SWAN: an easy-to-use web-based annotation system

Timo Gühring[1,2] **Nicklas Linz**[2,3] **Rafael Theis**[2] **Annemarie Friedrich**[1]
[1]Department of Computational Linguistics, Saarland University
[2]Department of Computer Science, Saarland University
[3]German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany
timo.guehring@googlemail.com nicklas.linz@dfki.de
s9rathei@stud.uni-saarland.de afried@coli.uni-saarland.de

Abstract

We present the Saar Web ANnotation system (SWAN), a lean web-based annotation system, which is optimized for annotator and project management usability.[1] SWAN is well-suited for various discourse annotation tasks, as arbitrarily large documents can be annotated seamlessly due to dynamic loading and rendering. A graph-based visualization box supports the user by providing both an overview of the existing annotations and a navigation option through the text. The admin view includes the web-based configuration of projects, annotation schemes and users. SWAN is based on JEE technology and compatible with several web browsers.

1 Overview

Manual annotation of text documents is the backbone of natural language processing (NLP) research. Among others, the annotation of linguistic phenomena is necessary for training and evaluating NLP systems. In recent years, NLP research increasingly addresses linguistic phenomena beyond the sentence level, i.e., discourse information (Webber et al., 2012). Text corpora have been annotated with discourse relations (Prasad et al., 2008; Carlson et al., 2002), temporal relations (Day et al., 2003) and coreference (Hovy et al., 2006). Discourse annotation tasks are characteristically different from sentence-level tasks in three major ways: (1) The annotator needs to have an overview of the entire document and potentially create links between spans that are far apart from each other. (2) Spans for annotation can be large, i.e., they may consist of clauses, sentences or even paragraphs. (3) The location of paragraph breaks may be relevant. Hence, unlike other existing web-based annotation systems, SWAN displays the text documents in their original formatting by default.

As the annotators are the main users of the system, we focus on optimizing usability of the annotators' view of the system. One key factor driving the development of our system was the need for an intuitive and usable interface for discourse annotation tasks such as labeling texts with event structures and temporal information, or marking paragraphs with their discourse mode. The latter comprise distinctions like *narrative*, *information*, *report*, *description* or *argumentative* (Smith, 2003). For these tasks, we have developed an editor that is responsive even for large documents, allows for easy and quick selection of arbitrary spans, as well as creating links between annotations. For example, in annotation tasks with a relatively small set of types and labels, SWAN can be configured to display all selectable labels once a span annotation or a link annotation has been created. This is especially useful for annotators who are getting familiar with a task, and allows for fast selection especially in cases where the tag set is limited.

In some of our use cases, large spans corresponding to paragraphs need to be selected. If the annotator decides that the span selection should be slightly different, deleting the annotation and renewing is time-consuming, especially if the annotation is already linked to others. Easily changing the extent of a span annotation is a novel feature developed specifically for these types of discourse annotation.

In addition, SWAN comes with a powerful admin and project management view, which provides for defining annotation schemes, managing users and tracking the progress of annotation projects.

SWAN has the following novel features:

- **Dynamic loading and rendering of the document.** This allows for arbitrarily large documents without having to split the document

[1]Web demo and code available at https://swan.coli.uni-saarland.de and https://github.com/annefried/swan.

into multiple pages, which is essential for our discourse-level annotation tasks.

- **Graph visualization.** The graph structure of the annotations is shown next to the text. When selecting an annotation either in the text or in the graph, annotations are highlighted correspondingly; the graph provides an overview as well as an option for easy navigation through the document.

- **Usability optimized for discourse annotation.** Various new features target the user experience for making changes that arise in discourse annotation tasks, such as easily changing the extent of a span for an annotation. In addition, we aim for simplicity and intuitiveness, keeping the interface and the concepts underlying the annotation scheme as simple as possible for annotators.

SWAN is a web application based on JEE technology, runs on the GlassFish server and uses a PostgreSQL database. The front-end is realized using HTML and JavaScript and is compatible with major recent web browsers. The source code, a pre-packaged WAR ready for deployment, as well as installation and set-up instructions are publicly available via GitHub. SWAN v2.0 is a stable release and the system is already in use in various projects at our department. Issue tracking and support is provided via GitHub.

