SALTO – A Versatile Multi-Level Annotation Tool

Aljoscha Burchardt, Katrin Erk, Anette Frank*, Andrea Kowalski, and Sebastian Pado

Dept. of Computational Linguistics
Saarland University
*and DFKI
Saarbrücken, Germany
{albu, erk, frank, kowalski, pado}@coli.uni-sb.de

Abstract

In this paper, we describe the SALTO tool. It was originally developed for the annotation of semantic roles in the frame semantics paradigm, but can be used for graphical annotation of treebanks with general relational information in a simple drag-and-drop fashion. The tool additionally supports corpus management and quality control.

1. Introduction

We present SALTO, a tool for manual annotation within an intuitive, easy to use graphical environment. Its purpose is to support the annotation of a *second structural layer* on top of an existing syntactic structure. Originally developed for the annotation of semantic roles and semantic classes in the FrameNet paradigm (Baker et al., 1998), it can be used for related tasks, such as annotation of discourse structure or anaphoric relations. The key features of SALTO include:

- Query-based selection of data sets for annotation.

- Definition of tag sets for the annotation.

- Distribution of corpora to annotators.

- Comfortable annotation with visual editor and mouse-menus.

- Quality control: inspection and correction of disagreements between annotators.

Many annotation tools, e.g. MMAX (Müller and Strube, 2001), work with text-based representation and thus have to resort to bracketing to represent more complex structure. SALTO, like the Annotate tool (Brants and Plaehn, 2000), represents syntactic structure graphically, as shown in Figure 1. But while Annotate supports the graphical annotation of plain text with syntactic structure, SALTO displays a fixed syntactic structure and allows the annotation of a second layer of structure on top of the first one, with the second layer referring to arbitrary nodes of the first layer.

This paper is structured as follows: Section 2 characterizes the kind of annotation tasks that SALTO can be used for, both theoretically and via two walk-through examples. In Section 3 we list the most important features that SALTO offers to support annotation. Section 4 describes the overall workflow that SALTO presupposes and supports as well as the quality control mode of the tool. Section 5 contains details on obtaining the SALTO tool.

2. SALTO: Annotation on Top of Syntax

In this section, we describe the type of annotation tasks that SALTO supports. First, we characterize the types of annotation which SALTO can be used for, including assumptions about input data; then, we provide detailed examples for two different example tasks.

2.1. Annotation Tasks that SALTO Supports

SALTO offers a graphical environment for linguistic annotation. The tool assumes that input corpora are syntactically annotated, then adds a second layer of structure, which can refer to arbitrary nodes in the syntactic structure.

SALTO supports any annotation task which can be phrased in terms of one or more trees, as long as each tree can be anchored at some overt expression in the sentence.

SALTO accepts input in TIGER XML (Mengel and Lezius, 2000) as well as its own output format, SALSA/TIGER XML (Erk and Pado, 2004). TIGER XML conceptualizes syntactic structure as a directed graph. It is capable of describing constituents as well as dependency structure and flexible enough to handle discontinuous constituents.

Transformation from many treebank formats to TIGER XML is available via TIGERRegistry, a component of TIGERSearch (Lezius, 2002). SALTO can also handle "pseudo"-analyses of unparsed sentences, consisting only of a sentence node and the terminals, so that annotation of data without syntactic analysis is easily possible as well.

2.2. Example 1: Semantic Role Annotation

SALTO was originally developed for the manual annotation of semantic roles in the context of the SALSA project[1] (Erk et al., 2003), which aims at annotating a large German corpus with role-semantic information in the Berkeley FrameNet (Baker et al., 1998) paradigm. The FrameNet resource associates words and expressions with semantic classes called *frames* and lists semantic roles, called *frame elements*, for each semantic class.

Figure 1 shows a screenshot of SALTO, displaying a sentence drawn from the TIGER corpus (Brants et al., 2002) and annotated with two frames: *He bought the wine bar in order to close it*.

Syntactic structure. The syntactic structure of the sentence in Figure 1 is shown as a tree with straight edges. The node labels (shown as dark circles) give the syntactic categories of constituents. Edge labels describing dependency relations can optionally be displayed but are disabled by default to avoid cluttering the picture.

