SANTO: A Web-based Annotation Tool for Ontology-driven Slot Filling

Matthias Hartung\textsuperscript{1,}*, Hendrik ter Horst\textsuperscript{1,}*, Frank Grimm\textsuperscript{1}, Tim Diekmann\textsuperscript{1}, Roman Klinger\textsuperscript{1,2} and Philipp Cimiano\textsuperscript{1}

\textsuperscript{1}CITEC, Bielefeld University
\textsuperscript{2}Institut für Maschinelle Sprachverarbeitung, University of Stuttgart
\{mhartung,hterhors,fgrimm,tdiekmann,cimiano\}@techfak.uni-bielefeld.de
roman.klinger@ims.uni-stuttgart.de

Abstract

Supervised machine learning algorithms require training data whose generation for complex relation extraction tasks tends to be difficult. Being optimized for relation extraction at sentence level, many annotation tools lack in facilitating the annotation of relational structures that are widely spread across the text. This leads to non-intuitive and cumbersome visualizations, making the annotation process unnecessarily time-consuming. We propose SANTO, an easy-to-use, domain-adaptive annotation tool specialized for complex slot filling tasks which may involve problems of cardinality and referential grounding. The web-based architecture enables fast and clearly structured annotation for multiple users in parallel. Relational structures are formulated as templates following the conceptualization of an underlying ontology. Further, import and export procedures of standard formats enable interoperability with external sources and tools.

1 Introduction

In most scientific and technical domains, the main medium for knowledge communication is unstructured text. Growing efforts are spent into literature-based knowledge discovery (Henry and McInnes, 2017) using information extraction or machine reading approaches for knowledge base population. The goal is to automatically transform the available domain knowledge into structured formats that can be leveraged for downstream analytical tasks.

Recent approaches reduced information extraction problems to binary relation extraction tasks at sentence level (Adel et al., 2016; Zhang et al., 2017, \textit{i. a.}). In such cases, the annotation procedure is comparably straight-forward to formulate. However, a substantial subset of information extraction tasks can be described as typed \emph{n}-ary relation extraction or slot filling, in which pre-defined sets of typed slots (templates) need to be assigned from information that may be widely spread across a text (Freitag, 2000). As such templates can contain many slots which may be recursively nested, an appropriate visualization is mandatory during annotation to handle the complexity of the task.

Relevant use cases for template-based information extraction exist in several fields of application: In the biomedical domain, there is a large body of work on database population from text in order to support translational medicine (Zhu et al., 2013; ter Horst et al., 2018, \textit{i. a.}). In digital humanities, there is a vital interest in detecting descriptions of historical events or artifacts from cultural heritage (Ruotsalo et al., 2009; Segers et al., 2011). In the context of manufacturing or retail, structured product descriptions are extracted from web pages or customer reviews (Bing et al., 2016; Petrovski and Bizer, 2017) to enable product comparisons.

In this work, we present SANTO, a lightweight, easy-to-use, domain-adaptive, web-based annotation tool specialized for complex slot filling tasks. SANTO is designed to address particular user needs that are recurrent in such scenarios. It enables data annotation and export into a machine-readable format based on the following features:

- Being based on information models such as ontologies for specifying the template scheme, it can be flexibly adapted to different domains and use cases.
- It enables annotations both at the textual and the template level; it is designed to support fast and seamless instantiation of new templates and their population from textual annotations, particularly in cases where slot fillers
pertaining to one template are widely distributed across the text.

- As common in slot filling, the tool builds on top of the fact that extraction models do not need to find every mention of an entity or relation in the text. Instead, the focus is on enabling the user to instantiate the correct cardinality of templates and ensuring referential uniqueness of slot fillers or templates.
- Annotations can be flexibly imported from different external sources. The RDF export of annotated relations within templates is conform to the underlying information model.
- It enables easy curation of annotations (e.g., from multiple imports).
- It is implemented as a web service in order to support remote annotation workflows for multiple users in parallel.

Availability and License. A demo installation is available at http://psink.techfak.uni-bielefeld.de/santo/. The source code of the application is publicly available under the Apache 2.0 License at https://github.com/ag-sc/SANTO.

2 Related Work

Most annotation frameworks for text focus on the sentence level. Examples include syntactic parsing (Burchardt et al., 2006) or semantic role labeling (Kakkonen, 2006; Yimam et al., 2013). Other tools focus on segmentation annotation tasks, for instance Callisto (Day et al., 2004), WordFreak (Morton and LaCivita, 2003), MMax2 (Müller and Strube, 2006), or GATE Teamware (Bontcheva et al., 2013) (though the latter also supports more complex schemata).

Brat (Stenetorp et al., 2012), WebAnno (Yimam et al., 2013), eHost (South et al., 2012) and CAT (Lenzi et al., 2012) support approaches for relational annotations. These tools are easy to use and highly flexible regarding the specification of annotation schemes. Projects are easy to manage due to administration interfaces and remote annotation is supported. However, these approaches share the limitation that all relational structures need to be anchored at the textual surface. Thus, annotating complex templates as in Figure 1 becomes tedious and visually cumbersome, especially in cases of complex nestings within or across templates, or when fillers are widely dispersed across multiple sentences in a text.