2 Related work

The most similar system to ours is WebAnno (Yimam et al., 2013), a web-based and configurable annotation tool. It is the state-of-the-art annotation system for many natural language annotation tasks such as part-of-speech tagging, syntactic dependency trees or coreference. WebAnno splits longer documents into multiple pages (configurable by the annotator) and loads the next part of the document if explicitly requested or if the end of a page has been reached. In SWAN, we overcome the efficiency problem underlying this design decision by dynamically reloading content and only rendering the visible parts of the document. WebAnno uses the front-end of BRAT (Stenetorp et al., 2012), the first web-based open source annotation tool, extending its functionality mainly regarding configuration options and supported file formats. In BRAT, links are always displayed on top of the text line, which makes sense for within-sentence annotation tasks. However, if links cross sentence boundaries, as is frequently the case in our discourse annotation tasks, BRAT displays the links as ending at the right side of a line and starting again at the left side of a new line, which is somewhat counter-intuitive. In SWAN, we solve this problem by directly connecting nodes in the text, but graying out non-selected links to improve readability. Related to our work are also GrapAT (Sonntag and Stede, 2014) and rstWeb (Zeldes, 2016), which are both web-based systems focusing on annotating and displaying graph structures on top of text. Some larger-scale discourse annotation projects (Prasad et al., 2008; Carlson et al., 2002; Cassidy et al., 2014) have developed their own annotation-scheme specific tools, which are implemented as locally-running applications.[2,3,4]

3 Annotation schemes

Annotation schemes in SWAN follow a simple concept, in accordance with our intuition that the full complexity of type systems should be represented in the logic of software processing the data, but not necessarily during annotation. Annotators, who often do not have a formal background, thus can focus on a particular task without having to worry about the big picture. Annotation schemes can be configured and modified by project managers using the scheme builder. A full example of an annotation scheme is given in Figure 1.

Span annotations in SWAN consist of spans (any number of contiguous tokens) and are assigned a **span type**, e.g., *NounPhrase*, *Clause*, or *Passage*. The first step after creating an annotation by selecting a span in the text is to choose its span type. **Label sets** are defined as sets of **labels**, and apply to particular span types. They are displayed as soon as a span annotation has been created and its type has been selected. A possible label set for the span type *Passage* would be *DiscourseMode* with labels including *narrative*, *report*, *information* or *description* (Smith, 2003). Another label set applying to the type *Clause* could be *EventType* including the labels *state*, *achievement*, *activity* and *accomplishment* (Vendler, 1957). Label sets can be configured with regard to whether they are **exclusive**, i.e., whether annotators can select only one

[2] www.seas.upenn.edu/~pdtb/tools.shtml
[3] www.isi.edu/licensed-sw/RSTTool
[4] www.usna.edu/Users/cs/nchamber/caevo

label out of a set or assign multiple labels from one set to one span annotation.

Link annotations are edges from one span annotation to another. **Link types** define the type of a link annotation. For each link type, the span types of the start and end annotations need to be defined. For *TemporalRelation*, the type of both the start and end annotations could be *Event*, but the types of the start and end annotations can in general be defined separately and may differ. A link type is also associated with a set of **labels** that can be assigned to a link of this type. Link labels for our *TemporalRelation* link type would be the senses of temporal relations such as *before*, *overlap*, *includes* or *simultaneous* (Day et al., 2003).

```
<?xml version="1.0"?>
<root>
  <name>ExampleScheme</name>
  <spanTypes>
    <spanType>Event</spanType>
  </spanTypes>
  <labelSets>
    <labelSet>
      <name>EventType</name>
      <exclusive>false</exclusive>
      <appliesToSpanTypes>
        <spanType>Event</spanType>
      </appliesToSpanTypes>
      <labels>
        <label>State</label>
        <label>Event</label>
        <label>Generic Sentence</label>
      </labels>
    </labelSet>
    <labelSet> ... </labelSet>
  </labelSets>
  <linkTypes>
    <linkType>
      <name>Temporal relation</name>
    <startSpanType>Event</startSpanType>
      <endSpanType>Event</endSpanType>
      <linkLabels>
        <label>before</label>
        <label>after</label>
        <label>overlap</label>
      </linkLabels>
    </linkType>
    <linkType> ... </linkType>
  </linkTypes>
</root>
```

Figure 1: Example SWAN annotation scheme. *Please check documentation for details / updates.*

