Documentation of SALTO (aka SALSA Tool)

SALSA Project Saarbrücken

http://www.coli.uni-saarland.de/projects/salsa/

September 4, 2007

Contents

1 Introduction

SALTO is a graphical tool that supports manual annotation of text corpora and annotation management. It has been developed and used by the SALSA project[1] for semantic annotation of corpora in the frame semantics paradigm ([3]). More precisely, the tool can be used to add a second (typically semantic) layer of annotation to corpora that are already syntactically analyzed (through manual annotation or automatically). Main features are:

- Query-based creation of subcorpora for annotation.
- Distribution of corpora to different annotators.
- Definition of Items and Classes/Tags to be annotated.
- Comfortable annotation with visual editor and mouse-menues.
- Semi-automatic merging and adjudication of parallel annotations in same editor.

[1] `http://www.coli.uni-saarland.de/projects/salsa/`

2 Example Annotation Session

Before we explain the various features of the tool in more detail, we illustrate the basic graphical annotation mode by way of a prototypical example annotation session.

Opening a corpus for annotation. Suppose you have a corpus in SALSA/TIGER XML format (see Section 8) in your working directory (in our example the file `kaufen.v`). By double-clicking, you open the corpus for annotation. (see Figure 1).

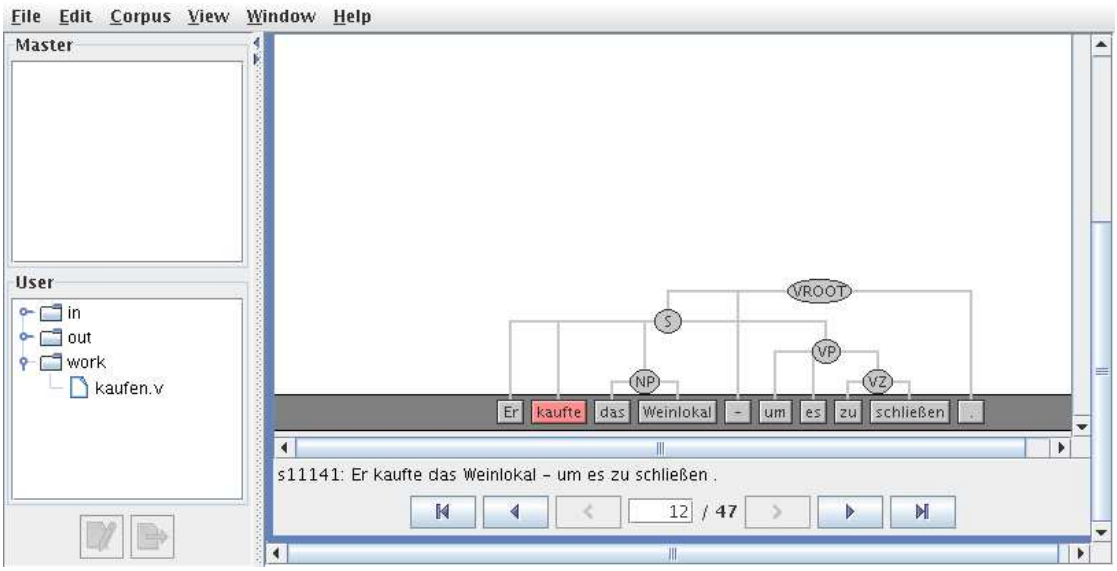

Figure 1: Corpus opened for annotation.

Frame and frame element annotation. By right-clicking a word you get a menu Choose `Invoke frame` and select the frame you want to annotate. In our example, we chose the frame COMMERCE_BUY for the word *kaufen* (see Figure 2).

Next, you can annotate frame elements by dragging the chosen frame element to a word, constituent or a set of words of the sentence. In our example, we dragged BUYER onto the word *Er* and GOODS onto the NP *das Weinlokal*. (see Figure 3).

If you are finished with the annotation of this sentence, you can move to the next sentence by clicking on the (filled) right arrow and continue annotation.

Saving and quitting. Choosing `File`⇒`Quit` from the menu you are asked if you want to save your changes. Click 'Save' if you want to. Then the session is closed.

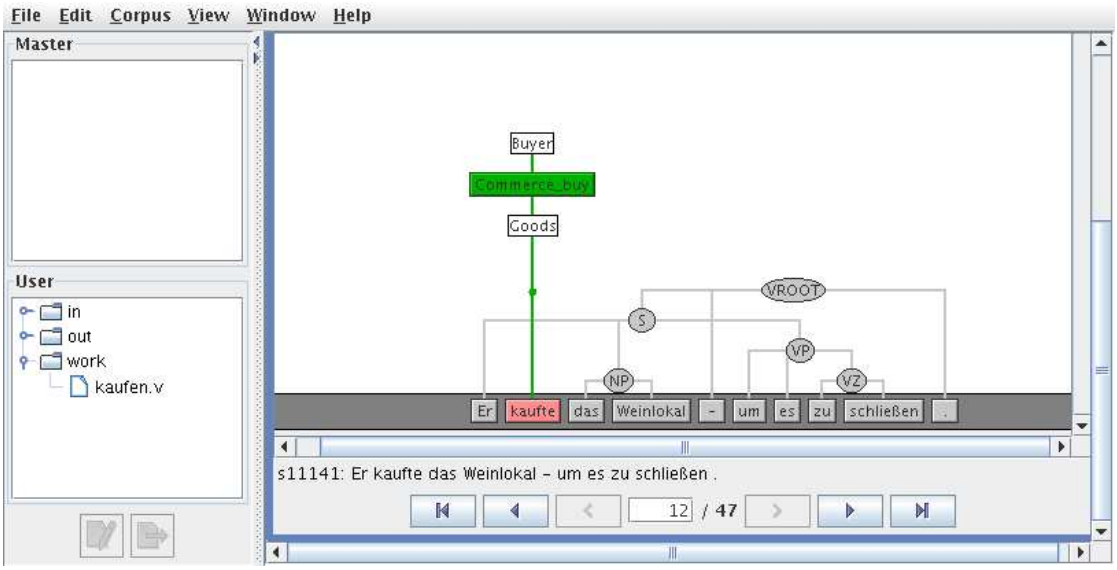

Figure 2: Frame annotation.

