

Bauhaus-Universität Weimar
Faculty of Media
Degree Programme Medieninformatik

Requirements engineering for natural-language annotation tasks

Bachelor's Thesis

Vincent Söllner

1. Referee: Prof. Dr. Benno Stein
2. Referee: Prof. Dr. Andreas Jakoby

Submission date: May 17, 2021

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Weimar, May 17, 2021

.....
Vincent Söllner

Abstract

In the field of natural language processing it has become the norm to utilize various forms of annotation software to help with collecting, creating and curating of an ever growing mass of data. The increasing amount of various annotation tasks has led to a similarly increasing quantity of different annotation software, such that the selection of suitable tools for an annotation task has become a non-trivial task of its own.

This thesis examines the current landscape of software solutions for NLP annotation tasks by identifying recurring tasks and phenomena, comparing existing annotation software with respect to commonalities and differences, and by identifying core functionalities needed for most annotation tasks. We propose a new type of annotation pattern to formalize the process of task analysis for annotation software, creating a catalogue of requirements for annotation software in the process.

Contents

1	Introduction	1
2	Background	4
2.1	Related research	4
2.2	Phenomena	5
2.2.1	Properties	8
2.2.2	Requirements for annotation	9
3	Review of existing software	13
3.1	Existing annotation software for textual data	13
3.2	Overview	14
3.2.1	Common annotation tasks	15
3.2.2	Common features	16
4	Software patterns	21
4.1	Interaction Design patterns	22
4.1.1	Import & Export	23
4.1.2	Automation	28
4.1.3	Source document management	30
4.1.4	Annotation level interactions	32
4.1.5	Quality assurance	34
4.1.6	Mode of interaction/modality	36
4.2	Software Design patterns	39
5	Annotation patterns	41
5.1	Proposal for new pattern structure	41
5.2	Annotation patterns	42
5.2.1	Selecting markables	43
5.2.2	Creating labels	44
5.2.3	Selecting markables via label	45
5.2.4	Editing annotations	46
5.2.5	Deleting annotations	47

5.2.6	Creating directed relations between markables	48
5.2.7	Creating undirected relations between markables	50
5.2.8	Selecting relations	52
5.2.9	Editing relations	52
5.2.10	Deleting relations	53
5.2.11	Reusing annotations	54
6	Conclusion and Future Work	56
A	Documentation	57
A.1	Documentation of annotation tasks in the tested software	57
A.2	Documentation of the installation process	57
A.3	Documentation of known interface issues in the annotation software	57
B	List of phenomena	58
	Bibliography	63

Chapter 1

Introduction

A trend observable nowadays in most areas of research is the steadily increasing amount of data. Every single day, video footage worth years of viewing time is uploaded to various internet platforms, and millions of pictures are posted on social media. Thousands of songs and other audio recordings are released, and millions of tweets, status-updates and other text-based posts appear on social media and forums. Vast quantities of books, papers and articles are being published, numerical values such as measurements of weather stations are catalogued, transaction data of various cryptocurrencies are made public on their respective blockchains, and point-clouds and 3D models are created and shared. In each and any domain of knowledge, data is being created and shared at any given moment.

The present thesis will be focusing on textual data, since efforts to annotate this type of data have not only started long before the rise of more modern forms of media, and therefore practices of annotation are well established. Furthermore, it is in most cases easier to apply the observations about textual data to more complex data than it is to do the reverse. Most of the textual data comes in the form of raw data, which often is not of immediate use for research purposes, because many research applications make use of tools or methods which necessitate the raw data to be augmented in order for it to be usable.

For textual data there are three main steps to be considered:

- Data acquisition: How the data is collected, e.g. by scraping websites, scanning physical documents, importing existing text corpora, etc.
- Data cleaning: How unnecessary information is removed from the collected data, e.g. by stripping out formatting, removing duplicate elements (if the existence of the duplicate is not useful information), removing incomplete records (if identifiable as such), etc.

DebateGuy9000; 2021:02:06; 12345; Pro; Another reason why we should abolish copyright all together is that it prevents my favourite movie "Ninja snakes" from entering public domain until 2087, I hate that! Unrelated note: I heard they're currently filming Ninja snakes 2????? Can't wait to see it!

Figure 1.1: Example of raw textual data

(Username)	(Date)	(ThrD)	(Sta)
DebateGuy9000	2021:02:06	12345	Pro
Another reason why we should abolish copyright all together is that it prevents my favourite movie "Ninja snakes" from entering public domain until 2087, I hate that!			
(Keyword)		(Argument)	(Keyword)
(NonArgument)			(Sentiment)
Unrelated note: I heard they're currently filming Ninja snakes 2????? Can't wait to see it!			

Figure 1.2: Example of argument text annotated in WebAnno

- Data enrichment: How existing data is enhanced by adding additional information, e.g. by annotating the document, adding metadata, putting the data into a context with or a relation to other data, etc.

Consider the following example: a system is supposed to be trained to recognize pro and con arguments in argumentative texts. For this purpose example text segments are required that represent pro and con arguments. In the data acquisition step a selected debate website is scraped for its contents. The raw textual data may contain individual forum posts, see Figure 1.1.

The text does contain information useful for the intended task, but in the current form it is not in a state that is immediately evaluable by most software, hence some preparation is still needed.

There are tools for the tasks of cleaning, structuring and labelling the data, but they tend to be very specialized and were often developed with only one single use case in mind. In our example there are different ways the task of data cleaning could be approached: e.g. not to scrape all arguments indiscriminately, but filter beforehand for certain completeness conditions such as an existing stance. Another example would be to strip all posts after scraping that do not fulfill certain criteria, such as a post having a certain stance towards the argument associated.

In our example, various kinds of information that are scattered throughout the text may need to be marked with labels, such as who posted the argument, when it was posted, what the stance on the overarching topic was, which part of the text actually is an argument and which part may be unrelated, or phrases that hint towards strong feelings about the topic, etc. Figure 1.2 shows what the same text looks like after applying labels to the appropriate parts of the text.

Which tools the annotation software provides often depends on the area the software was developed for, with the spectrum of available tools being as varied and specialized as the spectrum of available fields of research. In the field of Natural Language Processing (NLP for short) annotation software tends to provide tools for the annotation of various, recurring language phenomena and tasks, such as part-of-speech-tagging, identification of time, dates and named entities, etc.

Not all annotation problems and tasks are of recurring nature however, this resulting in a large variety of extremely specialized annotation software, often developed by a research group to tackle a single specific annotation project. This trend of creating new tools for every research task (instead of relying on preexisting solutions or expanding existing software), points to an underlying problem of annotation software as a research field. The latter is severely lacking a structure that would allow researchers to make use of existing knowledge and software, rather than writing annotation software for a single specific annotation project or trying to make an existing solution work that was not designed to be usable for a wide range of tasks.

In this thesis we develop and formalise overarching annotation categories, annotation tasks and annotation-related software patterns by examining various NLP phenomena, comparing existing annotation software for differences and commonalities and expanding existing software pattern formats. We lay the groundwork for future research efforts by facilitating the identification of existing solutions to similar annotation problems, and proposes a new approach to designing annotation software based on patterns. We contribute a collection of linguistic phenomena observed in various NLP related research papers, detailed documentation of annotation tasks in a various annotation tools, documentation of the installation process of all annotation software tested in this thesis and a collection of observed interface issues and annotation edge cases in various annotation tools. We also define several new Interaction Design patterns about annotation-adjacent tasks, and several Annotation Patterns in a newly developed Annotation Pattern format.

Chapter 2

Background

2.1 Related research

A myriad number of raw data exists nowadays that in many cases only becomes valuable for research through enrichment and interpretation. The way we interpret data is strongly influenced by the knowledge domain it belongs to and the form of data, e.g. audio, images, videos, 3D models, text, etc. One of the ways data is interpreted, or meaning and structure is extracted from it, is by means of annotation. In this recent thesis we focus purely on the requirements of annotation tasks in the field of NLP on textual data. However, annotation software also exists in other areas of research and is being developed for non-textual data. Nixon and Troncy (2014) surveyed semantic media annotation tools in regards to Linked Media principles. Media annotation tools possess requirements that do not apply to the annotation of textual data, e.g. support for pixel annotation for image or video based instance segmentation tasks (Watanabe and Wolf (2019)), time-series annotation for audio labeling tasks (Coviello et al. (2011)) or point-cloud annotation for 3D annotation tasks (O’ Mahony et al. (2019)).

