SAM - an annotation editor for parallel texts

Markus Geilfuss, Jan-Torsten Milde

Department of Computer Science, Fulda University of Applied Sciences
business@microtide.de, milde@fh-fulda.de

Abstract

Parallel corpora are an important resource for quantitative and qualitative linguistic research. This paper describes the design and implementation of an interactive system allowing the user to annotate parallel texts: SAM, the Script Annotation Manager.

1. Introduction

Annotated parallel texts are an important resource for quantitative and qualitative linguistic research. Creating parallel corpora enables the generation of (bilingual) lexica, provides a basis for the extraction of data used for translation memories, makes is possible to describe the differences between text versions (e.g. multi level annotated corpora of texts by Goethe ((Seiler and Milde, 2004b) and Fontane (Seiler and Milde, 2004a))), or simply allows scientists to create texts in cooperation.
As many linguistic ressources are created as a collection of text files, setting up parallel corpora also provides interesting perspectices to corpus linguistics in general. Linguists are facing an number of problems, when trying to efficiently create corpora:

- *Corpus creation is a dynamic process*. The annotation structure and the interpretation of the annotation varies over time and depends on the linguistic community.
- *Extensability and interoperability*. Corpora are generally created and optimized to fulfill a single limited task. Reusing exsting corpora in different setting is currently hard to achieve.
- *Adaptation to linguistic work flows*. Scientists in different areas of linguistics tend to use differing approaches for setting up corpora. Even within a single linguistic area there is much debate on how to setting up a corpus ”correctly”.

By creating parallel corpora these problems could be solved, as the linguist is alway free to add more parallel text layers, adopting to the specific goals of the research, without destroying the existing corpus structure .
In this paper we describe the design and implementation of an interactive editor allowing the user to annotate parallel texts. SAM, the Script Annotation Manager, has been developed as part of a diploma theses by first author ((Geilfuss, 2004), (Geilfuss and Milde, 2005)).
The paper will outline the requirements to met by the parallel text editor, show how text annotation and text linking is supported, explain the implementation of SAM in more detail, and give some explaination on the underlying XML formats.

2. SAM - der Script Annotation Editor

The primary goal with the development of SAM was to support the annotation process of parallel text. At first the system should support the user with the actual annotation of a *single* text file, c.f. by providing means to select parts of the text, then label this selection with an apropriate categorie and optionally enter a description for this annotation. This is comparable to perform a colored text marking on sheet of printed text. In addition the system should fulfill the specific requirements related to the annotation of *parallel* texts. Espcially the linking between multiple texts is an important feature to be implemented by the system. These links should be mutli dimensional and bi-directional and the physical text representation should be seperated from the linking representation. The implementation of SAM therefore targeted a system fulfilling the following requirements:

- to annotate texts on the basis of a common vocabulary
- to link texts and parts of texts across document boundaries and to add a semantic description to these links
- to allow the interactive definition of a descriptives vocabulary and to import/export vocabularies for specific annotation tasks, thus enabling the re-use of a given vocabulary
- process texts in parallel

3. Annotation and linking

With SAM the user is able to annotate the texts in an intuitive way. Typed links between annotation can be set. The display of both texts will be sychronized and the annotated parts are visually highlighted (very much like using a text marker for printed texts). The annotation process is split up into two steps:

- first, the user has to define the *annotation vocabulary*, which
- then will be used during the *parallel* annotation of the texts. This step also includes the *typed* linking of the parallel texts.

The annotation is stored in an XML format, that can be processed efficiently using standard XML technologies. SAM is following a hybrid approach for storing the data. For the text an *inline* annotation is used, while the linking is done using *standoff* markup ((Barnard et al., 1995)). Mixing the two approaches overcomes the well known structural restrictions of XML and at the same time leads to a clean separation of linking structure and text structure. The

inline annotation is defined through the annotation vocabulary, while the typed links are created separately. For each typed link an arbitrary type description can be defined. The link types are then organised in an (hierarchical) structure.
SAM distinguishes between the *creation of a vocabulary* and the *process of annotating texts*. This separation allows the user to create specific vocabularies for specific annotation tasks. A vocabulary is therefor a reusable annotation model. Its structure has to be flexible in order to be applicable to a large set if annotation problems. In SAM the annotation vocabulary is defined by two parts:

1. *AnnotationTypes* define an annotation template and its content model

2. *AnnotationLink Type* define the semantic description of a link

The AnnotationType defines the common features of an annotation. This includes visual features like color or background color. The AnnotationType also defines the semantic features of an annotation, e.g. the conditions that have to be fulfilled in order to synchronize annotations. SAM organizes AnnotationsTypes in a hierarchical manner. The graphical user interface is shown in figure 1.