---

[1]`www.coli.uni-sb.de/projects/salsa`, funded by the German Science Foundation DFG, Title PI 154/9-2.
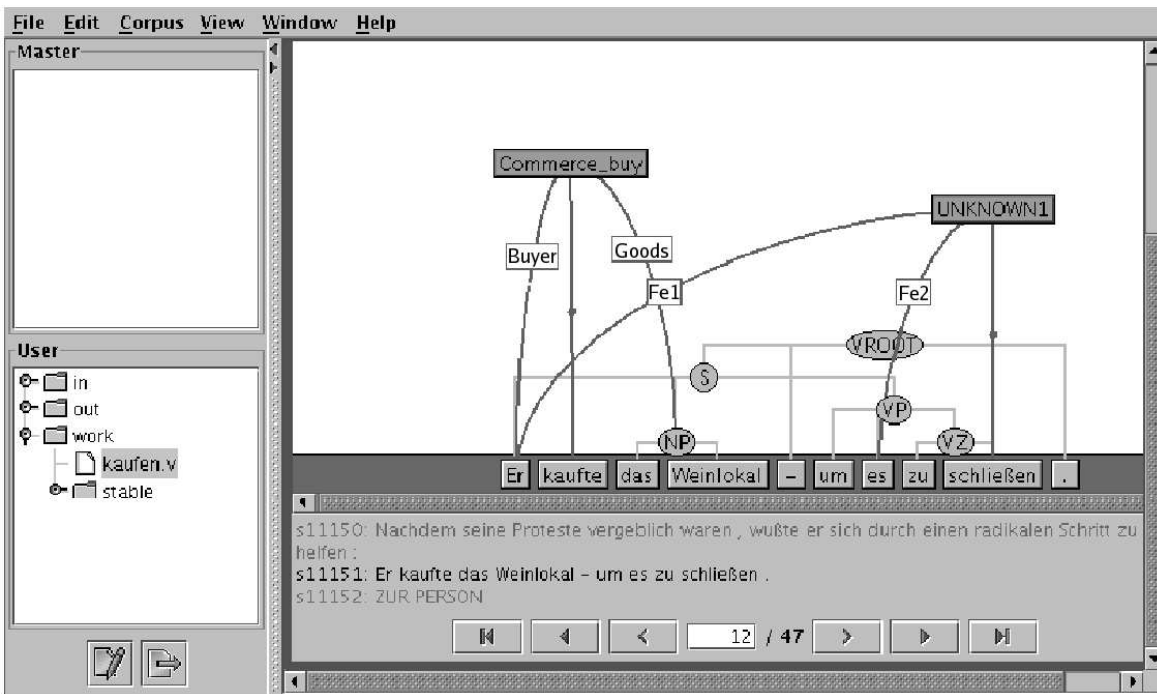
Figure 1: A snapshot of SALTO: *He bought the wine bar in order to close it.*

Semantic classes. In Figure 1, the second layer annotated on top of the syntax is shown as darker trees with bent edges. The word *kaufte* (*bought*) has been associated with the semantic class COMMERCE_BUY, a FrameNet frame. The user assigns a semantic class by right-clicking on a terminal, in this case *kaufte*, and then choosing a semantic class from a list of pre-selected candidate classes. The word *schließen* (*close*) in Figure 1 has also been assigned a semantic class, UNKNOWN1. This frame was missing in FrameNet and has been added by SALSA. It describes the event of an agent (FE1) terminally closing down some institution (label FE2).

Semantic roles. Once the user has assigned a semantic class, the semantic roles associated with the class can be assigned by simply dragging them to the appropriate node in the syntactic tree. In Figure 1, the terminal *Er* (*he*) is the BUYER of the COMMERCE_BUY event, while the NP *das Weinlokal* (*the wine bar*) constitutes the GOODS. At the same time, *Er* (*he*) is the FE1 (the agent) of the UNKNOWN1 event, and *es* (*it*) is the FE2 (the institution).