The ACL Anthology (2021), as of time of writing, hosts “[...]64745 papers on the study of computational linguistics and natural language processing” and was one of the most useful sources for discovering papers during the course of research for this thesis. Various areas of research exist that utilize annotation tools for textual data, but introduce domain-specific requirements which cannot be described as requirements for natural-language annotation tasks at large, e.g. biomedical annotation (Jovanovic and Bagheri (2017); Neves and Leser (2012)). Complaints about missing functionality in these tools were taken into consideration during the development of basic annotation units and patterns for this thesis, but their influence was limited; most complaints were of universal nature and were also observed in general purpose annotation tools,

e.g. insufficient documentation, installation problems and proprietary import and export formats.

This observed universal nature of usability complaints was one of the main motivations for this thesis. Previous efforts in this field try to approach the problem from different angles. Neves and Seva (2019) curated a list of 78 different annotation tools, 15 of which they reviewed in detail according to a proposed list of evaluation criteria. This list of criteria directly influenced the formulation of some of the annotation patterns proposed in chapter 5. Reidsma et al. (2004) curated a similar list of criteria, as well as a list of properties of annotation problems, and discussed how they can influence the design of annotation software. Burghardt (2012) evaluated three annotation tools and formulated a list of usability recommendations based on problems observed during the review of the tools. These recommendations were the basis for various annotation-adjacent patterns proposed in section 4.1.

Lin (2016) notes that “[t]here has not been careful consideration of software complexity in current annotation tools. Due to the problems of complexity, new annotation tools must reimplement common annotation features despite the availability of implementations in open sourced tools.”. He further introduces a ClojureScript library called Notate, which according to the paper “provides pure functions for building annotation tools with”. Unfortunately we were not able to locate the Notate library, but only the paper describing the functionality of this library.

The Text Encoding Initiative (TEI (nd)) define a community-developed text encoding format, used by various annotation projects. Unfortunately it is just one of many competing formats, rather than a definitive industry standard. The lack of a single universally accepted file format was one of the motivations for comparing file formats across multiple annotation tools tested for this thesis (Figure 3.2).

Folmer (nd) and Gamma et al. (1994) defined different pattern formats and terminology that were used as basis for the patterns proposed in this recent thesis. These patterns are intended to be used in combination with existing User Interface patterns like the ones provided by Toxboe (nd) during the development of annotation software.

2.2 Phenomena

The scope of research in the framework of the present thesis was limited to the annotation of textual data. However, NLP is still a large field with many different annotation tasks and various types of source documents. Consequently, a myriad number of phenomena exist that would be candidates for annotation,

with new ones regularly being discovered or selected for annotation for the first time. Moreover, it is worth considering that many phenomena have overlaps, are poorly or insufficiently defined, or are so complex or domain-specific that they are incomprehensible to non-experts. Therefore, when trying to compile a list of existing NLP phenomena, it is impossible to ensure any completeness of such a list. Ide and Pustejovsky (2017) describe linguistic annotation as follows:

“Linguistic annotation involves the association of descriptive or analytic notations with language data. The raw data may be textual, drawn from any source or genre, or it may be in the form of time functions (audio, video and/or physiological recordings). The annotations themselves may include transcriptions of all sorts (from phonetic features to discourse structures), part-of-speech and sense tags, syntactic analyses, named entity labels, semantic role labels, time and event identification, co-reference chains, discourse-level analyses, and many others.”

In this thesis the focus is limited to the annotation of textual data, not considered are data in the form of time functions. A list of phenomena is gathered below; it is sorted according to the level at which annotation takes place. The selection of phenomena included in the list is exemplary and intended to illustrate common sizes of annotation units in NLP annotation projects. A more exhaustive list of phenomena was produced during the research of this thesis, this extended list including sources can be found in Appendix B.

These phenomena have been gathered from various papers on topics related to NLP, such as annotation guidelines, linguistic annotation, argumentative text analysis, etc. This results in a collection of phenomena of varying granularity and specificity. In this chapter we include a few examples of phenomena of various annotation unit sizes. The list is intended to illustrate possible levels of granularity of annotation, and to highlight the possibility that different levels of granularity are one of the reasons for the wide range of annotation software.

We categorize the phenomena according to the following levels: sub-word level, word level, clause level, sentence level and discourse level. We propose these categories based on observations made during the collection phenomena for this paper. A considerably longer list of linguistic phenomena was collected during the research of this thesis, the complete list including sources can be found in Appendix B.

- The sub-word level contains any phenomenon comprised of annotation units smaller than a word, e.g. characters or syllables.

- The word level contains any phenomenon annotated via one annotation unit of the size 'word'.
- The clause level contains any phenomenon comprised of the annotation unit 'clause'.
- The sentence level contains any phenomenon annotated via one annotation unit of the size 'sentence'.
- The discourse level contains any phenomenon at the discourse level. Nordquist (2021) defines discourse as follows: "In linguistics, discourse refers to a unit of language longer than a single sentence".

We emphasize that most phenomena can be annotated at different levels. These phenomena therefore fall into different categories, depending on the granularity chosen by the annotator for the annotation task.

Consider the following example: an English sentence and its German translation are annotated in a text-alignment task. The annotation project might decide to annotate on the word- or even sub-word level and align words with their translated counterparts. However, it is also thinkable that the sentences were already aligned in a previous annotation project as preparation for this annotation task. An English document was paired with its German translation, and in this previous work the text alignment was annotated on the sentence-level. Both are text-alignment tasks containing the same sentences, yet they are annotated at different levels.

Sub-word level: character-level annotation in the Chinese language, as described by Li et al. (2019).

Word level: named entity annotation. Consider the following example: in the sentence "Sebastian was born in Berlin" the named entity "Sebastian" and "Berlin" would both be annotated at the word level. Sebastian would get labelled as entity-type Person, while Berlin would get labelled as entity-type Location.

Clause level: subordinate clause annotation. Consider the following example: in the sentence "Sebastian had a Bavarian dialect, despite being born in Berlin" the subordinate clause "despite being born in Berlin" would be annotated at the clause level.

Sentence level: comparative illusion. Consider the following example: the sentence "more people have visited Sebastian in Berlin than I have" seems grammatically correct at first glance, but does not actually have

any understandable meaning on closer consideration. Annotation would likely take place at the sentence level, and the entire sentence would get labelled as comparative illusion.

Discourse level: argument annotation. Consider the following example: the sentences "Sebastian should move out of Berlin. He can barely afford the rent there." contains an argument, which is made up of a conclusion (Sebastian should move out of Berlin) and a premise (He can barely afford the rent there). The argument would get annotated at the discourse level.

2.2.1 Properties

The present study seeks to identify common properties of the phenomena collected. Since the starting point for collecting phenomena was the selection of already existing annotated phenomena, it is necessary to acknowledge an inherent selection bias: the present collection does not - and cannot - contain phenomena which are not currently annotateable. Any generalizations derived from the collected list of phenomena are not applicable to phenomena which are not annotateable.

One of the most important properties of any phenomenon is their annotation unit size. In the previous section the collected phenomena were ordered according to these annotation unit sizes. While the categories chosen for the grouping of phenomena are motivated by the linguistic nature of the phenomena collected, they are not definitive. Other groupings are conceivable and are potentially better fits for differently sized collections of phenomena.

Regardless of the chosen grouping size, two distinct types of phenomena can be identified:

- Phenomena annotateable via a single annotation unit (e.g. word level).
- Phenomena only annotateable via a collection of annotation units (e.g. multi-sentence level).

