Interactive Predictive Parsing using a Web-based Architecture*

Ricardo Sánchez-Sáez[†] **Luis A. Leiva**[‡] **Joan-Andreu Sánchez**[†] **José-Miguel Benedí**[†]

Instituto Tecnológico de Informática
Universidad Politécnica de Valencia

{rsanchez,luileito,jandreu,jbenedi}@{†dsic,‡iti}.upv.es

Abstract

This paper introduces a Web-based demonstration of an interactive-predictive framework for syntactic tree annotation, where the user is tightly integrated into the interactive parsing system. In contrast with the traditional post-editing approach, both the user and the system cooperate to generate error-free annotated trees. User feedback is provided by means of natural mouse gestures and keyboard strokes.

1 Introduction

There is a whole family of problems within the parsing world where error-free results, in the form of perfectly annotated trees, are needed. Constructing error-free trees is a necessity in many tasks, such as handwritten mathematical expression recognition (Yamamoto et al., 2006), or new gold standard treebank creation (de la Clergerie et al., 2008). It is a fact that current state-of-the-art syntactic parsers provide trees that, although of excellent quality, still contain errors. Because of this, the figure of a human corrector who supervises the annotation process is unavoidable in this kind of problems.

When using automatic parsers as a baseline for building perfect syntactic trees, the role of the human annotator is to post-edit the trees and correct the errors. This manner of operating results in the typical two-step process for error correcting, in which the system first generates the whole output and then the user verifies or amends it. This paradigm is rather inefficient and uncomfortable for the human annotator. For example, in the creation of the Penn Treebank annotated corpus, a basic two-stage setup was employed: a rudimentary parsing system provided a skeletal syntactic representation, which then was manually corrected by human annotators (Marcus et al., 1994). Other tree annotating tools within the two-step paradigm exist, such as the TreeBanker (Carter, 1997) or the Tree Editor TrEd[1].

With the objective of reducing the user effort and making this laborious task easier, we devised an Interactive Predictive framework. Our aim is to put the user into the loop, embedding him as a part of the automatic parser, and allowing 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 that have not been validated by the corrector.

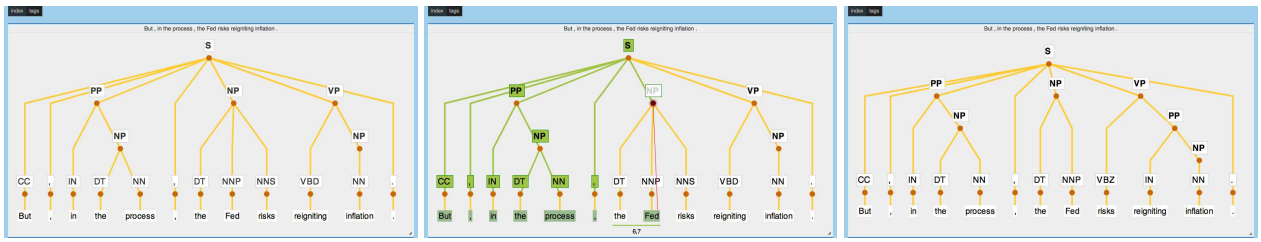
In this paper, we present a Web-based demo, which implements the Interactive Predictive Parsing (IPP) framework presented in (Sánchez-Sáez et al., 2009). User feedback (provided by means of keyboard and mouse operations) allows the system to predict new subtrees for unvalidated parts of the annotated sentence, which in turn reduces the human effort and improves annotation efficiency.

As a back-end for our demo, we use a more polished version of the CAT-API library, the Web-based Computer Assisted Tool introduced in (Alabau et al., 2009). This library allows for a clean application design, in which both the server side (the parsing engine) and the client side (which draws the trees, captures and interprets the user feedback, and requests

*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. The authors wish to thank Vicent Alabau for his invaluable help with the CAT-API library.

[1] *http://ufal.mff.cuni.cz/∼pajas/tred/*

Proceedings of the NAACL HLT 2010: Demonstration Session, pages 37–40,
Los Angeles, California, June 2010.



(a) System: output tree 1 (b) User: *span modification* (c) System: output tree 2

Figure 1: An interaction example on the IPP system.

parsed subtrees to the server) are independent. One of the features that steam from the CAT-API library is the ability for several annotators to work concurrently on the same problem-set, each in a different client computer sharing the same parsing server.