2.3. Example 2: Annotation of Discourse Relations

Figure 2 shows the use of SALTO for a different annotation task: the annotation of discourse connectives. In this setting, the assigned tags describe not word sense, as in the previous example, but types of conjunctions. The sentence,

[Mary went to the party] although [she was tired].

is taken from the Penn Discourse Treebank tutorial[2] and has been parsed automatically using Collins' (1997) parser.

The connective *although* has been assigned the label *Subordinate Conjunction*. The second argument ARG2 points to the constituent *she was tired*, the first argument ARG1 points to three constituents, *Mary*, *went*, and *to the party*, as there is no single constituent in the parse tree for *Mary went to the party*.
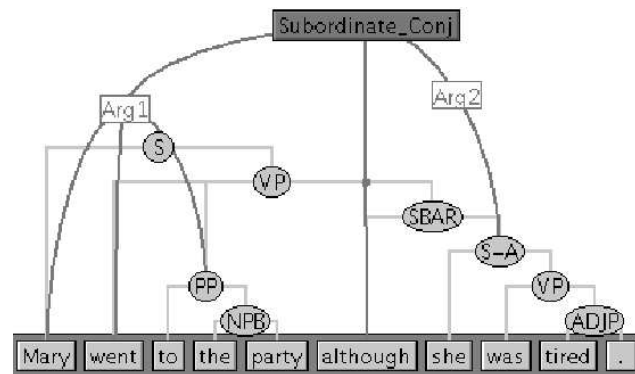

Figure 2: Annotation of discourse relations.

3. SALTO Annotation Features

In the previous section we have described two quite simple and straightforward annotation examples. This section gives a brief overview of the most important additional features SALTO offers for annotation. For a complete and in-depth description, the reader is referred to the SALTO manual available at our project page.

Deep or flat structure. SALTO offers annotation in a "flat tree" mode as well as a "deep structure" mode. In "flat tree" mode, as shown in Figure 1, edges of the second annotation layer always point to nodes of the syntactic structure, yielding a set of unconnected trees of depth one. In "deep

structure" mode edges of the second annotation layer can point to the syntactic structure or to nodes of the second layer. Figure 3 shows an example of embedded annotation we discussed in Burchardt et al. (2005b):

> *The 28-year-old Moroccan was **found** guilty as an **accessory** to **murder** in more than 3000 cases.*

The nested dependency of *find* and *accessory* and *accessory* and *murder* is expressed in an embedded frame structure: the frame VERDICT embeds the frame ASSISTANCE in its CHARGES role. In turn, ASSISTANCE embeds KILLING via its FOCAL_ENTITY role.

Discontinuous annotation. A single label may apply to more than one node, e.g. in the case of intersecting hierarchies. The example in Figure 2 demonstrates this flexibility of annotation: *Mary went to the party* can be annotated although it is no single constituent in the parse tree. Discontinuous structures in general can be treated this way.

Context sentence annotation. SALTO allows access to an arbitrarily large context window surrounding the current sentence. First, the context can be viewed to support annotation decisions, e.g. in Figure 1, one sentence before and after the current sentence are displayed in light gray text color at the bottom of the main window. Second, annotation can extend into context sentences: if an annotator draws an edge into the corner of the main window, the display jumps to the next (previous) context sentence(s). This allows to annotate inter-sentential dependencies.

Underspecification. It is agreed upon that it is not always possible to assign a single clear-cut tag in semantic annotation (e.g. (Kilgarriff and Rosenzweig, 2000)). In order to deal with vagueness and ambiguities in a principled way, SALTO allows the annotator to assign multiple annotations to the same markable and then join them into an "underspecification" set. This saves the annotator from making impossible decisions and makes it possible to access all annotation alternatives in later processing stages.

Tagset definition. The tagset for a corpus to be annotated can be pre-specified during corpus creation (see next section). In addition, annotators can define new tags on the fly during annotation.

Flag assignment. Sentences as well as nodes and edges of the annotated structure can be associated with flags. For example, SALSA annotation uses node flags for marking metaphoric usages of semantic classes. While an initial set of flags can be pre-defined, annotators can modify and add to this set according to their needs.

4. Workflow and Quality Control

In addition to the annotation process itself, SALTO supports the selection of sentences for annotation, the distribution of annotation datasets to annotators, the collection of annotated datasets, and the manual inspection and correction of inter-annotator disagreements. Figure 4 sketches the basic workflow assumed by the tool.