It is technically possible to annotate phenomena with more annotation units than necessary (e.g. annotating a word level phenomenon not with the annotation unit size 'word', but with a collection of annotation units of the size 'character'). However, it was assumed here that the definitions of phenomena typically follow Occam's razor and attempt to explain the phenomena in the simplest form possible. Similarly, while it is technically possible to annotate any phenomenon with a single annotation unit by introducing more complicated annotation units, it was assumed here that the definitions of phenomena typically attempt to keep the annotation units necessary to annotate the phenomenon as simple as possible.

2.2.2 Requirements for annotation

To enable researchers to annotate the observed categories of phenomena, definitions of fundamental annotation units of textual annotation are needed. We observe that annotation units are typically defined by research teams for their specific annotation projects. This results in a large variety of definitions for annotation units referred to by similar terminology. An example of this are guidelines for two different coreference annotation projects, Lanfranchi et al. (2013) and Hirschman (1997). Both use project-specific definitions for "markables", with slight differences between the definitions, despite the already very specific task of coreference annotation. Other definitions for the term "markable" exist outside the area of coreference annotation, with even bigger differences.

In our research we managed to identify Dipper et al. (2004b) as the only paper attempting to generalize annotation units across several annotation projects. To quote their paper:

The SFB "Information structure: the linguistic means for structuring utterances, sentences and texts" consists of 12 individual research projects from disciplines such as theoretical linguistics, psycholinguistics, first and second language acquisition, typology, and historical linguistics. The overarching objective of these projects is the investigation of information structure (IS). This is an area well-known to be prone to terminological or even conceptual confusion – many different theories of how to partition utterances into IS-relevant segments compete with each other, and, furthermore, there is little agreement on what level(s) of utterance representation IS should be located.

Our thesis attempted to use the definitions proposed in the paper wherever possible, but modified and expanded the definitions when it became clear that some formulations were insufficient to describe some annotations identified during its research.

We extend the existing definitions based on our observations.

Markables are considered to be the basic units of annotation of textual data. This thesis proposes the following definition for markables:

Markables (segments): The basic units referenced by the annotation are defined by inclusion/embedding (e.g., <markable>...</markable>) within the source material that is to be annotated vs. specifying a set of start and endpoints (e.g., <markable span="id1..id4|,

`id7..id12]/>` or specifying a set of start-points and offsets (e.g., `<markable span="id1+3[id7+5]/>`), which refer to character positions within a separate source document that is to be annotated.

The definition is derived from the markable definition by Dipper et al. (2004b). The proposed modification addresses several issues with the original formulation:

Markables (segments): The basic units referenced by the annotation are defined by inclusion/embedding (e.g., `<markable>...</markable>`) vs. specifying a start and endpoint (e.g., `<markable span="id2..id4"/>`).

The original definition describes markables in two different ways: referenced by embedded tags (commonly referred to as "inline") and markables referenced by specified start and endpoints (commonly referred to as "stand off"). The reason for the inclusion of both is the incongruity of both approaches. Dipper et al. (2004b) note that conflicting hierarchies, such as overlapping markables or trees, may only be represented by markables which are referenced by their start and endpoints. Despite this limitation and the adoption of stand off-annotation as best practice, annotation based on inline markables still exists and therefore needs to be included in a definition of the term markable.

The proposed modification attempts to address other limitations of this definition of markables. Various phenomena are composed of discontinuous text spans, that is to say, segments of text which are interrupted by other text segments which do not belong to the phenomena that is supposed to be annotated. Annotators have worked around this issue by annotating such phenomena with separate markables. However, such a workaround is counter-intuitive to annotators as it does not represent the properties of the annotated phenomenon accurately. Another risk of this approach to annotating discontinuous text spans is the distortion of Inter-Annotator Agreement (IAA) calculation, as illustrated in Figure 2.1.

To address this limitation, the new definition allows for the inclusion of several start- and endpoints in a single markable in a stand off-format. The annotation of discontinuous spans remains a limitation in the inline format.

Another limitation is the incomplete definition of markables in a "stand-off" format. Specifying a start and endpoint is sufficient to describe a text segment, but an alternate specification via startpoint and offset also exists. The modified definition addresses this oversight by expanding the definition to include this alternate specification with start-points and offsets.

Annotators are supposed to annotate the main clause of a sentence.
The main clause in the example is interrupted by an embedded subordinate clause.

Correct annotation	Incorrect annotation
--------------------	----------------------

Annotator A: This sentence, which is annotated as discontinuous span, has an embedded clause.
Annotator B: This sentence, which is annotated as discontinuous span, has an embedded clause.

The annotators have no agreement about the correct annotation of this phenomenon.
The implementation with markable as discontinuous span correctly reflects this.

Annotator A: This sentence, which is annotated with two separate markables, has an embedded clause.
Annotator B: This sentence, which is annotated with two separate markables, has an embedded clause.

The annotators have no agreement about the correct annotation of this phenomenon.
However, the implementation with two separate markables incorrectly implies that an agreement exists in 50% of the annotated cases.

Figure 2.1: Example of agreement distortion for implementation of phenomenon as one discontinuous markable, vs. two continuous markables

Labels are a store of properties, typically assigned to markables or other annotation units to annotate attributes of the annotation unit. "Labels" is a common term for this functionality, but no universally valid definition of the term seems to exist. This problem stems from the fact that several competing terms exist, many of which are synonymous, adjacent, or overlapping in their meaning. Even terms which aren't technically synonymous are often used interchangeably, likely due to the fact that few papers about annotation actually focus on the minute differences between low level terminology, and instead explain higher level functionality. In these cases, the usage of terms like label, tag, property and similar terms is often considered self-explanatory through context. Terms often used in a similar manner include, but aren't limited to: labels, tags, properties and attributes.

The paper by Dipper et al. (2004b) is very abstract in this regard. The closest thing mentioned to labels is a given example of part-of-speech tags as a data structure:

Secondary data This criterion concerns properties of the annotations. [...]

(5) Data structure: Secondary data consist of:

- (a) atomic features of a markable (e.g., part-of-speech tags)
- [...]

No explanation of what other atomic features a markable might possess is given. This definition is abstracted to a degree that we find unwieldy.

This thesis proposes a new definition for the term "label":

Label: A store of properties in key-value format which describe a markable.

This new definition is inspired by the entity-attribute-value model, a well established logical data model used by various databases. The markable here assumes the role of the described entity.

Relations are another fundamental building block of annotation projects, describing relationships between annotation units.

Dipper et al. (2004b) define relations by example:

Secondary data This criterion concerns properties of the annotations. [...]

(5) Data structure: Secondary data consist of:

[...] (b) relations between markables: (undirected) relations, pointers

While the definition is sufficient to describe all known relation types (directed and undirected), the definition is hindered by its formatting as subitem in a list. To provide a more readable definition and emphasize the possibility of a markable to have a relation to itself or more than one other markable, we propose the following new definition:

Relation: a directed or undirected link between a markable and itself or an arbitrary amount of other markables.

This new definition is inspired by the definition of vertices in graph theory, a well established field of research. In combination with our proposed definitions for markables and labels, it should theoretically be possible to implement annotations as graphs, which in turn would allow the transfer of graph-based algorithms into the field of annotations.

To our knowledge, no previous efforts to define annotation units for the annotation of textual data in a graph-like structure exist. The closest efforts appears to be limited to Zhao et al. (2017), who propose "[...]a dynamic graph visualization that enables meta-analysis of data based on user-authored annotations", as well as Maeda et al. (2001), who propose "[...] an efficient and expressive data model for linguistic annotations of time-series data". If previous efforts exist, we weren't able to identify them. It is worth pointing out that internet searches about this topic are severely complicated by the existence of various papers about the topic of annotating graphs (plots).

Chapter 3

Review of existing software

3.1 Existing annotation software for textual data

To help with the annotation of language phenomena, various annotation tools were developed. Annotation software often developed naturally along the lines of the research focus the developers had, at times becoming increasingly specific tools for niche annotation tasks, other times getting expanded to support a variety of annotation tasks. Many of these tools were originally only intended for internal use, and only got released to the public at a later point, resulting in a lot of parallel development and redundant solutions to similar problems. A few tools with more general annotation capabilities established themselves over time as standard go-to tools for annotation tasks. However, if limited capabilities or complicated workflows make the tools unsuited for the intended annotation project, researchers often decide to develop their own annotation software from scratch instead. This results in a large collection of annotation tools with limited audience.

