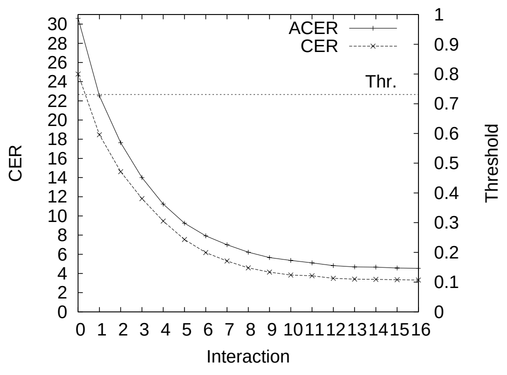
Our experiments were carried out over the Wall Street Journal Penn Treebank (PTB) manually annotated corpus. Three sets were defined over the PTB: train (sections 2 to 21), test (section 23), and development (the first 346 sentences of section 24). Before carrying out experimentation, the *NoEmpties* transformation was applied to all sets (Klein and Manning, 2001).

We implemented the CYK-Viterbi parsing algorithm as the parse engine within the IPP framework. This algorithm uses grammars in the Chomsky Normal Form (CNF) so we employed the open source Natural Language Toolkit[2] (NLTK) to obtain several right-factored binary

[2]`http://www.nltk.org/`

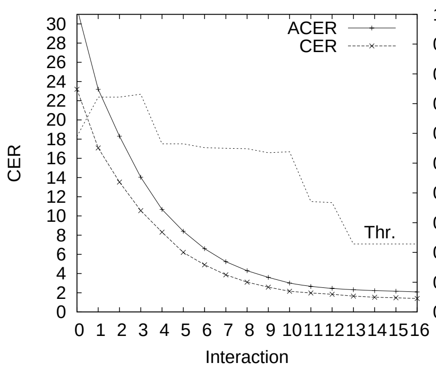


(a) PCFG: h=0,v=1

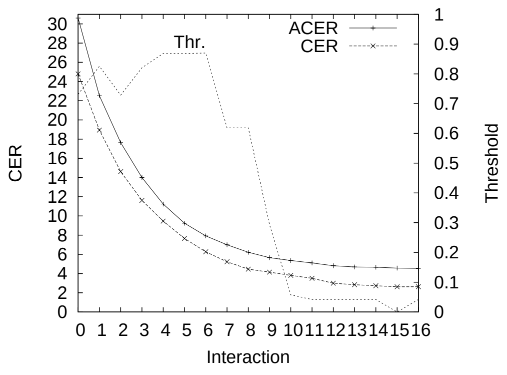


(b) PCFG: h=0,v=2

Figure 3: CER results over IPP system interaction. Threshold fixed at before the interactive process.



(a) PCFG: h=0,v=1



(b) PCFG: h=0,v=2

Figure 4: CER results over IPP system interaction. Threshold optimized for each step of the interactive process.

grammars with different markovization parameters from the training set (Klein and Manning, 2003).

The purpose of our experimentation is to determine if CMs can successfully discriminate erroneous constituents from correct ones within an IPP process, that is, if they help the user to find errors in a hastier manner. For this we need to assess if there exists discriminant information in the CMs corresponding to the constituents of the unvalidated part of the successive IPP-proposed trees.

With this objective in mind, we introduced a CM calculation step after each user interaction within the IPP process. CMs for all constituents in each tree were obtained as described in section 3. After each simulated interaction, we also calculated the ACER and CER over all the syntactic constituents of the whole test set.

Each IPP user interaction yields a parse tree which can be seen as the concatenation of two parts: the validated prefix tree (which is known to be correct because the user, or the user simulation subsystem in this case, has already reviewed it) and a new suffix tree which is calculated by the IPP system based on the validated prefix, as shown in section 2.

The fact that the validated prefix is already known to be correct is taken into account by the CM calculation process, and the confidence of the constituents in the prefix tree is automatically set to their maximum score, equal to 1. This fact causes that the CMs become more discriminant after each interaction, because a larger part of the tree (the prefix) has a completely correct confidence. The key point here is to measure if this increasingly reduced CER (CM error rate) maintains its advantage over the also increasingly reduced ACER (absolute constituent error rate without taking CMs into account) which would mean that the CMs retain their discriminant power and can be useful as an aid for a human annotator using an IPP system.

Two batches of experiments were performed and, in each of them, two different markovizations of the vanilla PCFG were tested as the parsing model.

In the first battery of experiments, the confidence threshold τ was optimized over the development set before starting the IPP process, remaining the same during the user interaction. The results can be seen in Fig. 3, which shows the obtained baseline ACER and the CER (the confidence assessing metric) for the test set after each user interaction. We see how CMs retain all of their error detection capabilities during the IPP process: in the h0v1 PCFG they are able to discern about 25% of incorrect constituents at most stages of the IPP process, with a slight bump up to 27% after about 7 user interactions; for the h0v2 PCFG they are able to detect about 18% of incorrect constituents at the first interactions, but go up to detect 27% of errors after about 7 or more interactions.

In the second experimental setup, a different threshold for each interaction step was calculated by performing the IPP user simulation process over the development set and optimizing the threshold value. The results can be seen in Fig. 4. We observe improvements in the discriminant ability of confidence values after 8 user interactions, with them being capable to detect more errors towards the end of each IPP session: about 34% of errors for h0v1, and 49% of them for h0v2.