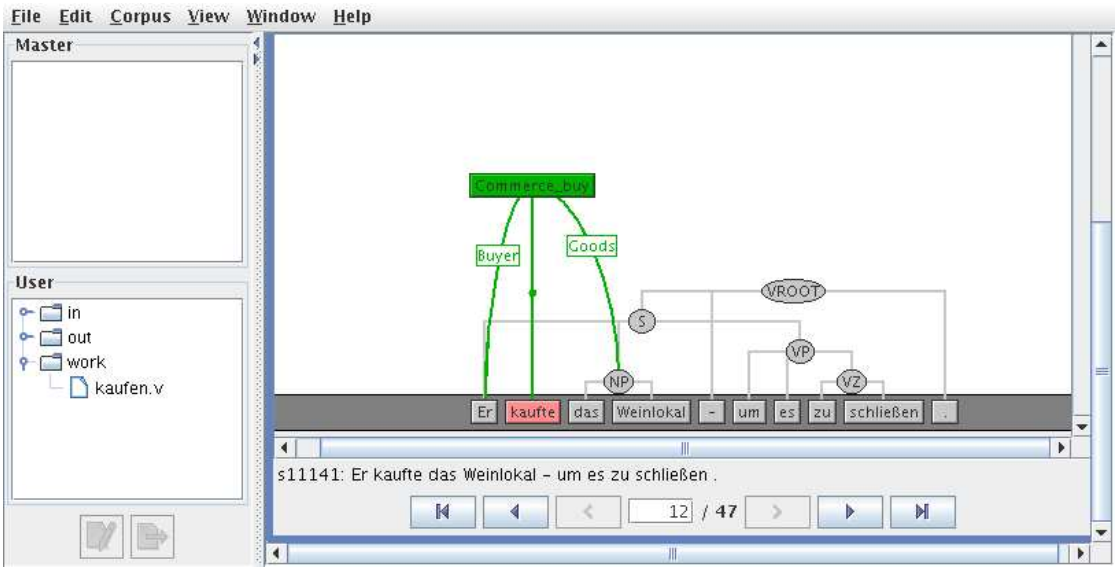

Figure 3: Frame element annotation.

3 Installation and Corpus Formats

Data formats

The tool expects TIGER XML [5] as input format for the corpora to be annotated. TIGER XML describes trees with node and edge labels. Secondary edges introduce full DAG power. So, the base format for the corpora to be annotated is highly flexible. The TIGERRegistry adminstration tool which is available at `http://www.ims.uni-stuttgart.de/projekte/TIGER/` can be used to import other corpus formats.[2]

The structures one can annotate on top of this input syntactic structure are flat tree structures or else embedded structurs, see Section 6.5.

Supported annotation tasks

In the SALSA project, the tool is used to annotate syntactically annotated corpora using Berkeley FrameNet [1] frames and roles.

In principle, any annotation task that can be phrased in terms of flat trees is supported by the tool. For example, if one wants to annotate co-reference, one could annotate structures with edge labels like `anaphor` and `antecedent`.

3.1 Installation

System requirements

The SALSA tool is written in Java and thus platform-independent. Java can be downloaded for free from `http://java.sun.com/`.

Installation

1. Copy the zipped tar archive to a place of your choice.
2. Unzip the archive.
3. Make sure that you have a current Java installation.
4. Start the salsa tool by running

   Windows: `Salto.bat`

   Unix/Linux: `Salto.sh`.

File structure

Corpora can be opened and saved anywhere in the file system via a regular file browser window.

[2] See `http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/doc/html/TIGERRegistry.html`.

By default, any corpora that are edited within the standard workflow (see the next Section) of the tool are located in directories that are located below the `repository` in the main directory of the tool (to change this, login as 'Administrator' and chose `Edit`⇒`Setup`).

As can be seen in Figure 4, each user has three working directories: `in`, `out`, and `work`. The purpose of these directories is further explained in the following section.

The `corpora` directory contains the main (reference) corpus or corpora in TIGER-Registry format, see Section 5.1 for details.

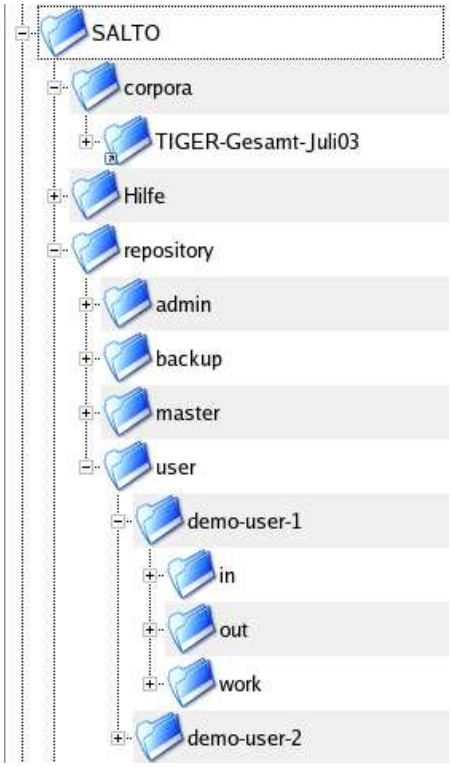

Figure 4: Directory structure with two users and main corpus 'TIGER-Gesamt-Juli03'.