Various attempts to catalogue and/or evaluate this plethora of annotation software exist, such as Neves and Seva (2020), Neves and Seva (2019), Neves and Leser (2012) (focused on biomedical literature) and Nixon and Troncy (2014) (focused on semantic annotation of media resources). Surveys like these help in getting an overview over the existing annotation software landscape and in the selection of existing tools suited for specific annotation tasks, if such a tool exists. Our thesis however focuses on the identification of commonalities between some of these annotation tools. We attempt to develop common terminology to communicate recurring features commonly found in annotation software, allowing the developers of future annotation tools to make use of existing solutions to known problems, saving time and resources in the process.

For this purpose a sample of annotation tools were selected based on a combination of previous familiarity with annotation software, frequent mentions in

other annotation papers, and reported popularity according to recent survey (A survey of NLP Annotation Platforms, 2020). The tools were compared in regards to support of the basic annotation units as defined in subsection 2.2.2. We checked for support of markables, whether one or multiple labels could be applied to a markable, whether the software supported highlights as a different store or properties (compared to labels), whether these types of storage could be combined, support for relations (directed, undirected and chains) and compared the results across the tested tools.

3.2 Overview

For this thesis we tested Appraise (Federmann (2012)), brat (Stenetorp et al. (2012)), CorA (Bollmann et al. (2014)), doccano (Nakayama et al. (2018)), FLAT (van Gompel et al. (nd); van Gompel and Reynaert (2013)), INCEPTION (Klie et al. (2018)), MMAX2 (Müller and Strube (2006)), Swan (Gühring et al. (2016)), WAT-SL (Kiesel et al. (2017)), WebAnno (Eckart de Castilho et al. (2016)) and Yedda (Yang et al. (2018)).

Each software was locally installed for testing. The documentation of the installation process, thoroughly documented for each annotation software, can be found in section A.2. In cases where the installation failed, installation was attempted a second time on different hardware. Installation failed for two of the selected tools, Appraise and CorA.

An unfortunate observation during the testing process was that even the tools we did manage to install were often not trivial to install. Outdated or incomplete documentation were a recurring issue, as well as outdated dependencies in pip-based installations breaking the installation process;

The problem is illustrated by observations of Neves and Seva (2020) and Neves and Seva (2019), who included ease of installation and (in)ability to install in their evaluation of annotation tools.

A major contribution of this thesis was the examination and documentation of the interface, supported formats and various features of each software, as well as the curation of known interface issues in various annotation tools.

- A full documentation of annotation tasks in the tested software can be found in section A.1.
- A full documentation of the installation process of the tested software can be found in section A.2.
- A full documentation of known interface issues in annotation software can be found in section A.3.

3.2.1 Common annotation tasks

Annotation tasks are just one part of the workflow of a typical software-driven annotation project; they need to be distinguished from annotation-adjacent tasks that commonly occur in the same project, but aren't directly involved with the annotation. The process of a typical annotation project can be split into three stages: data gathering, data cleaning and data annotation.

- Data acquisition contains all tasks regarding the collection of documents and creation of a corpus of source material for later annotation. The tasks during this step depend on the source of the data. E.g. website scraping, document digitization, document extraction from an existing dataset, etc.
- Data cleaning contains all tasks regarding the preparation of the collected data for annotation. The tasks during this step depend both on the source of the data and the details of the intended annotation in a later step. E.g. stripping unwanted or incorrect data, automated tokenization, normalisation, stemming, etc.
- Data annotation contains all tasks regarding the actual annotation of data. The tasks during this step depend on the annotation guidelines.

Annotation guidelines are often written iteratively. One method for this is the MATTER methodology as described by (Pustejovsky and Stubbs, 2012, p.23-32), though the training and testing steps can be ignored in the case of non-algorithmic annotation. Researchers typically start the development of their annotation guidelines by describing an initial model for what they want to annotate. Annotation schemes for specific phenomena are created, then tested on example documents. The proposed annotation scheme is evaluated for potential issues and results between annotators are compared. Once the causes of problems with the proposed scheme are identified, a new iteration of the annotation guidelines is proposed and the process starts from the beginning. These steps are repeated until the guidelines are sufficient to produce correct annotations in a repeatable manner.

These annotation schemes are then, if the chosen annotation software allows, converted into annotation types for the annotation project. In modern annotation projects it isn't unusual for annotation guidelines to be developed alongside specific annotation software. This development of annotation guidelines in tandem with the annotation software can speed up the development process, but can also introduce limitations to the annotation scheme, as software limitations are transferred into the annotation schemes. It is this stage after the creation of the annotation guidelines that existing annotation tools

tend to be designed for: annotating documents using predefined annotation schemes, by selecting markables within the document, assigning labels and indicating relations between markables. Comparing and correcting the annotations of different annotators and exporting the resulting corpus.

Optional, role-dependent and non-sequential steps can exist within this annotation workflow. For example, the tasks of an administrator in an annotation project might be to set up the annotation project within one of the selected annotation tools, but not necessarily to participate in the actual annotation themselves. In contrast, a crowd sourced annotator might get on-boarded onto the project after the guidelines are finalized to annotate the phenomena, but wouldn't be involved in the design of the actual project. Project-specific tasks might be related to methods such as the use of crowd-sourcing for annotation, e.g. making the annotation guidelines, corpus and software available to workers via some crowd-sourcing platform. Some annotation projects utilize automation, which has to be programmed or setup using existing tools, some projects make use of live data and require different data pipelines. And as diverse as the projects and associated workflows are, as diverse are the existing annotation tools and requirements or wishes researchers have for them.

An software to fulfill every step of the annotation workflow would be the preferred working environment for some researchers, while others prefer lightweight annotation tools for each individual task of the process, allowing for greater flexibility by choosing the appropriate tools for each step.

This chapter will focus on the part of the workflow addressed by all examined annotation software, the annotation tasks. However, even with this limitation differences between the tools already become apparent. The most basic annotation tasks can be categorized into two types: assigning labels to markables, and creating relations between markables.

3.2.2 Common features

One of the common features of the examined annotation tools, implemented differently across software, were the support for various import and export formats. As mentioned in section 2.2.2, two different ways of annotating markables exist: via inline annotation or stand off-annotation. Each of the formats have their own benefits and downsides, but stand off-annotations are considered the best practice nowadays.

Inline annotations require the source document or a working copy of the document to be modified when working on it. Markables are inserted directly into the text at the locations they respond to. This way of working with annotations has several benefits:

	brat	doccano	flat	INCEption	MMAX2	Swan	WAT-SL	WebAnno	YEDDA
Markables	True	True	True	True	True	True	True	True	True
Single property (stored as label)	True	False	False	True	False	True	False	True	False
Multi property (stored as label)	True	False	False	True	False	True	False	True	False
Highlights as separate property storage	False	True	True	False	True	False	True	False	True
Single property (stored as highlight)	True	True	True	True	True	False	True	True	True
Multi property (stored as highlight)	True	False	True	True	False	False	False	True	False
Multi property (combined)	True	False	True	True	False	True	False	True	False
Relations	True	False	True	True	True	True	False	True	False
Directed	True	False	True	True	True	True	False	True	False
Undirected	True	False	True	False	True	False	False	False	False
Chains	nan	nan	nan	nan	nan	nan	nan	True	nan

Figure 3.1: Overview of support for basic annotation types in tested software

- Simple annotations (just assigning labels to markables) and the visualization of them are very easy to achieve.
- Annotations in the inline format remain readable to human annotators even outside of annotation software.
- Modifications to the source document, such as fixing typos or grammar, are relatively easy to implement.