The calculated thresholds have also been plotted in the aforementioned figures. For the per-interaction threshold experimentation, we can see how the threshold gets fine-tuned as the IPP process advances. The lower threshold values for the last interactions were expected due to the fact that more constituents have been validated and have the maximum confidence. This method for precalculating one specific threshold for each of the iterations could be useful when incorporating CM to a real IPP based annotator.

5 Conclusions and Future Work

We have proved that using Confidence Measures can be used to discriminate incorrect constituents from correct ones over an Interactive Predictive Parsing process. We have show two methods for calculating the threshold used to mark constituents as correct/incorrect, showing the advantage of precalculating a specific threshold for each of the interaction steps.

Immediate future work involves implementing CMs as a visual aid in a real IPP system like the one presented in (Sánchez-Sáez et al., 2010). Through he use of CMs, all constituents in the successive trees could be color-coded according to their correctness confidence, so the user could focus and make corrections faster.

Future research paths can deal with applying CMs to improve the output of completely automatic parsers, for example, using them as a component of an n-best re-ranking system.

Additionally, the IPP framework is also suitable for studying and applying training algorithms within the Active Learning and Adaptative/Online Parsing paradigms. This kind of systems could improve their models at operating time, by incorporating new ground truth data as it is provided by the user.

References

Baker, JK. 1979. Trainable grammars for speech recognition. *Journal of the Acoustical Society of America*, 65:132.

Barrachina, S., O. Bender, F. Casacuberta, J. Civera, E. Cubel, S. Khadivi, A. Lagarda, H. Ney, J. Tomás, E. Vidal, and J.M. Vilar. 2009. Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1):3–28.

Benedí, J.M., J.A. Sánchez, and A. Sanchís. 2007. Confidence measures for stochastic parsing. In *Proc. of RANLP*, pages 58–63, Borovets, Bulgaria, 27-29 September.

Carter, D. 1997. The TreeBanker. A tool for supervised training of parsed corpora. In *Proc. of ENVGRAM Workshop*, pages 9–15.

Collins, M. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

de la Clergerie, E.V., O. Hamon, D. Mostefa, C. Ayache, P. Paroubek, and A. Vilnat. 2008. Passage: from French parser evaluation to large sized treebank. *Proc. of LREC*, 100:2.

Hiroshi, I., N. Masaki, H. Taiichi, T. Takenobu, and T. Hozumi. 2005. eBonsai: An integrated environment for annotating treebanks. In *Proc. of IJCNLP*, pages 108–113.

Huang, L. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proc. of ACL*.

Klein, D. and C.D. Manning. 2001. Parsing with treebank grammars: Empirical bounds, theoretical models, and the structure of the Penn treebank. In *Proc. of ACL*, pages 338–345, Morristown, USA. ACL.

Klein, D. and C.D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. of ACL*, volume 1, pages 423–430, Morristown, USA. ACL.

Marcus, M.P., B. Santorini, and M.A. Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

McClosky, D., E. Charniak, and M. Johnson. 2006. Effective self-training for parsing. In *Proc. of NAACL-HLT*, pages 152–159.

Oepen, S., D. Flickinger, K. Toutanova, and C.D. Manning. 2004. LinGO Redwoods. *Research on Language & Computation*, 2(4):575–596.

Petrov, S. 2010. Products of Random Latent Variable Grammars. *Proc. of NAACL-HLT*.

Sánchez-Sáez, R., J.A. Sánchez, and J.M. Benedí. 2009a. Interactive predictive parsing. In *Proc. of IWPT'09*, pages 222–225, Paris, France, October. ACL.

Sánchez-Sáez, R., J.A. Sánchez, and J.M. Benedí. 2009b. Statistical confidence measures for probabilistic parsing. In *Proc. of RANLP*, pages 388–392, Borovets, Bulgaria, September.

Sánchez-Sáez, R., L.A. Leiva, J.A. Sánchez, and J.M. Benedí. 2010. Interactive predictive parsing using a web-based architecture. In *Proc. of NAACL-HLT*, Los Angeles, United States of America, June.

Serrano, N., A. Sanchis, and A. Juan. 2010. Balancing error and supervision effort in interactive-predictive handwriting recognition. In *Proc. of IUI*, pages 373–376. ACM.

Tarazón, L., D. Pérez, N. Serrano, V. Alabau, O. Ramos Terrades, A. Sanchis, and A. Juan. 2009. Confidence Measures for Error Correction in Interactive Transcription of Handwritten Text. In *Proc. of ICIAP*, pages 567–574, Vietri sul Mare, Italy, September. LNCS.

Toselli, A.H., V. Romero, and E. Vidal. 2008. Computer assisted transcription of text images and multimodal interaction. In *Proc. MLMI*, volume 5237, pages 296–308. Springer.

Ueffing, N. and H. Ney. 2007. Word-level confidence estimation for machine translation. *Computational Linguistics*, 33(1):9–40.

Vidal, E., F. Casacuberta, L. Rodríguez, J. Civera, and C. Martínez. 2006. Computer-assisted translation using speech recognition. *IEEE TASLP*, 14(3):941–951.

Wessel, F., R. Schluter, K. Macherey, and H. Ney. 2001. Confidence measures for large vocabulary continuous speech recognition. *IEEE TSAP*, 9(3):288–298.

Yamamoto, R., S. Sako, T. Nishimoto, and S. Sagayama. 2006. On-line recognition of handwritten mathematical expressions based on stroke-based stochastic context-free grammar. In *Proc of ICFHR*, pages 249–254.