4 Functionality

4.1 Roles and projects

SWAN defines two primary roles: **annotator** and **project manager**. A project consists of an annotation scheme, a set of text documents and a set of annotators assigned to it. Project managers can create user accounts for annotators, annotation schemes and projects, while annotators can add annotations to the text documents of projects that they were assigned to. Project managers can see, edit, delete or export only the projects that they have created or that they have been assigned to, but all annotation schemes existing in the database are available to all project managers. Consistency of the underlying database is ensured as the scheme used in a particular project is immutable in the current release. For convenience, however, schemes can be copied and then modified when creating new projects. In addition, the system allows for **admin** users, typically the persons who administrate the installation. Admins have access to all projects, and only they can create or delete project manager accounts, while project managers can manage annotator accounts. In addition, project managers can view the annotations of their annotators using a "non-editable" version of the annotator's view.

4.2 File formats

In order to be able to display documents in their original formatting, SWAN provides for the upload of plain text files, and bases internal representations of annotations on character offsets pointing to spans in the original plain text document. If project managers want to pre-define annotations along with their types as the **targets** for an annotation task, they can upload this information in a separate file along with the plain text documents. Input and export formats for annotations use JSON or XML (for an example annotation scheme, see Figure 1). In addition, annotated documents can be downloaded directly in the UIMA XMI format (Ferrucci and Lally, 2004). Export is tied to projects, i.e., one zip file can be downloaded per project containing all annotations of all annotators.

4.3 Web interface components

The **project explorer** allows project managers to edit projects, i.e., assign annotators to a project or upload text documents optionally along with some pre-defined span annotations. Annotators see the projects that they have been assigned to and which documents they have or have not yet completed. The **scheme builder** allows project managers to create annotation schemes. Once saved and entered in the database, schemes can be modified by creating a copy and editing this copy. Projects need to be assigned an existing scheme at the time of their

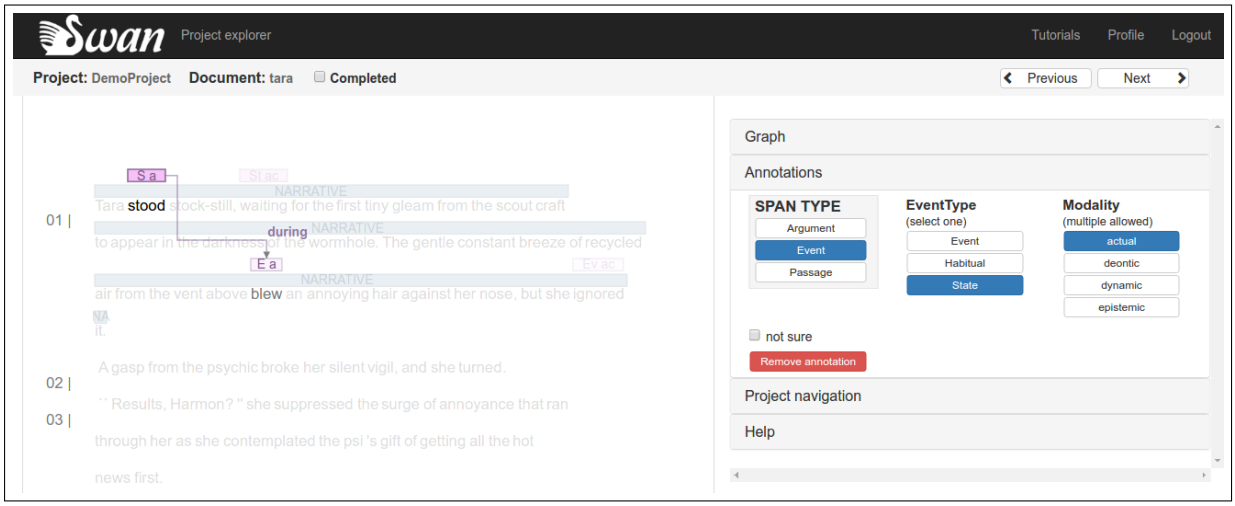

Figure 2: SWAN editor showing type / labels for highlighted annotation ("S a" = "State, actual").