Annotators nowadays tend to consider the downsides to outweigh the benefits. Downsides include, but are not limited to:

- It is harder to implement relations.
- It is harder to implement the deletion of markables.
- Conflicting hierarchies of markables are impossible to implement.
- It is significantly harder to combine multiple sets of annotations.

One example of inline annotation is the FoLiA-format as described in (van Gompel, 2019, p.88-108).

Stand off-annotations do not require the modification of the source document. Annotations are stored in relation to character positions within the document, e.g. the start-point of the annotation is at character position 1000, and the endpoint of the annotation is at character position 1024. This way of working with annotations has several benefits and is widely considered to be the best practice:

- It is possible to implement conflicting hierarchies.
- It is easy to combine multiple sets of annotations.
- It is easier to implement relations.
- It is easier to implement the deletion of markables.
- Changes are easy to revert due to a non-destructive workflow.

Downsides of working with stand off-annotations include:

- It is almost impossible to modify the source document. If such functionality is implemented, it needs to include functionality to update all annotations on the document afterwards, as the modification of the source document can falsify the calculation of offsets.
- Annotations in the stand off format can be hard or even impossible to read for human annotators outside of annotation software.
- It is comparatively harder to implement simple annotations (just assigning labels to markables).

One example for stand off annotation is the brat standoff format as described in (the brat documentation).

Most annotation software nowadays use formats that belong to the stand-off category, though exceptions exist. There is however not one, but a large variety of different formats that store annotations in a stand-off format. Even further, certain formats may appear similar at first glance (e.g. two formats being both stand-off having an XML file extension), but still be incompatible. In Figure 3.2 the different import and export formats of the tested software are listed, in which we clustered formats according to underlying technical implementations (JSON, CSV, XML, plaintext, other), this however doesn't necessarily mean that two XML based import format of two different annotation tools are compatible with one another. To determine cross-compatibility between all tools and formats would've exceeded the scope of this thesis, but might be worth considering for future research.

We also documented 'noteworthy' formats, 'noteworthy' being the loosely defined criteria whether we noticed the adoption of a format by other software, which implies some useful properties of the format recognized by users in the specific knowledge domain. It was surprising to note that few of the tested tools had exactly the same export and import formats.

One important feature of annotation software is the ability to present relevant information to annotators. A common implementation of this is to show

	brat	doccano	flat	INCEption	MMAX2	Swan	WAT-SL	WebAnno	YEDDA
Document formats	nan	nan	nan	nan	nan	nan	nan	nan	nan
(import) plaintext support	TRUE	TRUE	FALSE	TRUE	False	TRUE	TRUE	TRUE	TRUE
(import) JSON support	FALSE	TRUE	FALSE	FALSE	False	FALSE	FALSE	FALSE	FALSE
(import) CSV support	FALSE	TRUE	FALSE	FALSE	False	FALSE	FALSE	FALSE	FALSE
(import) XML support	FALSE	FALSE	FALSE	TRUE	True	FALSE	FALSE	FALSE	FALSE
(import) other format	FALSE	TRUE	TRUE	TRUE	False	FALSE	FALSE	TRUE	FALSE
Noteworthy import formats	nan	CoNLL	FoLiA	WebAnno TSV	nan	nan	nan	WebAnno TSV	nan
Import format type	nan	plaintext based	XML-based	plaintext based	nan	nan	nan	plaintext based	nan
Standoff / separate annotation format	TRUE	TRUE	Mix	TRUE	True	TRUE	TRUE	TRUE	TRUE
Annotation formats	nan	nan	nan	nan	nan	nan	nan	nan	nan
(export) plaintext support	FALSE	FALSE	FALSE	TRUE	False	FALSE	FALSE	TRUE	FALSE
(export) JSON support	FALSE	TRUE	FALSE	FALSE	False	FALSE	FALSE	FALSE	FALSE
(export) CSV support	FALSE	FALSE	FALSE	FALSE	False	FALSE	TRUE	FALSE	FALSE
(export) XML support	FALSE	FALSE	FALSE	TRUE	True	TRUE	FALSE	FALSE	FALSE
(export) other format	TRUE	FALSE	FALSE	TRUE	False	TRUE	TRUE	TRUE	TRUE
Noteworthy export formats	.ann	nan	FoLiA	WebAnno TSV	nan	UIMA XML	.ann	WebAnno TSV	.ann
Export format type	plaintext based	nan	XML-based	plaintext based	nan	XML-based	plaintext based	plaintext based	plaintext based

Figure 3.2: Overview of supported import and export formats in tested software

additional information about any selected annotation to the user. We documented which interactions by users triggered the display of such additional information for the three most common annotation categories. The information can be found in Figure 3.3. One surprising observation was that single clicks triggering the display of additional information tended to be inconsistent across different types of annotations within the same software.

We also documented which type of information specifically was communicated via tooltip. Most tools had a predefined set of information on display in their tooltips, with FLAT being a notable outlier, in that it allowed the customization of the information being displayed.

	brat	doccano	flat	INCEption	MMAX2	Swan	WAT-SL	WebAnno	YEDDA
Single label	True	False	False	True	False	True	False	True	False
Multi label	True	False	False	True	False	True	False	True	False
Additional information on mouse-over (label)	True	False	False	True	False	True	False	True	False
Additional information on single click (label)	False	False	False	False	False	True	False	False	False
Additional information on double click (label)	True	False	False	True	False	False	False	True	False
Single highlight	True	True	True	True	True	False	True	True	True
Multi highlight	True	False	True	True	False	False	False	True	False
Highlights independent from label boxes	False	True	True	False	True	False	True	False	True
Additional information on mouse-over (highlight)	True	False	True	True	False	False	True	True	False
Additional information on single click (highlight)	False	False	True	False	True	False	True	False	False
Additional information on double click (highlight)	True	False	False	True	False	False	False	True	False
Overlapping highlights of differing lengths	True	False	True	True	False	False	False	True	False
Relations	True	False	True	True	True	True	False	True	False
Additional information on mouse-over (relation)	True	False	nan	True	False	False	False	True	False
Additional information on single click (relation)	False	False	nan	False	True	True	False	False	False
Additional information on double click (relation)	True	False	nan	True	False	False	False	True	False

Figure 3.3: Overview of support for additional information display after mouse interactions in tested software

	brat	doccano	flat	INCEption	MMAX2	Swan	WAT-SL	WebAnno	YEDDA
Information communicated on mouse-over	nan	nan	nan	nan	nan	nan	nan	nan	nan
ID	True	False	False	True	False	False	True	True	False
Content	True	False	True	True	False	True	False	True	False
Type	True	False	True	True	False	True	False	True	False
Error message	True	False	nan	True	False	False	False	True	False
Customizable	False	False	True	False	False	False	False	False	False

Figure 3.4: Overview of type of information displayed in tooltips of tested software

Chapter 4

Software patterns

The principle of design patterns in software development is a well-known one. We examined various types of design patterns to determine their usefulness for the development of annotation software. We tested various annotation tools as described in chapter 3 and gathered various complaints about and known issues in annotation software (see section A.3) to identify the appropriate level of abstraction of design patterns for this thesis. The stated goal of this thesis is to compile the requirements to annotation software for the purpose of being usable to fulfill natural-language annotation tasks.

It was determined that User Interface patterns (UI patterns for short) and User Experience patterns (UX patterns for short) tended to be too specific to be useful in the development of these requirements, generally describing specific forms of an interface (e.g. navigation tabs) rather than describing interactions with the software or the structure of underlying data. We concluded that Software Design patterns (SD patterns for short) and Interaction Design patterns (ID patterns for short) would be better suited for these purposes. UI patterns and UX patterns can be described as a Interaction Design-adjacent, as they tend to be motivated by Human-Computer Interaction principles and problems.

By describing observed patterns in the more abstract ID pattern format, we ensure that existing UI and UX patterns can be used as solutions for the formulated problems.

For certain core annotation tasks however, the ID and SD pattern formats ended up being too abstract and thus sub-optimal for attempting to document requirements of the given field in detail. We propose a new pattern format for these annotation tasks, which we describe in detail in chapter 5.