4 Workflow

4.1 User model

There are two types of user modes:

User(s) do the annotation.

Administrator(s) ("Admin") supply corpora and control the annotation results.

User mode is the default. Admin mode is entered by ticking the box labelled "Administrator" on the login screen.

4.2 The complete annotation process

The standard annotation process includes three steps:

Corpus creation: Admin extracts one or more files (sub-corpora) to be annotated out of a given corpus.

- Standard means for this task is TIGERSearch which is incorporated into the SALSA tool and supports queries to the corpus. The query results are written to new files (i.e. sub-corpora).
- The sub-corpora are then distributed to the annotators' `in` directories.

Annotation: Users find new subcorpora in their `in` directories and move them to their `work` directories for annotation. The annotation proceeds one file (sub-corpus) at a time.

- The words/constituents to be annotated are chosen by the User during annotation with the mouse.
- A file-specific list of structures to be annotated (e.g. frames in SALSA)
  - has either been defined by Admin when file was generated or
  - is specified/modified during the annotation process by the User
- After the annotation is finished, the User moves the annotated subcorpus to the `out` directory.

Adjudication: Admin collects the annotated sub-corpora from the Users' `out` directories. These are then adjudicated, i.e. checked for correctness within the tool. Currently, two annotations of the same sub-corpus can be merged into one final version at a time.

More details for these steps can be found in subsequent Sections 5 (corpus creation), 6 (annotation), and 7 (Adjudication).

5 Corpus Management

5.1 Corpus creation

The tool expects TIGER XML [5] as input format of the corpora to be annotated. The TIGERRegistry adminstration tool which is available online can be used to import various corpus formats.[3]

Each resulting TIGER-XML corpus is stored in a "corpus directory". In the SALSA tool the "master" directory containing such "corpus directories" has to be specified via the menue `Edit`⇒`Setup`. Note that you must not choose the TIGER-XML corpus directory itself here but the 'master' directory containing it (see Figure 4). Otherwise the corpus is not found by the tool.

5.1.1 Subcorpus creation

TIGERSearch is incorporated into the tool and can be used extract sub-corpora to be annotated from a "main"-corpus. For example, to create a subcorpus of all sentences containing the noun *Reform*, one has to:

1. Login as Admin.
2. The directory containing the main corpus (in a subdirectory in TIGER Registry format) has to be specified via the menue `Edit`⇒`Setup`.
3. Open TIGERSearch query window: `File`⇒`Create Subcorpus`. If there is more than one corpus in the directory specified above, choose the one to be taken.
4. Enter a query like `[word = /Reform/ & pos=/N.*/]`.
5. Enter a name for the resulting sub-corpus.

The resulting subcorpus then appears in the Admin-window and can be opened.

5.1.2 Query Format

A documentation of TIGERSearch including the query format can be found online[4]. Figures 5 and 6 give an impression of the kinds of queries that are used in the SALSA context.

5.1.3 Specifying what (which frames) to annotate

Open the (sub-)corpus and select `Corpus`Edit Frames. Then choose

- `Add Frames` and select predefined frames (from the file `frames.xml`, see 8.4) or
- `Create New Frame` and specify name and desired roles.

[3] See `http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/doc/html/TIGERRegistry.html`.

[4] `http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/manual_html.html`

```
[word = /([Bb]elausch(e|st|est|t|en|et)?|
          [Bb]elausch(te|test|tet|ten)|
          [Bb]elauschend(e|es|er|en|em)?|
          [Bb]elauscht(e|es|er|en|em)?)/ & pos=/V.*/]
```

Figure 5: Query for all forms of the verb *belauschen*.

```
([word=/([Aa]nstarr(e|st|t|en|et)|
         [Aa]nstarr(te|test|tet|ten)|
         [Aa]nstarrend(e|er|es|en|em)?|
         [Aa]ngestarrt(e|er|er|en|em)?|
         [Aa]nzustarren)/ &pos = /V.*/])|
((#nl:[cat="S"]>
[word=/([Ss]tarr(e|st|est|t|en|et)?|
        [Ss]tarr(te|test|ten|tet))/ &pos = /V.*/])&
(#nl> [word=/an/ &pos =/PTKVZ/]))
```

Figure 6: Query for all forms of the verb *anstarren* with separable prefix *an*.

5.2 Corpus distribution

An administrator can publish a subcorpus, i.e. assign it to one or more users by moving the respective file to the users' **in-**directories. In order to work on a file, the user then has to move it to his **work-**directory. If a user has finished annotation of a corpus, he can move it to the **out-**directory where it can be recollected by the administrator.

5.2.1 Publishing a subcorpus

To publish a subcorpus, select it in the Admin window and click on it with the right mouse-button. Select `Publish...`. Choose one or more users from the list. To select a range of users, keep the Shift-key pressed while pressing the left mouse button. To select individual users, keep the Ctrl-key pressed. A copy of the subcorpus is moved to the in-folder of the user.