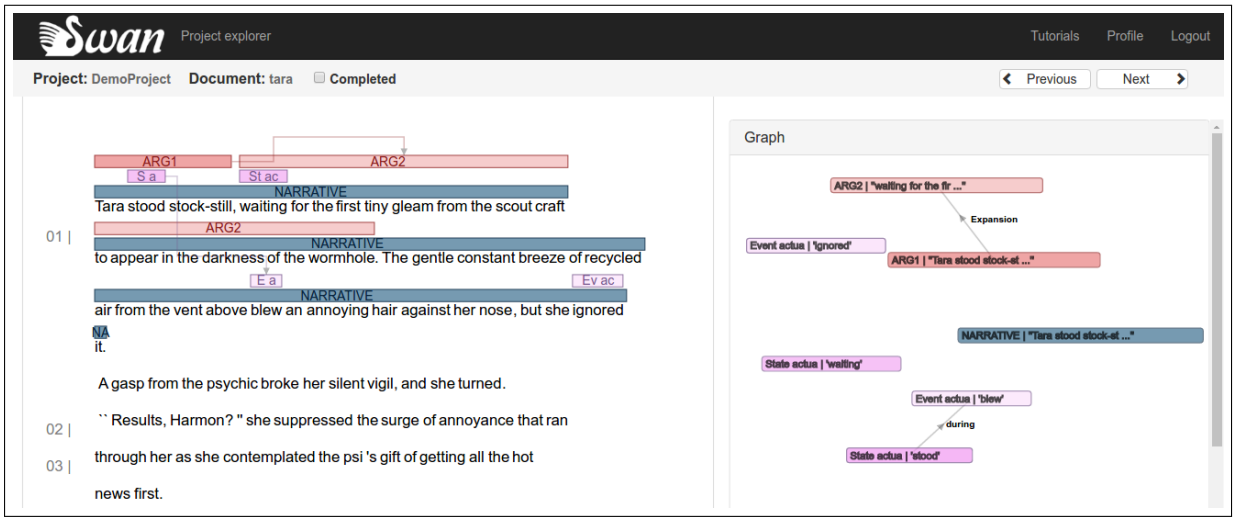

Figure 3: SWAN editor, no annotation selected, graph visualization.

creation. These restrictions ensure that the underlying database is always in a consistent state. In the near future, we will also explore options to add support for modification of schemes that are already in use. The **editor** displays the text document, the labels available for selection and the graph visualization box. Spurious spaces are removed in the editor in order to save space, but they are kept in the background, i.e., the annotation offsets relate to the original uploaded plain text file. Annotations are created by selecting tokens in the text using the mouse or a range of predefined keyboard shortcuts. This includes the quick shortcut-based selection of large spans based on selecting or de-selecting entire lines.

Links are created by selecting the start annotation and dragging the mouse to the end annotation. When removing a span annotation, all links starting or ending at this node are also automatically removed, as links cannot exist without a start and end annotation. In addition, existing annotations can be extended or made smaller token-wise to the left or right using simple keyboard shortcuts. Once an annotation has been created, the available types and respective labels are displayed.

Graph visualization. Links between annotations are visualized only selectively on top of the text by displaying links starting or ending at the selected annotation, and graying out the remaining links (see Figure 2). An optional visualization box (see Figure 3) shows the graph structure of the document. When clicking on a node, the document text view scrolls to the position of the annotation, the corresponding annotation is highlighted in the document, and the local graph structure is also highlighted on top of the text. Thus, in ad-

dition to allowing an overview of the document's discourse structure in terms of the respective annotation scheme, the graph visualization box provides an additional possibility for users to navigate through the annotations they have created for a document. Showing this graph structure directly on top of the text is often not possible as a sensible arrangement of nodes in the graph does not always follow textual order, e.g, for annotating temporal structure. We are also currently working on additional layout options for the graph visualization box such as tree structures and a layout option that is appropriate for temporal relation annotation, i.e., where the selected link labels decide on the node's arrangement in the graph.

5 Software architecture

SWAN is a Java Enterprise Edition (JEE) web-based application (see Figure 4). Being distributed as a Web application ARchive (WAR), it is easily deployable. Back-end and front-end both use lightweight RESTful web services for communication, sending data in a compact JSON format.

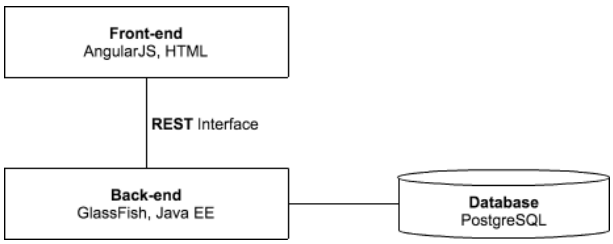

Figure 4: The system architecture of SWAN.