We propose a tool to frame complex relational annotation problems as slot filling tasks. To our knowledge, the only existing tool for this purpose is the Protégé plugin Knowtator (Ogren, 2006), which is, however, not web-based, comparably difficult to use with multiple annotators, and no longer actively supported since 2009. Thus, our main contribution is an annotation tool which combines the advantages of (i) enabling complex relational slot filling with distant fillers, and (ii) ease of use in web-based environments in order to facilitate remote collaboration.

SANTO is ontology-based, i.e., entity and relation types are derived from an underlying ontology. The same idea is prominent in several annotation tools within the Semantic Web community (cf. Oliveira and Rocha, 2013). Contrary to SANTO, these tools support annotations only at the level of individual entities without capturing relational structures as given in template-based slot filling.

3 Use Case: Information Extraction for Database Population

SANTO is designed to create machine-readable annotation data. As a motivating example, our presentation is guided by the use case of generating annotated training data for an ontology-based information extraction system that supports database population in the PSINK project1 (ter Horst et al., 2018). In this context, our goal is to annotate scientific publications reporting on the outcomes of pre-clinical trials in the spinal cord injury domain. The annotation schema complies to definitions of the Spinal Cord Injury Ontology (SCIO; Brazda et al., 2017). SCIO provides the concept RESULT as a top-level class which subsumes all classes and properties to represent the key parameters of an outcome in a study. Several properties are recursively sub-structured (e.g., EXPERIMENTAL-

Figure 1: Example template following a schema derived from the Spinal Cord Injury Ontology.

http://psink.de

\begin{tabular}{|c|c|}
\hline
\textbf{ORGANISM MODEL} & \textit{six-week-old} \\
\hline
\textbf{SPECIES} & SpragueDawleyRat \\
\hline
\textbf{GENDER} & Female \\
\hline
\textbf{AGE CATEGORY} & Adult \\
\hline
\textbf{WEIGHT} & “192–268g” \\
\hline
\end{tabular}

1http://psink.de
Figure 2: Entity Annotation: Relevant text mentions spanning one or multiple tokens are highlighted in green; entity type annotations are displayed on top of them. A new WEIGHT annotation is being added using the filtered drop-down menu.

Figure 3: Curation: Annotations of several annotators are displayed colorized. Annotations on the same token are on top of each other. Correct or spurious annotations can be accepted or rejected, respectively. Accepted annotations keep the color code, but are marked with dotted boundaries.

GROUPS, INVESTIGATION METHOD, ORGANISM-MODEL). The annotation task is framed as slot filling, i.e., pre-defined templates (corresponding to classes in SCIO) need to be populated based on evidence found in the document.

Figure 1 shows an example template which unfolds the ORGANISM-MODEL concept from SCIO into slots (corresponding to ontological properties) and their fillers. Fillers for such templates may be distributed across an entire document, which holds in particular for complex, recursively substructured templates such as RESULT.

4 User Interface and Annotation Workflow

Our web application provides three views to the user, which reflect the typical annotation workflow that is encountered in (slot filling) annotation tasks: (i) entity annotation, (ii) curation, (iii) template instantiation and slot filling.

4.1 Entity Annotation

In order to annotate entities at the level of tokens in the text, we support two workflows: Annotations are created by an annotator from scratch, or existing annotations can be imported in order to be altered by the annotator. In SANTO, this is done in the entity annotation view (cf. Figure 2). A new annotation can be created by selecting tokens in the text. Being forced to span full tokens, annotations are automatically adjusted to the onset of the first and the offset of the last token. Entity types are chosen from a drop-down menu. Discontinuous annotations are supported as well. Existing annotations can be altered by expanding or shrinking their span, or by changing the entity type (without the need to remove the existing annotation and adding a new one). After a document has been marked as complete by an annotator, it is available for curation.

4.2 Curation

The curation view is designed to help a curator adjudicating possibly divergent mention annotations produced by several annotators. For that purpose, the curator can select annotations of multiple annotators that should be displayed for adjudication. Each annotator corresponds to a unique color. As can be seen from Figure 3, all annotations are arranged on top of each other in order to make divergences clearly visible at a glance. Based on this visualization, the curator can select individual annotations and accept or reject them. Accepted annotations are marked with dotted boundaries while keeping the color. During curation, the original annotations are not altered but cloned into a new curation document which serves as the basis for the subsequent slot filling.

4.3 Template Instantiation and Slot Filling

In this mode, SANTO provides a combined view of curated entity annotations and pre-defined tem-

---

2 The list can be filtered by name prefixes or taxonomically.
3 If no curation is needed, all annotations can be easily batch-accepted.
plates to be filled (cf. Figure 4). During template filling, it is no longer possible to change entity annotations. Instead, the goal is to support the user in instantiating the correct **cardinality** of different template types, and **filling their slots** with previously annotated entities, while taking **referential grounding** into account where necessary.