5.2.2 Collecting a subcorpus

Every published subcorpus is displayed in blue in the Admin window. The administrator can (re-)collect files from the users by marking them, clicking the right mouse button and selecting `Collect`. Typically, published corpora are re-collected from the user's out-directory after annotation is finished. If they are still in the user's in- or working-directory, a warning is displayed.

6 Advanced aspects of annotation

This section describes detailed aspects of the annotation process introduced in Section 2. It assumes that you are in user mode (see Section 4). The aspects covered here are:

- Loading other corpora
- Extending the annotation scheme
- Annotating special phenomena
- Annotating context sentences

6.1 Accessing corpora through the menu

In addition to using the 'official' repository structure shown at the left on the screen, the tool can be used to annotate XML files at arbitrary locations. Choose `File`⇒`Open File` from the menu to open a new file.

6.2 Customising corpus display

The view on the current subcorpus can be customised in the `View` menu.

6.3 Dynamically extending the annotation scheme

The User can extend the annotation scheme (i.e. the entities which can be annotated) by new frames, frame elements and flags. The newly added (or changed) frames/elements/flags can then directly be assigned in the currently opend sub-corpus.

Adding new frames. Clicking `Corpus`⇒`Edit Frames` opens a menue. `Add Frames` lets the user select predefined frames from the file `frames.xml` (see Section 8.4). `Create New Frame` lets the user define new frames (default name is *UNKNOWN*) and their elements. By selecting some frame(s) in the top menue, the buttons `Edit Frame` and `Delete Frames` become active and frames can be changed or deleted again.

Adding new frame elements. Frame elements can be edited by choosing `Corpus`⇒`Edit Frames`, marking the respective frame and then choosing `Edit Frame`. Elements can then be added (`Add Element`) or selected elements deleted (`Delete Elements`).

Adding (new) flags. `Corpus`⇒`Edit Frame Flags` opens a menue where flags can be added or deleted. The sub-menues `Frame`, `Frame Element`, and `Target(FEE)` switch to the respective flag lists. Flags can be added (`Add Flag`) or selected flags can be deleted (`Delete Flags`).

6.4 Annotating of special phenomena

Underspecification for Frames and FEs. Frames (elements) can be annotated as underspecified in cases where an annotator cannot decide whether or not to annotate a frame (element) or which of multiple suitable frames (elements) to annotate. To mark a frame (element) as underspecified, select the frame (element) and click `strg-u`. Underspecified frame (elements) appear in blue color. This default color can be changed under `Edit`⇒`Preferences` and `Underspecification`. See Section 8.3 on how underspecification is coded in the xml files.

Flags for Frames and FEs Frames (and likewise frame elements) can be flagged by right-clicking the frame and selecting `Frame Flags` if flags have been definied for the current sub-corpus as explained in Section 6.3.

Flags for Sentences Sentences can be annotated with flags and it is possible to show only flagged sentences. To annotate a sentence with a flag, choose `Corpus`Flag Current Sentence (or right-click on the black square in the upper left of the main window). Either choose a pre-defined flag or add a new flag by clicking `Corpus`⇒`Edit Current Sentence Flags`.

To show only flagged sentences, select the respective entry from the pull-down menue in the upper left of the main window with the default *All sentences*. Four flags have been pre-defined by SALSA:

REEXAMINE for sentences that need to be reexamined,

WRONGSUBCORPUS for sentences that have been falsely included in the sub-corpus,

INTERESTING , and

LATER for annotations that are postponed.

If a sentence is flagged, the respective part of the black box in the upper left of the main window changes color.

Split words. If only a part of a word is to be annotated, like e.g. the noun *programm* within the German noun composite *Parteiprogramm*, the word can be split. To do so, right-click on the word and choose `Split`. In a sub-menue, split marks can be entered. See Section 8.3 on how split words are represented in the xml files.

6.5 Annotation of embedded frame semantic structures

It is possible to allow frame elements to point to other frames, i.e. to build nested frame structures. This feature is enebles by choosing `Edit`⇒`Preferences`⇒`Frames can be elements of other frames`.

6.6 Annotating context sentences

When loading a new subcorpus, the tool checks if the complete corpus from which the current subcorpus has been extracted is installed. If this is the case, *context annotation* is possible. This means that the User can use the 'angle bracket'-style buttons (directly left and right of the sentence index) to browse the sentences preceding and following the current sentence in the original corpus. These sentences can also be annotated. If context annotation is possible, dragging a frame element will make arrows appear in the upper left and right corners of the screen (see Figure 7). Dragging the frame element

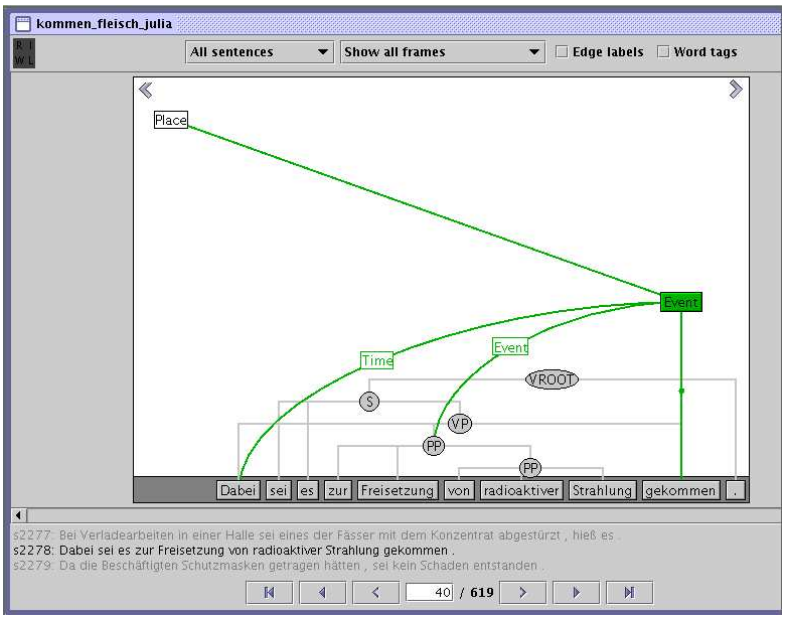

Figure 7: Dragging FE towards contex sentence.