Selecting data for annotation. SALTO offers an inbuilt interface to TIGERSearch (Lezius, 2002), a search engine for treebanks. Provided that the corpus to be annotated is available in TIGER XML (e.g. through conversion
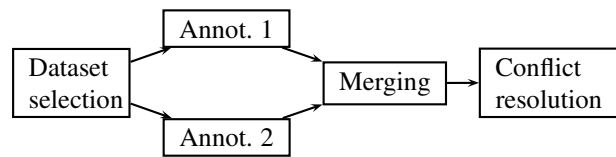

Figure 4: Typical workflow supported by SALTO

with TIGERRegistry, see Section 2.1), this interface can be used to extract datasets for annotation declaratively, using TIGERSearch queries. For example, the SALSA project annotates data one lemma at a time; hence dataset selection extracts all instances of the lemma under consideration.

Distribution, processing and collection of annotation datasets. SALTO offers an admin mode in which a dataset selected for annotation can be distributed to (one or more) annotators, and finished datasets can be collected. Each annotator has an `in` folder holding new datasets to be annotated, a `work` folder, and an `out` folder in which finished datasets can be placed, as shown in the lower left-hand corner of Figure 1.
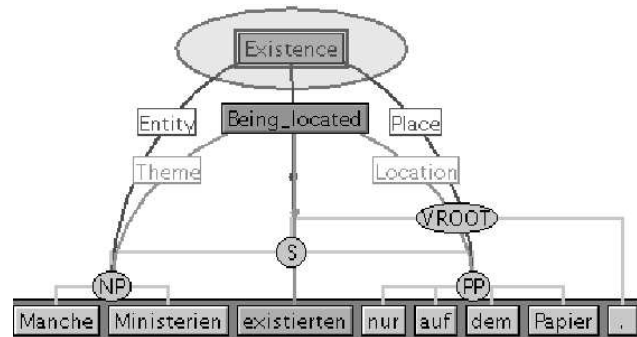

Figure 5: Quality control: inter-annotator difference between semantic classes 'Existence' and 'Being_located'.

Quality control. If the same dataset has been annotated independently by two different annotators, the two versions can be merged into a single set in which inter-annotator disagreements are represented and highlighted. In quality control mode, the SALTO tool walks the user through those differences for manual inspection and correction.

Figure 5 shows an example of an inter-annotator disagreement: the sentence *Manche Ministerien existieren nur auf dem Papier* (*Some ministries exist only on paper*). One annotator has tagged the word *existieren* (*exist*) with the semantic class EXISTENCE, while the other annotator has chosen BEING_LOCATED. The tool has circled EXISTENCE to show that this is the next annotation choice to be either confirmed or denied by the user.

5. Obtaining SALTO

SALTO was implemented by a team at CLT Sprachtechnologie GmbH[3] under the direction of Daniel Bobbert. It is implemented in Java using the Swing library for the GUI. This operating system-independent design allows the application to be run on any platform with a recent Java runtime
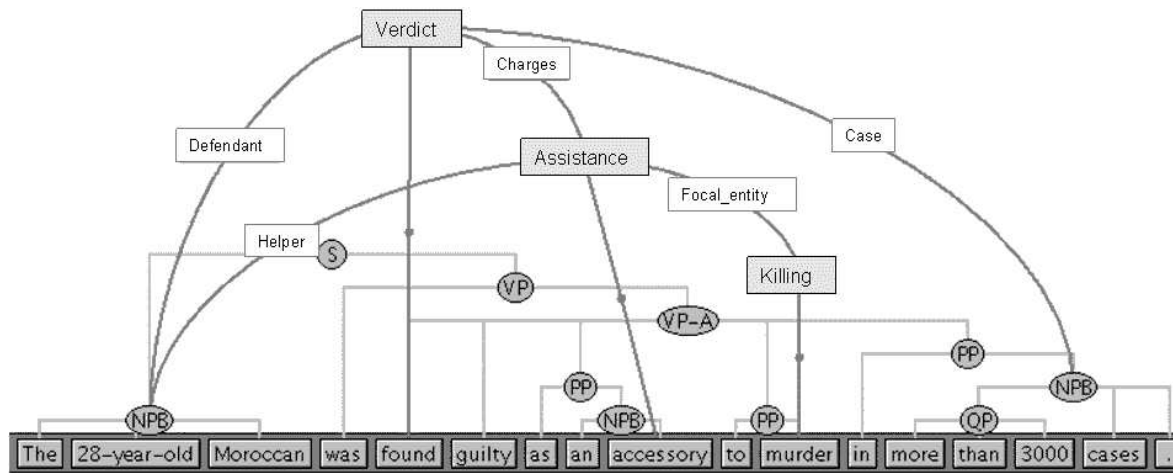
---

