GLOSS: A Visual Interactive Tool for Discourse Annotation

Dan Cristea, Ovidiu Crăciun, Cristian Ursu

University “A.I.Cuza” Iasi
Faculty of Computer Science
16, Bethelot St.
6600 – Iasi, Romania
{dcristea, noel, cursu}@thor.infoiasi.ro

Themes: current work on annotation tools, interaction of human and automatic annotation tools.

Abstract

We present an annotation tool, called GLOSS, that manifests the following features: accepts as inputs SGML source documents and/or their database images and produces as output source SGML documents as well as the associated database images; allows for simultaneous opening of more documents; can collapse independent annotation views of the same original document, which also allows for a layer-by-layer annotation process in different annotation sessions and by different annotators, including automatic; offers an attractive interface to the user; permits discourse structure annotation by offering a pair of building operations (adjoining and substitution) and remaking operations (undo, delete parent-child link and tree dismember). Finally we display an example that shows how GLOSS is employed to validate, using a corpora, a theory of global discourse.

A demo can be offered on a PC platform running Windows’95 or NT.

Submitted to Recent Advances in Corpus Annotation, workshop to be held as part of the 10th European Summer School in Logic, Language and Information (ESSLLI’98) 17-28 August Sàarbruecken, Germany
GLOSS: A Visual Interactive Tool for Discourse Annotation

1. Introduction

In (Cristea, Ide, Romary, 1998a) a marking procedure was proposed in order to deal with the goal of annotating corpora from two different perspectives: discourse structure and reference. This scheme yielded by the necessity to validate a theory on discourse structure in correlation with anaphora (Cristea, Ide, Romary, 1998b).

This paper presents an annotation tool, called GLOSS, that implements the ideas presented in (Cristea, Ide, Romary, 1998a) regarding multiple annotation views on the same original document, while also providing a powerful visual interactive interface for annotation of discourse structure. The result of the annotation process is a pair of files: an ASCII SGML document and its database mirror, which can be SQL-accessed.

2. The annotator's features

SGML compatibility

The annotator accepts in input original un-annotated texts as well as SGML documents that must be paired with their DTD files¹. At any moment of the annotation process the document under annotation can be saved in the SGML format.

Database image copy

During the annotation process, an internal representation of the markups is kept in an associated database. When an annotation session is finished, the associated database can be saved for interrogation purposes. Queries addressed to the database can be expressed in the SQL language.⁵

Once a database image of a document exists, it can act as input for a subsequent annotation session with GLOSS. This yields the interesting feature of permitting the enriching of certain types of tags by an automatic procedure, as sketched in figure 1.

Figure 1: Mixed manual-automatic annotation with GLOSS

¹ The current implementation allows for a simplified DTD syntax.
² The current implementation does not allow that a database interrogation to be conducted from inside the annotator itself.
Multiple documents /multiple views feature

The annotator allows for simultaneous opening of more than one document, the same way an editor behaves. The user can commute from one document to another, each document having an associated database image.

The multi-document feature is justified by the multiple-views behaviour (Cristea, Ide, Romary, 1998a). According to this philosophy a document can inherit from multiple documents, considered as its parents, in a hierarchical structure. The overall architecture is that of a DAG (directed acyclic graph) with a base document as root. All documents in the hierarchy represent annotations made from different perspectives of the same original hub document. If a document is defined to inherit from one or more parents, then all the contributing markups are displayed simultaneously on the screen.

The multiple views behaviour is obtained by a process of intelligent unifying of the parent document's database representations. So, when a document is declared to inherit from two or more parent documents, another database is generated that copies once from the parents the common parts and than adds the markups that are specific to each of them. Ulterior to the moment the declaration of parentage is made, which must occur when a new view is created, the current document looses any connection with the parent documents, such that any modification can be made to this version without affecting in any way the original markups.

The main interactive workspace

Each opened document under annotation has a main window where the original text is displayed. The SGML tags do not appear on this screen. Instead, the tagged segments of text are highlighted in colours. The user can assign a different colour to each type of tag. Highlights are sensible to the mouse click and can open for visualisation and/or modification the corresponding attribute-value structures.

Text-empty tags, as for instance co-reference links (Brunesseaux and Romary, 1997) or (in a certain representation) discourse relations, are also displayed. Since there is no text that could anchor, by highlighting, these tags, only their IDs are displayed. This happens in a child window where the text-empty tags are grouped by their types. As in the case of tags that surround text, the text-empty tags can be opened for inspection/modification with the mouse click.

The discourse structure annotation workspace

Annotating the discourse structure in GLOSS is an interactive visual process that aims at creating a binary tree (Marcu, 1996, Cristea and Webber, 1997), where intermediate nodes are relations and terminal nodes are units. The discourse structure window appears near the main document window, as shown in figure 2.
Figure 2: The main and the discourse structure workspaces

An incremental, unit by unit evolution, that would precisely mimic an automatic expectation-based parsing, as described in (Cristea and Webber, 1997), is not compulsory during a manual process. A manual annotation process resembles more a trial and error, islands driven process. To facilitate the tree structure building, GLOSS allows development of partial trees that could subsequently be integrated into existing structures. Each unit or partial structure is incorporated within one of the already existing trees either by an adjoining (adding to a developing structure an auxiliary tree - one that has an foot node, marked by a *), on the left extreme of the terminal frontier) or a substitution operation (replacing of a substitution node, marked by a ‡, of a developing structure with a whole tree) (Cristea and Webber, 1997). Adjoining of a partial tree (minimum a unit) into an already existing, developing tree is allowed only on the nodes of the outer or inner right frontier\(^3\), as defined in (Cristea and Webber, 1997). The approach in (Cristea and Webber, 1997) that restricts substitution only into the most inner substitution node is also preserved.