onto on of these arrows will 'beam' it into the respective adjacent sentence (see Figure 8). One or more context sentences can be dislpayed before/after the textual representation of the current sentence in the lower part of the window. The number can be defined under `Edit`⇒`Preferences`⇒`Number of context sentences`.

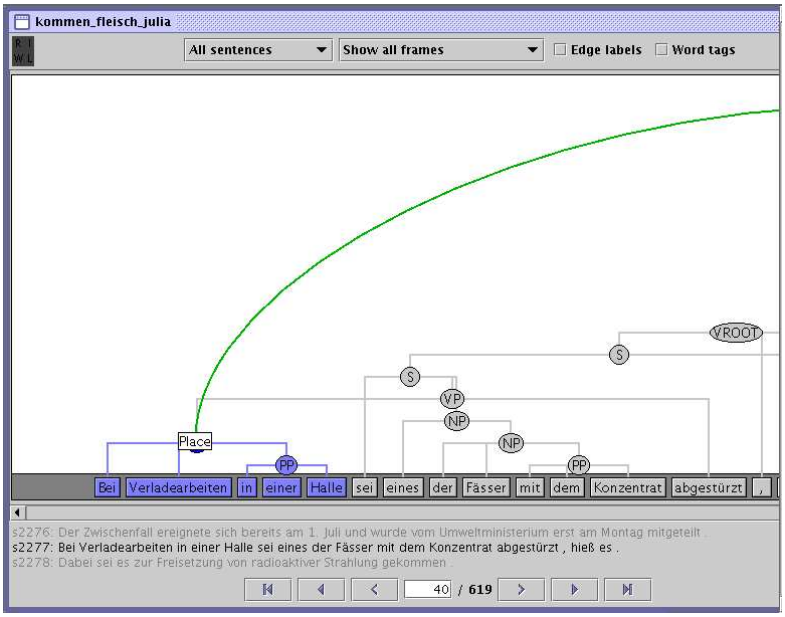

Figure 8: Dragging FE onto constituent of contex sentence.

7 Merging and Adjudication

7.1 What is adjudication?

The SALSA tool supports adjudication of corpora that have been independently annotated by two annotators.

Merging In a first step, the two corpora are *merged*. The output of this step is a single corpus that integrates the annotations of both annotators, including the explicit representation of divergences. That is, annotations that are identical in both corpora will be represented as such in the resulting corpus. Annotation differences will be represented as disjunctive.

Adjudication In the second step, an *adjudicator* investigates the marked differences, and resolves the divergences. The tool supports the process of selecting one or the other of the diverging annotations. At the same time, it offers all the standard annotation facilities. That is, in cases where neither of the two annotations are acceptable, both versions can be deleted and substituted by another annotation. In the same way, if the divergences motivate the need to underspecify the distinct readings, underspecification of frames and frame elements can be specified as usual (see 6.4).

7.2 Merging corpora

To merge two corpora, you have to be logged in as Administrator and the corpora must be in your `admin` directory. Mark both corpora in the Admin-window of the SALSA tool by clicking their names while holding the (ctrl) key. Clicking the right mouse button then opens a menue where you can choose `Merge Two Corpora`.

7.3 Adjudicating corpora

The merged corpus contains all annotated sentence in the original sequence, whether or not they contain divergences. To allow efficient search for differences, you can chose the `Corpus`⇒`Find First Difference` menue entry or type (ctrl-d) to automatically jump to the next annotation divergence in the corpus sequence. Alternatively, you can choose `Corpus`⇒`Find Next Difference` (ctrl-shift-d) to skip earlier divergences.

7.3.1 Viewing and resolving divergences

Annotation differences can be complete frames, or one or more frame elements within a frame. Annotation differences are highlighted by different colors as can be seen in Figures 9 and 10. One color displays the annotation of one annotator, the second color displays the annotation of the second annotator. The default colors for highlighting differences can be user-defined in the `Edit`⇒`Preferences` menu. Note that single underspecified FEs are marked by a blue role label. This color can also be customized.

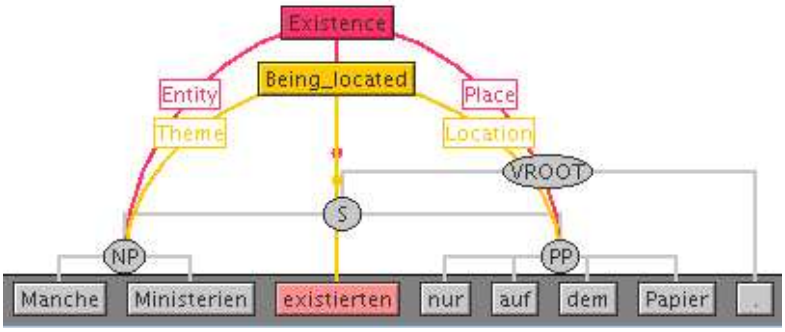

Figure 9: Annotation difference between Frames 'Existence' and 'Being_located'.