[3] www.clt-st.de/

Figure 3: Embedded semantic annotation.

environment (>1.3). We have tested it successfully under Windows, Linux, SunOS and Mac OS X.

SALTO is available free of charge for academic research. It can be downloaded from our project page[4]. We encourage user feedback, which can – to some degree – guide our further development of SALTO.

6.   Conclusion and Outlook

We have presented SALTO, a tool for graphical annotation of a second structural layer on top of a syntactic structure given in TIGER format. It supports corpus management, quality control, i.e. resolution of inter-annotator disagreements, and offers a number of special annotation features, among others underspecified tags and role assignment beyond the sentence boundary. Its open architecture makes it possible to use it for various annotation tasks.

SALTO has been adjusted to new needs from the SALSA project in the past and is currently being extended with an interface to external software.

We plan to integrate SALTO into a GUI for the visualization of different stages in the process of automatic assignment of semantic roles by systems developed in the SALSA project: the shallow semantic parser SHALMANESER (Erk and Pado, 2006), and the WordNet based "Detour (to FrameNet)" system (Burchardt et al., 2005a).

7.   References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of COLING-ACL 1998*, Montréal, Canada.

Thorsten Brants and Oliver Plaehn. 2000. Interactive corpus annotation. In *Proceedings of LREC 2000*, pages 453–459, Athens, Greece.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol.

Aljoscha Burchardt, Katrin Erk, and Anette Frank. 2005a. A WordNet Detour to FrameNet. In Bernhard Fisseni,

Hans-Christian Schmitz, Bernhard Schröder, and Petra Wagner, editors, *Sprachtechnologie, mobile Kommunikation und linguistische Resourcen*, volume 8 of *Computer Studies in Language and Speech*, pages 408–421. Peter Lang, Frankfurt am Main.

Aljoscha Burchardt, Anette Frank, and Manfred Pinkal. 2005b. Building text meaning representations from contextually related frames – a case study. In *Proceedings of IWCS 2005*, Tilburg, The Netherlands.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL/EACL 1997*, pages 16–23, Madrid, Spain.

Katrin Erk and Sebastian Pado. 2004. A powerful and versatile XML format for representing role-semantic annotation. In *Proceedings of LREC 2004*, Lisbon, Portugal.

Katrin Erk and Sebastian Pado. 2006. Shalmaneser – a toolchain for shallow semantic parsing. In *Proceedings of LREC 2006*, Genoa, Italy.

Katrin Erk, Andrea Kowalski, Sebastian Padó, and Manfred Pinkal. 2003. Towards a resource for lexical semantics: A large German corpus with extensive semantic annotation. In *Proceedings of ACL 2003*, pages 537–544, Sapporo, Japan.

Adam Kilgarriff and Joseph Rosenzweig. 2000. Framework and results for English Senseval. *Computers and the Humanities*, 34(1-2).

Wolfgang Lezius. 2002. TIGERSearch - ein Suchwerkzeug für Baumbanken. In *Proceedings of Konvens 2002*, Saarbrücken.

Andreas Mengel and Wolfgang Lezius. 2000. An XML-based encoding format for syntactically annotated corpora. In *Proceedings of LREC 2000*, pages 121–126, Athens, Greece.

Christoph Müller and Michael Strube. 2001. Mmax: A tool for the annotation of multi-modal corpora. In *Proceedings of the 2nd IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, pages 45–50, Seattle, Wash.