Back-end. The back-end of SWAN runs on GlassFish,[5] which is an open-source application server that supports JEE, and ships with a minimum of configuration effort. All user, text and annotation data is stored in the open-source PostgreSQL database,[6] enabling simple back-up solutions by regularly creating backups of the database. Texts are tokenized using the Stanford PTB Tokenizer (Manning et al., 2014). We currently support English, German, Spanish and French as well as custom white-space-based tokenization. Tokenization options for more languages will follow in future releases.

Front-end. SWAN's front-end, running in a web browser,[7] is based on the AngularJS JavaScript framework[8] and standard HTML. For data visualization, the D3 framework,[9] an industry standard for the visualization of large amounts of data, is used. It is applied to render the text, annotation boxes, graph and timeline as Scalable Vector Graphics (SVG). Using this framework, approximating which parts of the document are visible and rendering only those enables SWAN to display long texts while offering reasonable performance.

6 Discussion and outlook

SWAN is a web-based annotation system focusing on usability for annotation tasks that require the annotator to freely navigate through the entire text document. While being optimized for our annotation projects related to discourse and event structure, SWAN is generally a good option for annotation projects requiring an easy-to-use interface. SWAN does not (yet) provide an adjudication view, as in our own research projects, in order to ensure replicability, we create gold standard data from voting between many annotators rather than simply modeling an adjudicator's view of the data. The near-future development efforts in SWAN will concentrate on implementing additional options for visualizing the document's structure in the graph visualization box, including tree structures and other arrangements useful for quick navigation through the document. Future releases will also include options for monitoring inter-annotator agreement and possibly additional input formats.

Acknowledgments

We thank the anonymous reviewers, Andrea Horbach and Manfred Pinkal for their helpful comments related to this work, and Stefan Grünewald, Julia Dembowski and Janna Herrmann for contributing to SWAN's implementation. We also thank our annotators and project managers Simon Ostermann, Hannah Seitz, Damyana Gateva, Melissa Peate Sørensen, Christine Bocionek and Fernando Ardente for their for their support and useful ideas. This research was supported in part by the Cluster of Excellence "Multimodal Computing and Interaction" of the German Excellence Initiative (DFG).

[5] https://glassfish.java.net/
[6] http://www.postgresql.org/
[7] We support Mozilla Firefox, Google Chrome and Safari.
[8] https://angularjs.org
[9] https://d3js.org

References

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2002. RST Discourse Treebank LDC2002T07. Web Download. Philadelphia: Linguistic Data Consortium.

Taylor Cassidy, Bill McDowell, Nathanael Chambers, and Steven Bethard. 2014. An annotation framework for dense event ordering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 501–506, Baltimore, MD, USA.

David Day, Lisa Ferro, Robert Gaizauskas, Patrick Hanks, Marcia Lazo, James Pustejovsky, Roser Saurı, Andrew See, Andrea Setzer, and Beth Sundheim. 2003. The TimeBank corpus. In *Corpus Linguistics*.

David Ferrucci and Adam Lally. 2004. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, New York City, NY, USA.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, MD, USA.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse TreeBank 2.0. In *In Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC)*, Marrakech, Morocco.

Carlota S Smith. 2003. *Modes of discourse: The local structure of texts*, volume 103. Cambridge University Press.

Jonathan Sonntag and Manfred Stede. 2014. Grapat: a tool for graph annotations. In *In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*, pages 4147–4151, Reykjavik, Iceland.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. BRAT: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France.

Zeno Vendler. 1957. Verbs and times. *The philosophical review*, 66(2):143–160.

Bonnie Webber, Markus Egg, and Valia Kordoni. 2012. Discourse structure and language technology. *Natural Language Engineering*, 18(04):437–490.

Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. 2013. WebAnno: A Flexible, Web-based and Visually Supported System for Distributed Annotations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 1–6, Sofia, Bulgaria, August. Association for Computational Linguistics.

Amir Zeldes. 2016. rstWeb - A Browser-based Annotation Interface for Rhetorical Structure Theory and Discourse Relations. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 1–5, San Diego, CA, USA.