4.1 Interaction Design patterns

We contribute a list of patterns for annotation adjacent tasks in the ID pattern format. Patterns for core annotation tasks were instead formulated In the present study a modification of the ID pattern definition by Folmer is used. An overview can be seen in table 4.1.

Pattern name This section wasn't included in the original definition, but was implied.

Author We excluded this part of the pattern format, as all patterns were written by the same author.

Problem "Problems are related to the usage of the system and are relevant to the user or any other stakeholder that is interested in usability."

Use when "a situation (in terms of the tasks, the users and the context of use) giving rise to a usability problem. This section extends the plain problem-solutions dichotomy by describing situations in which the problems occur. "

Principle We excluded this part of the pattern format. Not only was the definition confusing, but we could also not find an example of any other ID pattern using this particular section of the format.

Solution "a proven solution to the problem. A solution describes only the core of the problem, and the designer has the freedom to implement it in many ways. Other patterns may be needed to solve sub problems. "

Why "How and why the pattern actually works, including an analysis of how it may affect certain attributes of usability. The rationale (why) should provide a reasonable argument for the specified impact on usability when the pattern is applied. The why should describe which usability aspects should have been improved or which other aspects might suffer. "

Examples Each example shows how the pattern has been successfully applied in a real life system. This is often accompanied by a screenshot and a short description.

Implementation We excluded this part of the pattern format, as this section turned out to be mostly redundant with the "Examples" section.

Table 4.1: Structure of the Interaction Design pattern format

We decided to forgo redundant formulations in the "Use when" section of the pattern, as every pattern would include a variation of "the objective is designing an application to annotate textual data in documents, wherein users [...]".

4.1.1 Import & Export

Import file formats

Problem: users want to import documents in a certain data format.

Use when: the application usually has a default data format or preferred way to handle the data behind the scenes, which differs from the file format of the documents the user intends to import. The users typically prefer a flexible system that allows for a variety of common formats to be imported.

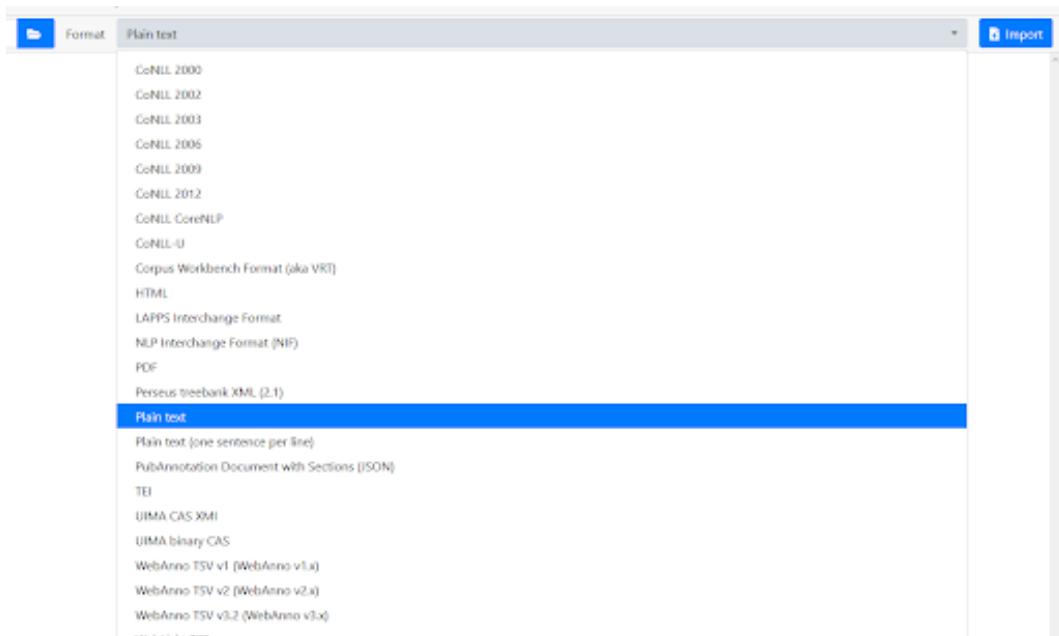
Solution: support for different import formats. The software contains support for the most common (and preferably also for some uncommon) file formats and automatically transfers the data from these file formats into that of the application-internal data storage.

Import options: certain formats are not standardized and allow for varying interpretations about how the import should be done (e.g. .txt files that separate text spans either by newline, or tabs or other separator characters such as commas). To ensure that the file is correctly imported, a menu offering different options on how to interpret the file can be shown to the user.

Conversion tool for different formats to a default import format: the software only contains support for one standardized import file format, but comes bundled with a different application that allows the conversion of various file formats into this standardized import file format.

Why: removing the need for users to worry about file formats shortens the workflow for annotating documents by one step. It makes the application more accessible to users who are not as familiar with the differences between file formats, and allows for easier integration of one annotation tool into a larger annotation pipeline.

Examples: INCEpTION allows users to choose from a variety of different file formats when importing documents.



Export file formats

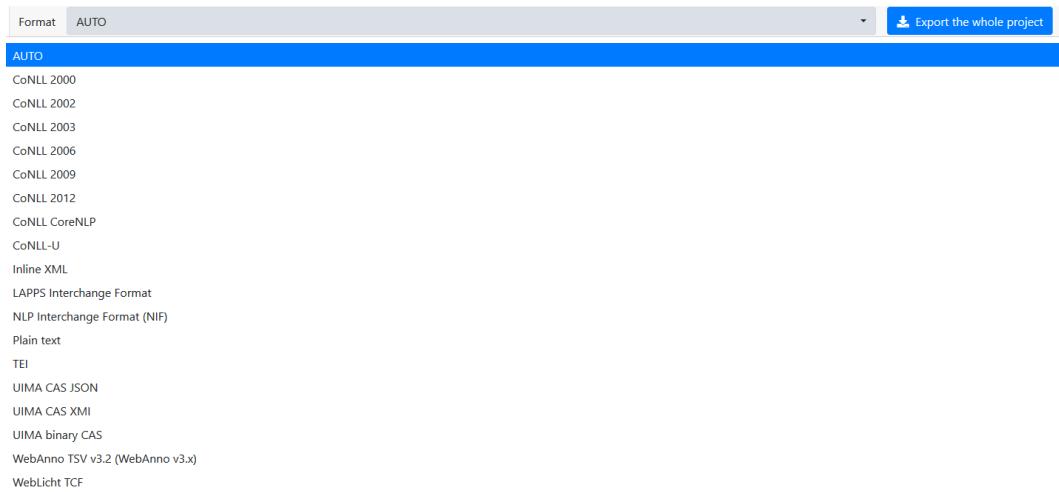
Problem: users want to export annotations in a certain data format.

Use when: the application usually has a default data format or preferred way to handle the data behind the scenes, which not always is the preferred data format the user wants to export their annotation data in. The users typically prefer a flexible system that allows for a variety of common formats to be exported.

Solution: support for different export formats. The software contains support for the most common file formats and allows the application-internal data storage to be exported in any of these formats.

Why: removing the need for users to worry about file formats shortens the workflow for annotating documents by one step. It makes the application more accessible to users who are not as familiar with the differences between file formats, and allows for easier integration of an annotation tool into a larger annotation pipeline.

Examples: INCEpTION allows users to choose from a variety of different file formats when exporting documents.



Saving of an incomplete document

Problem: users want to stop annotating and resume at a later date.

Use when: the document length may sometimes be too large to annotate comprehensively in one single sitting, or different users may want to share incomplete or work-in-progress annotations. The users will prefer to have an option that allows them to resume work on incomplete annotations instead of having to start over from scratch.

Solution: support for annotation data import. The software contains support for import formats that do not contain just the source documents, but also allow the user to import annotations.

Why: removing the need for users to worry about completing annotations in one go makes it easier to annotate large documents. It is also more in line with expectations users have from using other software (e.g. familiarity with office software). It makes the annotation process more resilient to data loss facilitates integration of annotation software into a larger annotation pipeline.