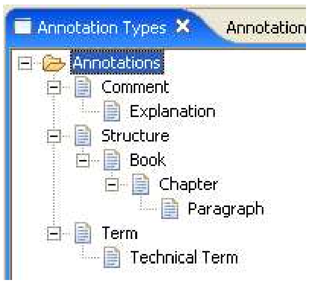


Figure 1: The AnnotationType View provides display and editing for all used AnnotationTypes of a given annotation vocabulary. AnnotationType can be created, edited and deleted from the vocabulary.

AnnotatonLink Types define the annotation of a link (that is link metadata). This can be used to describe the links in some detail and can later be used the distinguish and process the links. The graphical user interface is shown in figure 2.
The connection between two annotations A and B is a directed labeled link. Annotations can be definied in multiple documents. As such links can span across document borders. If a link is set, the user selects the annotation link type. The definition of the types is not restricted in any way. Arbitrary strings can be used. SAM does not interprete the annotation link types (e.g. $A = B$ could mark equal links, if an annotation A extends annotation B this could be denoted by $A > B$).
Currently we do not use X-Link/X-Pointer ((Wilde and Lowe, 2002)), which would be a natural choice for a defining syntax. In order to implement the proposed linking model with X-Link, a locator definition for both annotations must be definied. These locator definition would need

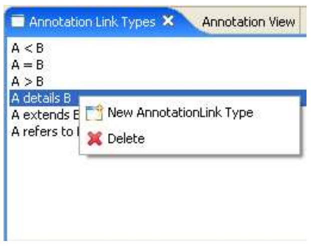
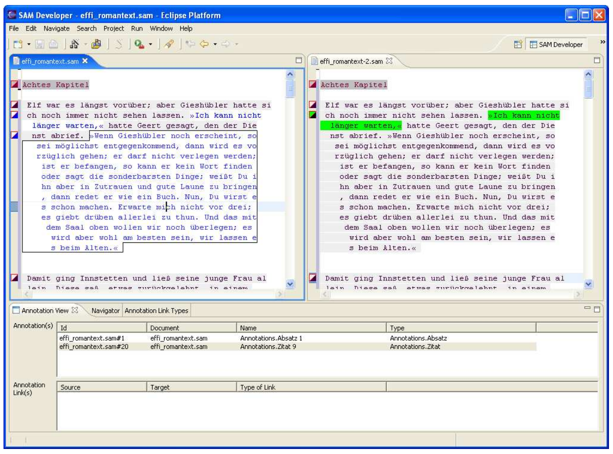


Figure 2: The annotation link type view displays information about the currently selected link type. The user is able to edit the information. New links types can be created, existing link type can be deleted. The graphical user interface corresponds to the annotation type view.

to carry a uniqe identifier and a X-Link/X-Pointer conformant link to the definition of the related annotation. Finally the link itself had to be definied. During the design phase of SAM no adaquate implementation of X-Link/X-Pointer existed and implementing a standard conformant engine was out of scope of the project.

4. Implementation

SAM has been implemented as a plugin for the open source platform Eclipse (see (`www.eclipse.org`), (Shavor et al., 2004)). The *Eclipse Workbench* provides a robust extensible software infrastructure, that makes it possible to efficiently design and implement linguistic applications. Eclipse offers excellent components (e.g. configurable editors, tree views) which are perfectly matching the structures of the scientific data under investigation. Existing tools are easily integrated into an Eclipse application. Furthermore, Eclipse-based tools are well portable to a number of operating systems.



```
Achtes Kapitel

Elf war es längst vorüber; aber Gieshübler hatte si
ch noch immer nicht sehen lassen. »Ich kann nicht
 länger warten,« hatte Geert gesagt, den der Die
 nst abrief. »Wenn Gieshübler noch erscheint, so
  sei möglichst entgegenkommend, dann wird es vo
  rzüglich gehen; er darf nicht verlegen werden;
   ist er befangen, so kann er kein Wort finden
   oder sagt die sonderbarsten Dinge; weißt Du i
   hn aber in Zutrauen und gute Laune zu bringen
   , dann redet er wie ein Buch. Nun, Du wirst e
   s schon machen. Erwarte mich nicht vor drei;
   es giebt drüben allerlei zu thun. Und das mit
    dem Saal oben wollen wir noch überlegen; es
     wird aber wohl am besten sein, wir lassen e
     s beim Alten.«

Damit ging Innstetten und ließ seine junge Frau al
```

Id	Document	Name	Type
effi_romantext.sam#1	effi_romantext.sam	Annotations.Absatz 1	Annotations.Absatz
effi_romantext.sam#20	effi_romantext.sam	Annotations.Zitat 9	Annotations.Zitat

Source	Target	Type of Link

Figure 3: The SAM perspective. Two texts are presented in parallel. Additional information about the annotations is given in lower views.