Interactive predictive methods have been successfully demonstrated to ease the work of transcriptors and translators in fields like Handwriting Text Recognition (Romero et al., 2009; Toselli et al., 2008) and Statistical Machine Translation (Ortiz et al., 2010; Vidal et al., 2006). This new paradigm enables the collaboration between annotators across the globe, granting them a physical and geographical freedom that was inconceivable in the past.

2 Interactive Predictive Parsing

A tree t, associated to a string $x_{1|x|}$, is composed by substructures that are usually referred as constituents. A constituent c_{ij}^{A} is defined by the non-terminal symbol A (either a *syntactic label* or a *POS tag*) 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 parsing scenario. Using a grammatical model G, the parser analyzes the input sentence $\boldsymbol{x} = \{x_1, \ldots, x_{|x|}\}$ and produces the parse tree $\hat{t}$

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

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 the interactive predictive scenario, after obtaining the (probably incorrect) best tree $\hat{t}$, the user is able to individually correct any of its constituents c_{ij}^{A}. The system reacts to each of the corrections introduced by the human, proposing a new $\hat{t}'$ that takes into account the afore-mentioned corrections.

The action of modifying an incorrect constituent (either setting the correct span or the correct label) implicitly validates a subtree that is 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. We will name this subtree the validated prefix tree t_p. When the user replaces the constituent c_{ij}^{A} with the correct one $c_{ij}^{\prime A}$, the validated prefix tree is:

$$\begin{aligned} t_p(c_{ij}^{\prime A}) = \{c_{mn}^{B} \ : \ & m \leq i, \ n \geq j, \\ & d(c_{mn}^{B}) \leq d(c_{ij}^{\prime A})\} \cup \\ \{c_{pq}^{D} \ : \ & p >= 1 \ , \ q < i\} \end{aligned} \tag{2}$$

with $d(c_{mn}^{B})$ being the depth of constituent c_{mn}^{B}.

When a constituent correction is performed, the prefix tree $t_p(c_{ij}^{\prime A})$ is fixed and a new tree $\hat{t}'$ that takes into account the prefix is proposed

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

Given that we are working with context-free grammars, the only subtree that effectively needs to be recalculated is the one starting from the parent of the corrected constituent.

3 Demo outline

A preview version of the demonstration can be accessed at *http://cat.iti.upv.es/ipp/*.

The user is presented with the sentences in the selected corpus, and starts parsing them one by one. They make corrections in the trees both with the keyboard and the computer mouse. The user feedback

is decoded on the client side which in turn requests subtrees to the parse engine.

Two kind of operations can be performed over constituents: span modification (performed either by dragging a line from the constituent to the word that corresponds to the span's upper index, or deleting a tree branch by clicking on it), and label substitution (done by typing the correct one on its text field). Modifying the span of a constituent invalidates its label, so the server recalculates it as part of the suffix. Modifying the label of a constituent validates its span.

When the user is about to perform an operation, the affected constituent and the prefix that will be validated are highlighted. The target span of the modified constituent is visually shown as well. When the user obtains the correctly annotated tree, they can accept it by by clicking on a new sentence.

As already mentioned, the user is tightly integrated into the interactive parsing process. They follow a predetermined protocol in which they correct and/or validate the annotated parse trees:

1. The parsing server proposes a full parse tree t for the input sentence. The tree t is shown to the user by the client (Fig. 1a).

2. The user finds the first[2] incorrect constituent c and starts amending it, either by changing its label or changing its span (Fig. 1b, note how the label is greyed out as it is discarded with the span modification). This operation implicitly validates the prefix tree t_p (highlighted in green).

3. The system decodes the user feedback (i.e., mouse gestures or keyboard strokes) which can either affect the label or the span of the incorrect constituent c:

   (a) If the span of c is modified, the label is not assumed to be correct. A partial constituent c^*, which includes *span* but no *label*, is decoded from the user feedback.

   (b) If the label of c is modified, the span is assumed to be correct. The corrected constituent c' is decoded from the user feedback.

   This step only deals with analysing the user feedback, the parsing server will not be contacted until the next step.

4. Either the partially corrected constituent c^* or the corrected constituent c' is then used by the client to create a new *extended consolidated prefix* that combines the validated prefix and the user feedback: either $t_p c^*$ or $t_p c'$. The client sends the extended prefix tree to the parsing server and requests a suitable continuation for the parse tree, or tree suffix t_s:

   (a) If the extended prefix is partial ($t_p c^*$), the first element of t_s is the label completing c^*, followed by the remaining calculated whole constituents.

   (b) If the extended prefix is complete ($t_p c'$), the parsing server produces a suitable tree suffix t_s which contains the remaining calculated whole constituents.