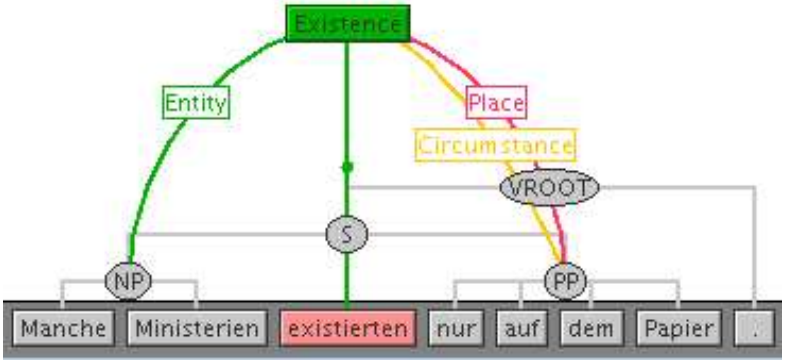

Figure 10: Annotation difference between FEs 'Place' and 'Circumstance'.

The currently active pieces of the annotation divergences can be approved (`Corpus⇒Accept`, or ctrl-j) or rejected (`Corpus⇒Discard`, or ctrl-n). Once the acceptance or rejection is processed, the color of the annotation changes, and the tool jumps to the next annotation divergence within this sentence[5], again highlighting the active choice. This automatic continuation can be switched off by selecting 'Ask before switching...' in `Edit⇒Preferences`.

Once the adjudicator clicks on some non-active annotation object (node or arc), the acceptance/rejection feature is inactive again. It will be activated again by pressing Ctrl-d (or chosing `Find Next Difference`).

7.3.2 XML representation of merged corpora

Representation of merging information in XML files. Annotation differences are currently marked 'source=1' and 'source=2'. Annotations from the file which was loaded for merging first are marked 'source=1', annotations from the second file are marked 'source=2'. An example is given in Figure 11: annotator 1 chose two contituents as

[5] or to the next annotation difference in some following sentence if all divergences of the current sentence have been resolved.

ADDRESSE while annotator 2 only chose one of them.

Once the adjudicator has resolved a divergence, the 'source' attribute of the accepted piece of annotation is deleted; the rejected piece of annotation is entirely removed.

```
<frame name="Request" id="s13065_f0">
  <target>
    <fenode idref="s13065_2"/>
  </target>
  <fe name="Addressee" id="s13065_f0_e2" source="1">
    <fenode idref="s13065_500"/>
    <fenode idref="s13065_502"/>
  </fe>
  <fe name="Addressee" id="s13065_f0_e7" source="2">
    <fenode idref="s13065_502"/>
  </fe>
  <fe name="Message" id="s13065_f0_e1">
    <fenode idref="s13065_505"/>
  </fe>
</frame>
```

Figure 11: Merged subcorpus sample with divergence in ADDRESSE-anntoation-

Representation of some special types of divergences Flags optional frame elements (arcs), underspecification, other?

[with screen shot]

7.4 Known problems, envisaged extensions and modifications

Representing specific types of differences Splitting FEs to point to different constituents is not possible if one of these constituents is located in a different sentence. This will be fixed in a future version of the tool. For the time being, double anntotation of the same role seems a good alternative.

8 Interfaces

8.1 Salsa/Tiger XML

In this section we present the TIGER/SALSA XML format. First we discuss the abstract model of syntactic and semantic information underlying the format, then we describe the XML representation. In Figure 15 in the Appendix, you can find an example XML representation extract.

8.2 The underlying model

The syntactic level of TIGER/SALSA XML corresponds to the syntactic representation of TIGER XML [5]. The syntactic structure is a tree with both node and edge labels[6]. Trees can contain crossing edges to encode discontinuous constituents.

On the role-semantic level, we model each frame instance as a *frame tree* of depth one with a root labelled with the frame name. A frame tree has at least one edge that points to the frame-evoking element, which is unlabelled in the graphical representation. All other edges point to frame elements and are labelled accordingly.

In order to make the annotation more flexible, we keep all frame trees separate. This means that leaves of frame trees are nodes of the syntactic structure. In principle, however, frame tree leaves could also figure as roots of other frame trees, resulting in a nested semantic structure.

In addition, a model for exhaustive semantic annotation must also be able to encode the following complications:

- A frame element may consist of more than one constituent. Consequently, our model (see [2]) allows for frame trees in which multiple edges bear the same label.
- A frame element or target may consist of only part of a word in the case of (German) compound nouns. For example, the German compound *Mietrechtsdiskussion* (tenant law discussion) contains both the target (*diskussion*) that introduces a `Conversation` frame and its `Topic` role (*Mietrecht*). Therefore, our model is able to make reference to sub-word units.
- A frame element may be situated in a different sentence than the target, as often happens with conversation frames. Hence, frame trees can refer to entities in adjacent sentences.
- At times, the meaning of a sentence is ambiguous or vague, and annotators cannot commit to a single tag. For these situations, the model allows to tag multiple annotation referring to the same entity as *underspecified*, both on the level of frames and the level of frame elements. Consistent with [4], it is left to the user how to interpret this representation (e.g. as disjunction or conjunction).

[6] Secondary edges, used to model ellipsis, raise the proper descriptive power to DAGs.