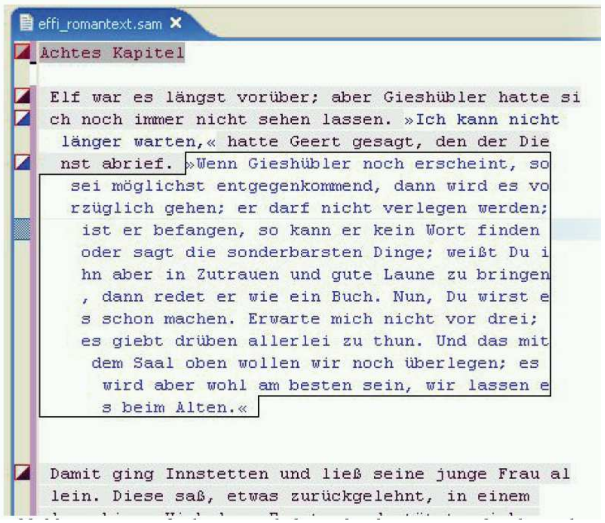
The SAM graphical user interface (see figure 3) is targeting the efficient annotation of parallel texts. The user is able to open and process two or more texts at the same time. Eclipse based text editors are responsible for the visual presentation of the textual data. The interactive components,

e.g. context sensitive popup menues, have been integrated into these editors. Most of the functions provided by the system are accessible from inside the text editing area, thus enlarging the space available for text display. With this approach, experienced users are able to work much faster, at least in comparison to a GUI, where the interactive components are provided by top level menue bars. In addition, this integration allows to limit the number of external views. A small number of views are provided, mainly to support the linking process. The inital creation of a parallel text corpus is automatized by wizards (c.f. a series of fill out forms), guiding the user through the configuration process.
All views and editors are integrated into the Eclipse Wokbench architecture. This means, that the components implement the Model-View-Controller design pattern (MVC). The data to be processed, in this case the texts to be annotated, are represented in an internal model. Views and editor are accessing the model via a controller class. Changes within the model are propageted by the Eclipse internal event mechanism. Additional components need to register themselves as event listeners and will then be informed about these changes. By adopting to this architecture principal, SAM components can be used outside the current configuration and new components can be added to the configuration.
Within the text editor, annotations are highlighted and surrounded by a border. Overlapping annotations are indicated by markers on the left hand side of the editor (see figure 4). The text editors are synchronized with the annotation views. As soon as the user moves the caret into an annotated region of the text, additional information on this annotation is displayed in the annotation view. This is especially important in the case of overlapping annotations. Here annotation view allows to select the relevant annotation. At the same time all linking information of the current annotation is displayed. The user is able to edit and delete this information. The source and target region of the link is displayed. This makes it relatively easy to keep track of the linking structure of complex parallel texts.
SAM is providing two ways for keeping the texts in synchronisation: *by selection* or *by scrolling*. The by selection mode adjusts the display as soon as a typed link has been selected, while the by scrolling mode will keep both texts synchronized constantly[1].