5. The client concatenates the suffix returned by the server to the validated extended prefix, and shows the whole tree to the client (Fig. 1c).

6. These previous steps are iterated until a final, perfect parse tree is produced by the server and validated by the user.

Note that within this protocol, constituents can be deleted or inserted by adequately modifying the span of the left-neighbouring constituent.

4 Demo architecture

The proposed system coordinates client-side scripting with server-side technologies, by using the CAT-API library (Alabau et al., 2009).

4.1 Server side

The server side of our system is a parsing engine based on a customized CYK-Viterbi parser, which uses a Probabilistic Context-Free Grammar in Chomsky Normal Form obtained from sections 2 to 21 of the UPenn Treebank as a model (see (Sánchez-Sáez et al., 2009) for details).

The client can request to the parsing server the best subtree for any given span of the input string. For each requested subtree, the client can either provide the starting label or not. If the starting subtree

[2]The tree visiting order is left-to-right depth-first.

label is not provided, the server calculates the most probable label. The server also performs transparent tree debinarization/binarization when communicating with the client.

4.2 Client side

The client side has been designed taking into account ergonomic issues in order to facilitate the interaction.

The prototype is accessed through a Web browser, and the only requirement is the Flash plugin (98% of market penetration) installed in the client machine. The hardware requirements in the client are very low on the client side, as the parsing is process performed remotely on the server side: any computer (including netbooks) capable of running a modern Web browser is enough.

Each validated user interaction is saved as a log file on the server side, so a tree's annotation session can be later resumed.

4.2.1 Communication protocol

This demo exploits the WWW to enable the connection of simultaneous accesses across the globe. This architecture also provides cross-platform compatibility and requires neither computational power nor disk space on the client's machine.

Client and server communicate via asynchronous HTTP connections, providing thus a richer interactive experience – no page refreshes is required when parsing a new sentence. Moreover, the Web client communicates with the IPP engine through binary TCP sockets. Thus, response times are quite slow – a desired requirement for the user's solace. Additionally, cross-domain requests are possible, so the user could switch between different IPP engines within the same UI.

5 Evaluation results

We have carried out experiments that simulate user interaction using section 23 of the Penn Treebank. The results suggest figures ranging from 42% to 46% of effort saving compared to manually post-editing the trees without an interactive system. In other words, for every 100 erroneous constituents produced by a parsing system, an IPP user would correct only 58 (the other 42 constituents being automatically recalculated by the IPP system). Again, see (Sánchez-Sáez et al., 2009) for the details on experimentation.

5.1 Conclusions and future work

We have introduced a Web-based interactive-predictive system that, by using a parse engine in an integrated manner, aids the user in creating correctly annotated syntactic trees. Our system greatly reduces the human effort required for this task compared to using a non-interactive automatic system.

Future work includes improvements to the client side (e.g., confidence measures as a visual aid, multimodality), as well as exploring other kinds of parsing algorithms for the server side (e.g., adaptative parsing).

References

V. Alabau, D. Ortiz, V. Romero, and J. Ocampo. 2009. A multimodal predictive-interactive application for computer assisted transcription and translation. In *ICMI-MLMI '09*, 227–228.

D. Carter. 1997. The TreeBanker. A tool for supervised training of parsed corpora. In *ENVGRAM'97*, 9–15.

E.V. de la Clergerie, O. Hamon, D. Mostefa, C. Ayache, P. Paroubek, and A. Vilnat. 2008. Passage: from French parser evaluation to large sized treebank. In *LREC'08*, 100:P2.

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

D. Ortiz, L. A. Leiva, V. Alabau, and F. Casacuberta. 2010. Interactive machine translation using a web-based architecture. In *IUI'10*, 423–425.

V. Romero, L. A. Leiva, A. H. Toselli, and E. Vidal. 2009. Interactive multimodal transcription of text imagse using a web-based demo system. In *IUI'09*, 477–478.

R. Sánchez-Sáez, J.A. Sánchez, and J.M. Benedí. 2009. Interactive predictive parsing. In *IWPT'09*, 222–225.

A.H. Toselli, V. Romero, and E. Vidal. 2008. Computer assisted transcription of text images and multimodal interaction. In *MLMI'08*, 5237: 296–308.

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

R. Yamamoto, S. Sako, T. Nishimoto, and S. Sagayama. 2006. On-line recognition of handwritten mathematical expressions based on stroke-based stochastic context-free grammar. In *10th Frontiers in Handwriting Recognition*, 249–254.