8.3 The representation

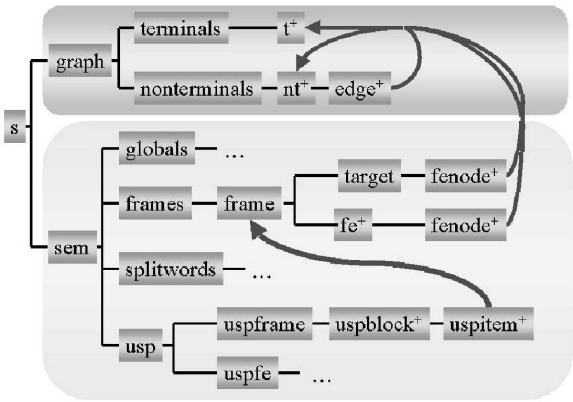

Figure 12: Structure of a TIGER/SALSA XML sentence

The tree in Figure 12 shows the implementation of this model in TIGER/SALSA. The nodes correspond to XML elements, and the edges to permissible embeddings. Elements that may be repeated are marked with a "+".

A sentence (an `<s>` element) has two parts, one for the syntactic structure, (the `<graph>` element in the upper part) and one for the semantic roles (the `<sem>` element in the lower part). In TIGER XML without semantic annotation, a sentence has only one child, `<graph>`. It lists each terminal node as a `<t>` element below `<terminals>`, and each nonterminal node as a `<nt>` element below `<nonterminals>`. Edges are realised not via XML element embedding, but with explicit `<edge>` elements that refer to nodes via their unique identifiers, depicted as arrows in Fig. 12 . This allows for crossing edges and hence for a uniform treatment of continuous and discontinuous constituents.

TIGER/SALSA XML adds a layer of semantic information by introducing the additional semantics element `<sem>`, leaving the syntactic representation in `<graph>` unchanged. `<sem>` contains a straightforward representation of the semantic annotation for the current sentence, as modelled in Section 8.2. Again, all references to (either syntactic or semantic) entities are expressed in terms of identifiers to keep the levels of representation separate.

The `<frames>` element contains the role-semantic information proper. Similar to syntax, nodes and edges of frame trees are represented as explicit elements `<frame>`, `<target>` and `<fe>`. For all semantic nodes and edges, we introduce new globally unique IDs, such that semantic roles crossing sentence boundaries do not need special treatment, as reference to unique

The `<globals>` element contains tags such as 'is metaphoric' or '(needs) reexamination'. In the `<splitwords>` element, we record the treatment of German compound nouns, effectively introducing new terminal nodes "below" the original terminals. Underspecification is recorded in `<usp>`. One `<uspblock>` inside `<uspframe>` describes one case of frame underspecification, each `<uspitem>` child referring to one frame involved

in the underspecification (by its unique ID). `<uspfe>` handles frame element underspecification in the same manner.

In TIGER/SALSA XML, the different annotation levels are kept in two separate blocks. The format is not standoff in the strictest sense, as all information about a sentence is collected within one `<s>` element. However, the annotation levels could in principle be decoupled completely because all reference between annotation levels is via identifiers that are unique throughout the corpus.

8.4 The FrameNet XML format for frames `frames.xml`

A frame description in frames.xml consists of:

- frame name
- frame definition, arbitrary text
- a description of each FE, consisting of:
  - name,
  - coreness type (core, extrathematic, peripheral)
  - definition, arbitrary text
  - optionally, a list of semantic types
- a description for each lexical unit, consisting of:
  - name and part of speech,
  - a definition, arbitrary text
  - annotation counts
  - the lexeme with name and part of speech (no idea how this differs from the lexical unit itself)
  - optionally, a list of semantic types
- optionally, a list of semantic types.

Figure 13 shows a sketch of the XML structure, Figure 14 shows an extract of the `frames.xml` file.

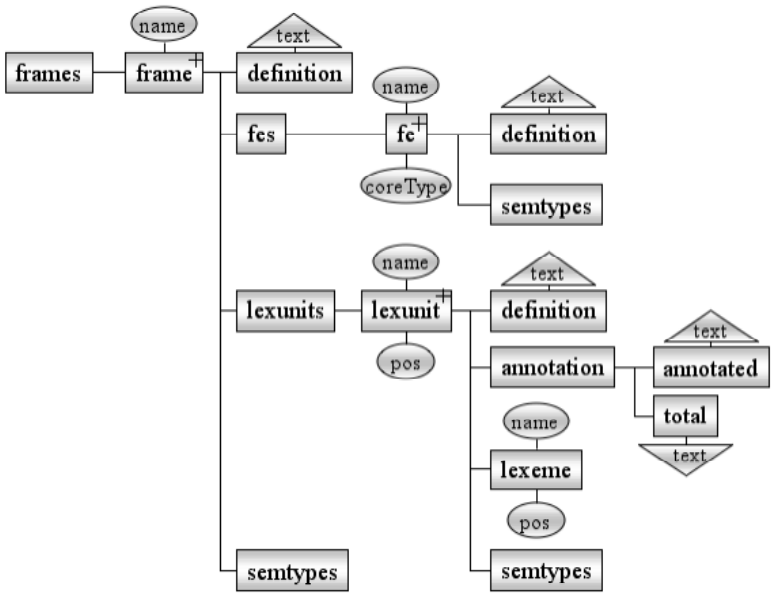

Figure 13: XML structure of the FrameNet frame description

```
<frame ID="269" name="Seeking">
  <definition>A Cognizer_agent attempts to find some Sought_entity by examining some Ground...</defi
  <fes>
    <fe ID="2327" name="Cognizer_agent" coreType="Core">
      <definition>The person who attempts to find the Sought_entity.</definition>
      <semtypes/>
    <fe ID="2329" name="Sought_entity" coreType="Core">
        <definition>This is the entity that the Ground may contain...</definition>
        <semtypes />
    </fe>
  </fes>
  <lexunits>
    <lexunit ID="4909" name="rummage.v" pos="V">
      <definition>COD: search unsystematically and untidily for something</definition>
      <annotation>
        <annotated>0</annotated>
        <total>78</total>
      </annotation>
      <lexeme ID="33763" name="rummage" pos="V"/>
      <semtypes />
    </lexunit>
  </lexunits>
</frame>
```

Figure 14: Parts from the definition of frame SEEKING (from file `frames.xml`).