Examples: PDF Annotator (GRAHL software design (nd)) allows users not only to import annotations, but also to transfer them from one document to another.

Exporting and Importing Annotations

With the export and import features, you can transfer annotations from one document to another, for example from an older version to a revised version of the same document. Annotations will be saved in files with the .paa file extension.

Partial saving

Problem: users want to only save parts of a document, or not repeat the saving process for annotations they already saved earlier.

Use when: the document length may sometimes be so large that the saving process takes a considerable amount of time. The users will prefer an option allowing them to only save those parts of the document in which changes occurred - this e.g. in order to reduce load on the workstation.

Solution: the software contains support for partial saving. In partial saving, the user designates a portion of the document which is supposed to be saved, and only annotations for this range of the document are updated in the saved annotation file.

Why: giving the users more flexibility in how they want to save their documents allows expert users to prevent system slowdowns and to become more efficient in their workflow.

Examples: Stata (not an annotation software, LLC (2021)) contains support for partial saving.

Background saving

Problem: users want to save large annotation projects without having to stop their ongoing annotation.

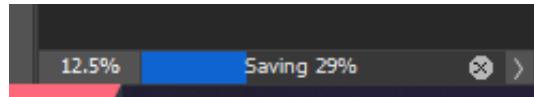
Use when: the document length may sometimes be so large that the saving process takes a considerable amount of time. The users will prefer an option that allows them to save the document and/or its annotations without having to wait for the saving process to complete before resuming their annotation work.

Solution: the software contains support for background saving. In background saving, the software keeps a copy of the data in storage at any given time, and upon the save-command starts to export the data, without blocking the access of the user to the other tools of the annotation software.

Why: removing the need for users to wait for the annotation software to finish its export shortens the time needed for annotating documents. It makes the annotation workflow more resilient to data loss by automatic and

frequent saving. It also facilitates integrating the application into an annotation pipeline working on real time data.

Examples: Adobe Photoshop (not an annotation software, Inc. (nd)) allows users to save files while continuing to work on them.



Automatic saving

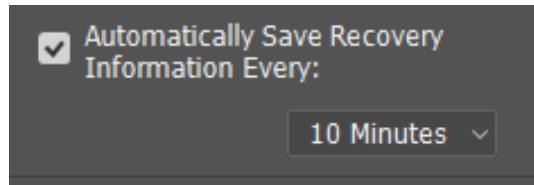
Problem: users seek to regularly save their annotation progress.

Use when: the users would prefer an option that allows them to schedule saving operations to occur in regular intervals.

Solution: the software contains support for automatic saving. The user can select a time interval after which the current and ongoing work is automatically saved.

Why: adding the option for automatic saving renders the annotation workflow more resilient to data loss.

Examples: Adobe Photoshop (not an annotation software, Inc. (nd)) allows users to set a save interval.



Export visualization as image

Problem: users want to export the visualization of data within the annotation software.

Use when: users can visualize certain aspects of the annotation, such as

- relations
- structures
- common properties

- overviews

The users may prefer an option that allows them to export these visualizations.

Solution: built-in screenshot support – The software helps with the creation of screenshots by including options such as auto-scrolling + stitching of screenshots, disabling of interface elements, etc.

Visualization rendering – the software creates a vector-based representation of the document and its annotations and provides options to either export the vector image and/or rasterize it to common image formats.

Why: adding the option to export visualizations makes it easier for annotators to present the results of their annotation projects. It also saves time if the alternative would be to recreate visualizations in third party software, or to use third party screenshot tools to take screenshots of the annotation tool's built-in visualizations.

Examples: brat (brat contributors (ndc)) offers the option to export visualization as .svg or .png file.

4.1.2 Automation

Preprocessing

Problem: users want to preprocess documents in a certain way to fit their annotation needs.

Use when: the users would prefer an option that allows them to automatically preprocess documents.

Solution: the software contains support for various common preprocessing operations, such as:

- Segmenting
- Stopword removal
- Tokenization
- Lowercasing
- Stemming
- Lemmatization
- Normalization

- Sanitisation/noise removal
- Text enrichment (adding information not previously in the text, e.g. part-of-speech-tagging)

Alternatively, or in addition, the software contains support for custom scripts.

Why: allowing the user to preprocess their documents increases the quality of the resulting dataset and speeds up the annotation process. It also facilitates integrating the application into an annotation pipeline.

Interactive learning of making annotations

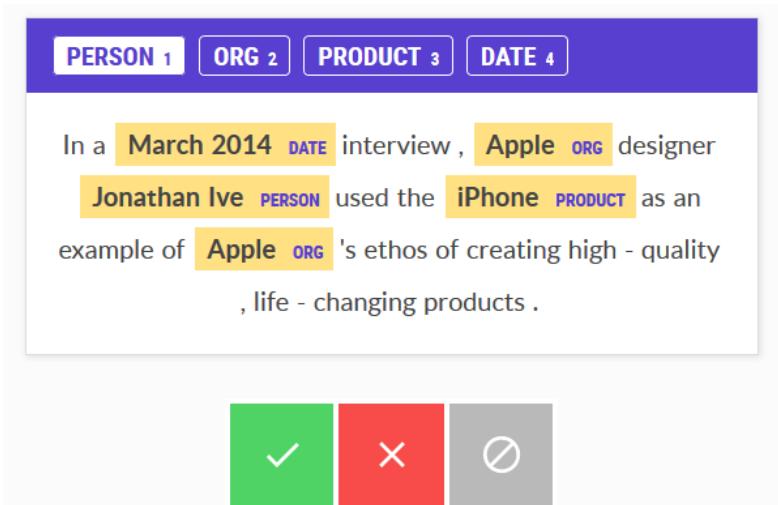
Problem: users want the annotation software to automatically suggest annotations.

Use when: the users would prefer an option that offers suggestions for annotations.

Solution: the software contains support for active learning systems that analyze existing annotations and offer suggestions based on them.

Why: enabling the user to select suggestions for annotations can speed up the annotation workflow, and it can make the annotation process simpler. Integrating active learning systems into the annotation software allows annotators to accomplish two tasks at the same time: annotation as well as machine learning model training.

Examples: Prodigy (Montani and Honnibal (2018)) offers interactive learning support.



Scripts

Problem: users want to automate certain annotation tasks.

Use when: the users would prefer the option to run custom scripts, these scripts automating functionality.

Solution: the software contains support for scripts that allow the user to interface with the data within the documents, with the annotations, and with the software itself, as well, via a simple API.

Why: allowing the user to customize the software and expand its functionality via scripts can speed up the annotation process.

Examples: WebAnno supports customizable automation (The WebAnno Team (nd)).

4.1.3 Source document management

Editing the source document

Problem: users want to directly edit the source document in an annotation project instead of just adding an annotation on top.

Use when: the document possibly contains simple mistakes (such as spelling errors, typos, etc.) the user may want to correct instead of just marking as errors, and would therefore prefer an option that allows for directly editing the source document. This would avoid having to do editing manually outside the annotation software and avoid re-importing the document and breaking any already existing annotations.

Solution: the software contains support for editing the source document and updating annotations in standoff format. Updating the document automatically recalculates the character offsets in standoff annotation file formats to ensure that the annotations remain valid and correctly positioned after editing the source document. If an automatic recalculation is not possible because it would introduce ambiguity, the software shows a menu with options for the user to manually realign the annotations.

Why: removing the need for users to manually edit the source documents outside the annotation software (in case changes are needed) increases the speed of the annotation workflow. It renders the annotation workflow less

prone to errors by ensuring that the annotations and document remain valid after applying the changes. It makes it easier to integrate the application into an annotation pipeline working on real time data.