**Cardinality of Templates.** Often, the number of instantiations of a particular template (e.g., number of **EXPERIMENTAL GROUPS** in a study) is variable and not known a priori. Thus, correctly determining the cardinality of the set of instantiated templates per type is a crucial subproblem in the slot filling task. In SANTO, users can add, remove, or duplicate instances of referential templates and rename them for easy reference (e.g., naming **EXPERIMENTAL GROUPS** as “ChABC-treated group” or “Untreated control group”).

**Slot Filling.** Based on their pre-defined role in the annotation schema, we distinguish three different types of slots in a template: **simple slots**, **recursive slots** and **template slots**.

Simple slots are directly filled with entity annotations (e.g., **STATISTICAL TEST** in Figure 5). In addition to being filled with entity annotations, recursive slots have subordinate slots (e.g., **INVESTIGATION METHOD** in Figure 5, marked with a drop-down symbol on the left). Simple and recursive slots can either be filled by clicking on an existing annotation in the document, or by selecting an annotation from a drop-down menu that appears when clicking on the slot. Here, it is also possible to select an entity type without textual reference, if necessary (e.g., **JUDGEMENT** in Figure 5). In order to facilitate slot filling, all suitable annotation candidates are highlighted in the document while hovering over a slot and vice versa. Template slots are special cases of recursive slots in order to facilitate referential grounding (see below).

---

4Based on type constraints defined in the ontology.

---

**Referential Grounding.** Slot filling tasks require referential grounding when several instances of complex sub-templates have to be distributed over various governing templates; e.g., an instance of an **EXPERIMENTAL GROUP** might take part in several, but not all of the **RESULTS** reported in a study. Moreover, the same instance of an **EXPERIMENTAL GROUP** may fill the **TARGET EXPERIMENTAL GROUP** slot in one **RESULT** and the **REFERENCE EXPERIMENTAL GROUP** in another. SANTO supports these cases by explicitly assigning previously instantiated templates to appropriate template slots via a drop-down menu (cf. Figure 5 for slot **REFERENCE EXPERIMENTAL GROUP**), thus ensuring referential uniqueness of fillers. Only templates of the appropriate type are available for selection.

---

**Figure 4:** Slot filling view displaying a combined view of curated entity annotations at text level (left) and templates to be filled (right). Type compatibilities between annotated entities and possible target slots are automatically highlighted by the tool.

**Figure 5:** Slot filling view (template pane only) displaying the six main templates according to SCIO. The **RESULT** template is opened; it includes simple slots (JUDGEMENT), recursive slots (e.g., INVESTIGATION METHOD) and template slots (e.g., REFERENCE EXPERIMENTAL GROUP, selectable via drop-down).
5 Technical Properties

Application Details. The application is hosted on a standard Linux web stack using Apache 2.4, PHP 5.6 with a MySQL (version 5.5) database backend. The frontend uses HTML-5, CSS-3, jQuery with jQuery UI to provide a fast and robust user experience compatible across modern browsers.

Specification of Annotation Scheme. As initial input, the system requires a description of the underlying annotation scheme which can be derived from an ontology. This includes the description of entity types and roles (Class vs. NamedIndividual etc.), the hierarchical structure of entities (subclass relations) and their relational structure (object-type and data-type properties). Further, the template types that are available for slot filling need to be specified. All specifications are provided in configuration files in CSV format. Thus, instead of a fully-fledged ontology, an ad-hoc information model can be provided as well. Support for automatically translating an RDF ontology to the custom specification format is currently limited to the following syntactic elements: owl:Class, rdfs:subClassOf, owl:NamedIndividual, rdf:type, owl:objectTypeProperty, owl:dataTypeProperty, and rdfs:description.

Pre-Processing of Input Documents. All documents need to be pre-processed before being uploaded to the application. This includes sentence splitting and tokenization. Note that entity annotations are limited to sentence boundaries and only full tokens can be annotated.

Import and Export of Annotations. For each document, it is possible to import entity annotations for individual annotators from different sources (e.g., external annotation platforms or automated systems). The syntactic format for importing annotations is based on a shallow tab-separated formalism in order to facilitate a wide range of annotation sources such as existing information extraction tools, lexicon matching routines, or manual workflows. Entity and template annotations can be easily exported. While entity annotations are simply described in a CSV-structured file, template annotations are exported as RDF following the underlying ontology and annotation schema.

User Management. User roles in SANTO comprise annotators and curators. Individual views per annotator and curator enable multiple annotations, curations and slot fillings for each document. The web-based architecture supports multiple annotations in parallel.

6 Conclusion

We have presented SANTO, the first web-based annotation tool which enables easy annotation of complex relational structures in a template-based slot filling setting. The tool is web-based and flexibly configurable. Being designed in order to address typical user needs in slot filling workflows, we expect SANTO to have a positive impact on research aiming at automated acquisition of comprehensive, highly structured domain knowledge from textual sources, as in biomedical information extraction, digital humanities, or similar fields.

Acknowledgments

This work has been funded by the Federal Ministry of Education and Research (BMBF, Germany) in the PSINK project (grant 031L0028A). We thank our PSINK colleagues Nicole Brazda, Veronica Estrada, Jessica Schira and Hans Werner Müller for helping shape the user requirements for SANTO.

References


https://jquery.com
https://jqueryui.com