References

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

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

[3] Charles Fillmore. Frame Semantics. *Linguistics in the Morning Calm*, 1982.

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

[5] Andreas Mengel and Wolfgang Lezius. An XML-based encoding format for syntactically annotated corpora. In *Proceedings of LREC-2000*, Athens, Greece, 2000.

Appendix: SALSA/TIGER XML Example

Figure 15 shows an example[7] XML representation of the frame-annotated sentence *Jetzt sucht man die flüchtigen Täter*. In the `<graph>` section, the words (terminals) and syntactic structure (nonterminals) are represented. The annotaded frame SEEKING with the two roles COGNIZER_AGENT and SOUGHT_ENTITY are displayed in the `<sem>` section pointing via `idref`s to the respective words and constituents in the `<graph>` section.

```
<s id="s2611">
  <graph root="s2611_VROOT">
    <terminals>
      <t id="s2611_1" word="Jetzt" pos="ADV" morph="--"/>
      <t id="s2611_2" word="sucht" pos="VVFIN" morph="--"/>
      <t id="s2611_3" word="man" pos="PIS" morph="--"/>
      <t id="s2611_4" word="die" pos="ART" morph="--"/>
      <t id="s2611_5" word="fluechtigen" pos="ADJA" morph="--"/>
      <t id="s2611_6" word="Taeter" pos="NN" morph="--"/>
      <t id="s2611_7" word="." pos="$." morph="--"/>
    </terminals>
    <nonterminals>
      <nt id="s2611_501" cat="S">
        <edge label="MO" idref="s2611_1"/>
        <edge label="HD" idref="s2611_2"/>
        <edge label="SB" idref="s2611_3"/>
        <edge label="OA" idref="s2611_500"/>
      </nt>
      <nt id="s2611_500" cat="NP">
        <edge label="NK" idref="s2611_4"/>
        <edge label="NK" idref="s2611_5"/>
        <edge label="NK" idref="s2611_6"/>
      </nt>
      <nt id="s2611_VROOT" cat="VROOT">
        <edge label="--" idref="s2611_501"/>
        <edge label="--" idref="s2611_7"/>
      </nt>
    </nonterminals>
  </graph>
  <sem>
    <frames>
      <frame name="Seeking" id="s2611_f1">
        <target>
          <fenode idref="s2611_2"/>
        </target>
      <fe name="Cognizer_agent" id="s2611_f1_e1">
        <fenode idref="s2611_3"/>
      </fe>
      <fe name="Sought_entity" id="s2611_f1_e2">
        <fenode idref="s2611_500"/>
      </fe>
      </frame>
    </frames>
  </sem>
</s>
```

Figure 15: SALSA/TIGER XML extract

[7] This representation does not contain empty elements, see example files for full structures.

Appendix: User License Conditions

IMPORTANT: THESE SOFTWARE END USER LICENSE CONDITIONS PROVIDE A LICENSE TO USE SALTO AND CONTAIN WARRANTY INFORMATION AND LIABILITY DISCLAIMERS. BY DOWNLOADING AND USING SALTO, YOU ARE AGREEING TO BECOME BOUND BY THESE CONDITIONS. IF YOU DO NOT AGREE TO BE BOUND BY THESE TERMS, THEN DO NOT INSTALL SALTO.

1. **Definitions**

   a) "SALTO" means a software program to annotate text corpora with FrameNet frames developed at Saarland University, Saarbrücken, Germany under the responsible direction and investigation of Prof. Dr. Manfred Pinkal and implemented by CLT Sprachtechnologie GmbH, Saarbrücken.

   b) "TIGER Search" means a software of the IMS, Stuttgart for the administration and exploitation of data collections of a restricted form of graph structures which is incorporated into SALTO.

2. **License Grants and Restrictions**

   a) The Licensee accepts the License agreements of TIGER Search and that his registration information may be reported to TIGER Search.

   b) The Licensee accepts the software and user license conditions for SALTO and subject to that shall be granted a royalty-free, non-exclusive, non-transferable Licence to use SALTO only for research and evaluation purposes.

   c) The Licensee may not distribute copies of SALTO.

3. **Warranty**

   a) The software is provided on an "as-is" basis and without any guarantee. Saarland University including its members, especially Prof. Dr. Pinkal, provides no technical support.

   b) The licensee may contact CLT Sprachtechnologie GmbH to negotiate possibilites for technical support.

4. **Liability Disclaimer**

   a) Neither Saarland University including its members, especially Prof. Dr. Pinkal, nor its suppliers shall be liable to the licensee or any third party for any damages arising out of the use of SALTO.

5. **Termination**

   a) The license shall be immediately terminated if the Licensee commits any material breach of this software end user license conditions for SALTO.

   b) On termination howsoever caused, the Licensee shall erase all copies of SALTO.

Index