Both build operations (adjoining and substitution) require two arguments: the inserted structure and the node of the final structure where the insertion takes place. The build operations are drag-and-drop from the root of the inserted partial structure to the node of an existing structure where the insertion must take place. During dragging, when the destination node is reached, a float appears near the point of insertion displaying all structures (auxiliary/elementary trees) allowed to be adjoined/substituted at that point. Moving the mouse cursor to one of these structures and releasing the button, the structure

\(^3\) An outer right frontier is the right frontier (the nodes on the path from the root to the most right terminal node) - in a tree without substitution sides. An inner right frontier is the right frontier of the tree rooted at the left sister of the most inner substitution site – in a tree with substitution sites.
for being inserted in that place is selected. Two possible floating windows are shown in figure 3, for adjoining and substitution. The material node in all cases must be understood as being replaced by the source tree.

Figure 3: Floating windows with adjoining and substitution structures

The only trees proposed in the floating are such that after completion of the intended operation on the active node the obtained tree to obey the Principle of Sequentiality\(^4\) (Cristea and Webber, 1997).

**Remaking the discourse tree**

As discourse annotation is an ever-refining process, we have implemented a feature that permits an unrestricted number of undoes of operations on the tree structures as well as cuts of father-child links and complete dismembering of parts of the developing tree.

Besides undo, the **remaking operations** are:
- **deleteUpper**: deletes the upper link of a node. As a result the tree under the node becomes unbound and the parent's corresponding link remains pending and is marked by a \(-\) because its further behaviour will be that of a substitution node;
- **dismemberTree**: forgets all the relations and the parent-child links of the structure under the selected node.

**3. Applying GLOSS for validating the Veins Theory**

In (Cristea, Ide, Romary, 1998b) a theory of global discourse called Veins Theory (VT) was proposed. It claims that there is a close correlation between a polarised discourse structure and accessibility domains for anaphora resolution. The theory also provides support for a generalisation of Centering Theory (Grosz, Joshi and Weinstein, 1995) (CT) beyond its original local settings. We are currently using GLOSS for testing VT.

The specific scheme proposed is a multi-level (hierarchical) parallel annotation architecture compatible with the data architecture defined in the CES (Ide and Priest-\(^\)\)

\(^4\) A left-to-right reading of the terminal frontier of the tree must correspond to the span of text covered in the same left-to-right order.
Dorman, 1996). The overall hierarchical architecture of this schema is a DAG of views as in figure 4. Following is a short description of each of them.

Figure 4: The hierarchy of views for the validation of VT

BD: the base document, the un-annotated text or possibly containing markup for basic document structure down to the level of paragraph;

RS-VIEW: marking isolated reference strings;

U-VIEW: marking discourse units (maximally sentences, but also clauses);

RS-IN-U-VIEW: reference-in-units view, inherits from both RS-VIEW and U-VIEW. It includes markup for referring strings within the discourse units;

RL-VIEW: reference links view, imposed over the RS-VIEW, includes reference links between an anaphor (source) and a referee (target);

REL-VIEW: relation’s view, reflects the discourse structure in terms of a tree-like representation. To obtain the REL-VIEW the interactive tree building feature is used;

CF-VIEW: marking a list of forward looking centers for each unit, according to CT;

CCT-VIEW: the classical CT-view. Could be used to evaluate transitions following classical CT. This view would encode backward looking centers (CBs) which could then be used to compute transitions between adjacent units. Also the global smoothness score following CT (Cristea, Ide and Romary, 1998b) can be computed;

VEINS-VIEW: marking veins of the discourse tree (Cristea, Ide, Romary, 1998b);

HCT-VIEW: the hierarchical CT-view. Includes markups for CBs computed along the veins of the discourse structure. It allows for the computation of the global smoothness score as given by VT, thus providing support for the validation of the claim of VT that states that the global smoothness score when computed along the veins must be at least as high as that computed following CT;

VEINS-RL-VIEW: combines the veins markups with those for reference links, thus enabling for the validation of the main claim of VT (references are possible only in their domains of accessibility).
4. Conclusions

We present an annotation tool, called GLOSS, that manifests the following features: accepts as input SGML source documents and/or their database images and produces as output source SGML documents as well as their associated database images; allows for simultaneous opening of more documents; can collapse independent annotation views of the same original document, which also allows for a layer-by-layer annotation process in different annotation sessions and by different annotators, includig automatic; offers an attractive interface to the user; permits discourse structure annotation by offering a pair of building operations (adjoining and substitution) and remaking operations (undo, delete parent-child link and tree dismember). Finally we display an example that shows how GLOSS is employed to validate, using a corpora, a theory of global discourse. The system currently runs on a PC under Windows'95 or NT.

Acknowledgements. We are indebted to Nancy Ide who gave suggestions and corrected many errors.

References