5. Data format of SAM

All data in SAM ist stored in XML files. The annotations are kept seperate from the text files. The relation between text and annotation is established by storing offset and length of the annotation. These must be updated, when the user is modifying the original texts. SAM performs these updates in a background process. The annotation are formally defined by the following DTD:

```
<!ELEMENT sam (annotations*, text)>
<!ELEMENT annotations (annotation*)>
<!ATTLIST annotations
    author CDATA #REQUIRED
    vocabulary CDATA #REQUIRED
 >
<!ELEMENT annotation (link*)>
<!ATTLIST annotation
    type CDATA #REQUIRED
    length CDATA #REQUIRED
    offset CDATA #REQUIRED
    name CDATA #REQUIRED
    id CDATA #REQUIRED
>
<!ELEMENT link EMPTY>
<!ATTLIST link
    href CDATA #REQUIRED
    linkId CDATA #REQUIRED
    type (target | source) #REQUIRED
>
<!ELEMENT text (#PCDATA)>
```



```
effi_romantext.sam
Achtes Kapitel

 Elf war es längst vorüber; aber Gieshübler hatte si
 ch noch immer nicht sehen lassen. »Ich kann nicht
  länger warten,« hatte Geert gesagt, den der Die
  nst abrief. »Wenn Gieshübler noch erscheint, so
   sei möglichst entgegenkommend, dann wird es vo
   rzüglich gehen; er darf nicht verlegen werden;
    ist er befangen, so kann er kein Wort finden
    oder sagt die sonderbarsten Dinge; weißt Du i
    hn aber in Zutrauen und gute Laune zu bringen
    , dann redet er wie ein Buch. Nun, Du wirst e
    s schon machen. Erwarte mich nicht vor drei;
    es giebt drüben allerlei zu thun. Und das mit
     dem Saal oben wollen wir noch überlegen; es
      wird aber wohl am besten sein, wir lassen e
      s beim Alten.«

 Damit ging Innstetten und ließ seine junge Frau al
 lein. Diese saß, etwas zurückgelehnt, in einem
```

Figure 4: Annotations are highlighted and surrounded by a border. Overlapping annotations are indicated by markers on the left hand side of the editor.

The annotation vocabulary is stored seperately from the annotations. This enables the user to extend and modify the *annotation types* and *linking types*, without destroying the references to text. Linking types are organized as linear list. Each type has a name and a uniqe identifier. The annotation types are organized as a hierarchical tree structure. Nodes store information about different aspects of the annotation. By establishing a tree structure, attributes in subtees could be intepreted in the context of its ancestor nodes. The following DTD fragment defines this structure more formally:

```
<!ELEMENT vocabulary
    (annotationTypes, annotationLinkTypes)>

<!ELEMENT annotationTypes (annotationType)>

<!ELEMENT annotationType (annotationType*)>
<!ATTLIST annotationType
    gotoRefScroll (false | true) #REQUIRED
    type CDATA #REQUIRED
    createSubdoc (false | true) #REQUIRED
    enabled (false | true) #IMPLIED
```

[1]More information on SAM can be found under `http://medien.informatik.fh-fulda.de`.

```
    target (false | true) #REQUIRED
    gotoRef (false | true) #REQUIRED
    source (false | true) #REQUIRED
    textColor CDATA #REQUIRED
    bgColor CDATA #REQUIRED
>

<!ELEMENT annotationLinkTypes
    (annotationLinkType*)>

<!ELEMENT annotationLinkType EMPTY>

<!ATTLIST annotationLinkType
    name CDATA #REQUIRED
    id CDATA #REQUIRED
>
```

6. Conclusions

The annotation editor SAM supports the efficient annotation of parallel texts. The chosen hybrid approach for storing the data overcomes the structural restrictions imposed by XML. The implementation is based on the Eclipse framework, that provides a robust and extensible system architecture. The graphical user interface is designed to optimally support the work flow during the annotation phase of corpus creation.

7. References

David Barnard, Lou Burnard, Lynne A. Price Jean-Pierre Gaspart, Michael Sperberg-McQueen, and Giovanni Battista Varile. 1995. Hierarchical encoding of text: Technical problems and sgml solutions. *Computers and the Humanities*, 29(3):211–231.

Markus Geilfuss and Jan-Torsten Milde. 2005. Sam - ein annotationseditor für parallele texte. In Rainer Eckstein und Robert Tolksdorf, editor, *Tagungsband XML Tage 2005, Berlin*.

Markus Geilfuss. 2004. *Planung, Entwicklung und Umsetzung eines Annotationseditors für parallele Texte in Eclipse*. FH Fulda, Diplomarbeit im Fachbereich Angewandte Informatik.

Bernd Seiler and Jan-Torsten Milde. 2004a. *Fontanes Effi Briest. Bilder - Texte - Töne. Ein Literaturkommentar auf CD-ROM*. CC Buchner.

Bernd Seiler and Jan-Torsten Milde. 2004b. *Goethes Werther. Bilder - Texte - Töne. Ein Literaturkommentar auf CD-ROM*. CC Buchner.

Sherry Shavor, Scott Fairbrother, Jim D'Anjou, and Dan Kehn. 2004. *Eclipse*. Addison-Wesley Professional.

Erik Wilde and David Lowe. 2002. *XPath, XLink, XPointer, and XML*. Addison-Wesley Professional.