Importing large amount of small documents

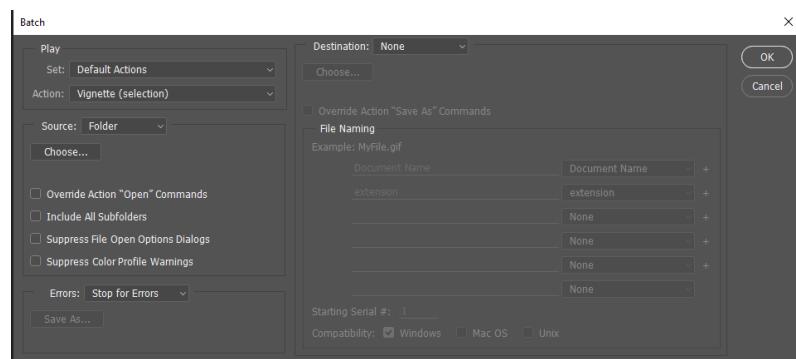
Problem: users desire to import a large amount of individual smaller documents in sequence.

Use when: the documents are possibly of very short length, which may result in the mere importing of documents taking up a significant amount of the total annotation time. The users will prefer an option allowing for importing several documents in one single process step.

Solution: the software contains support for batch importing. The user can specify import options in a menu – and then import multiple documents at once using the same settings.

Why: removing the need for users to manually import every single document of many speeds up the annotation workflow considerably.

Examples: Adobe Photoshop (not an annotation software, Inc. (nd)) allows for batch processing of files.



Switching to next document within source

Problem: users may want to edit several documents in sequence, without importing them all at once.

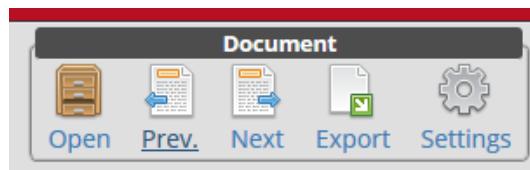
Use when: the documents may possibly be of very short length, resulting in the mere importing of documents taking up a significant amount of the total annotation time. The users will prefer an option allowing for

quickly switching to the next document in line, without having to use the import options every time.

Solution: the software contains support for designating a source of documents (e.g. a folder) and an option that allows for opening the next document in line using the same import options as the current document.

Why: removing the need for users to manually import every single document speeds up the annotation workflow considerably.

Examples: WebAnno allows for fast switching to the next or previous document via dedicated buttons.



4.1.4 Annotation level interactions

Customizing visualization

Problem: users seek to visualize annotated data in a certain way.

Use when: the users will prefer to have an option to customize visualizations to suit their needs.

Solution: the software contains support for customized visualization options, which allow the user to change how different types of markables are displayed.

Why: allowing the user to customize the visualization of annotations makes it easier to explore datasets using the software. It speeds up the annotation process when reviewing existing annotations for errors.

Examples: brat has rudimentary support for the customization of annotation visuals (brat contributors (nda)).

Changing visualization focus

Problem: users want to be able to focus on parts of their annotations, while masking out others.

Use when: the users will prefer an option that allows for focusing on and thus visualizing only a subsection of all annotations.

Solution: the software contains a filter option that either highlights annotations matching the filter criteria, or disables or tones down the visualization of annotations that do not match the filter criteria, or includes a combination of both.

Why: allowing the user to focus on only a subsection of all annotations allows for an easier exploration of annotation datasets by reducing unnecessary visual clutter.

Examples: FLAT support multiple views of the same annotation project.



Summaries/Overages/Basic stats

Problem: users prefer to be able to see meta-information about their annotations.

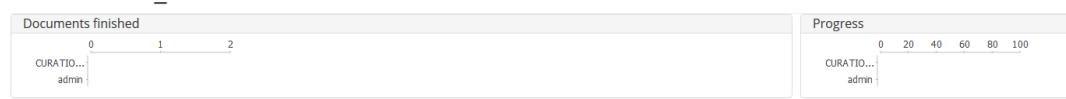
Use when: the users would prefer an option allowing for an overview regarding all the annotations.

Solution: the software contains support for collecting statistics about the annotations in a document or project, and also visualization options that enable displaying these statistics in a way supporting readability for humans.

Why: allowing the user to see statistics about the annotations in a document facilitates insights into the annotation task which otherwise would not be possible.

Examples: WebAnno offers support for project monitoring.

annotation_test



4.1.5 Quality assurance

Computing inter-/intra-annotator agreement

Problem: users seek to get an overview regarding the similarity of results from several annotators.

Use when: in an application where multiple users can annotate their own copies of the documents, the users would prefer an option to determine similarity between the annotations.

Solution: the software contains a support for the calculation of various metrics which indicate agreement, such as

- Cohen's Kappa
- Fleiss' Kappa
- Scott's Pi
- Krippendorff's Alpha

Why: enabling the user to calculate the agreement between different annotations on the same document is helpful when trying to construct a gold standard corpus, and to find faults in the individual annotations. It will result in a higher quality dataset and speeds up the process of finding consensus on the correct annotation.

Examples: GATE offers the option to calculate IAA (The University of Sheffield (nd)).

WebAnno includes an entire agreement section:



Comparing results from multiple annotators

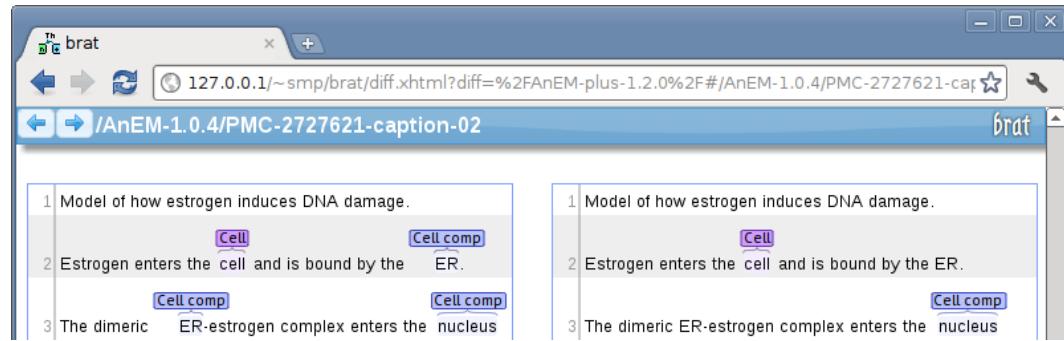
Problem: users want to compare the results of several annotators.

Use when: in an application where multiple users can annotate their own copies of the documents, the users will prefer an option to compare the results of different annotators with one another.

Solution: the software contains a visualization option in which multiple annotation files for the same document are compared and differences between the annotations are highlighted.

Why: allowing the user to compare different annotations regarding the same document is helpful when trying to construct a gold standard, and to find faults in the individual annotations. It will result in a higher quality dataset and speeds up the process of finding consensus on the correct annotation.

Examples: brat supports side-by-side view of annotated documents:



(source: official brat documentation brat contributors (ndb))

Error detection

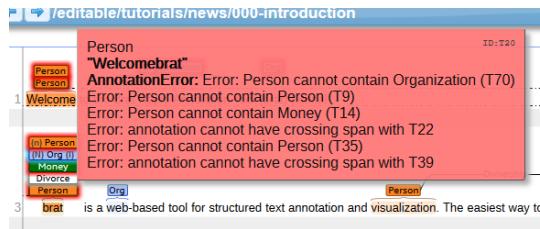
Problem: users want to be able to spot mistakes in their annotations to fix them.

Use when: the users will prefer to have an option that warns them if their annotation contains mistakes, such as not meeting certain requirements.

Solution: the software contains an annotation scheme editor, in which the user can specify requirements for certain types of annotation. The software also contains visualization options notifying the user if annotations of a given type do not meet the requirements specified in the scheme.

Why: allowing the user to compare different annotations regarding the same document is helpful when trying to construct a gold standard, and to find faults in the individual annotations. It will result in a higher quality dataset and speeds up the process of finding consensus on the correct annotation.

Examples: brat includes support for error messages:



4.1.6 Mode of interaction/modality

Annotating via keyboard

Problem: users want to annotate with a minimal amount of mouse clicks slowing them down.

Use when: the users would prefer to have an option to complete the annotation process without the need to select markables or data fields using the mouse.

Solution: the software supports the use of keyboard shortcuts that fulfill the same functionality as usually achieved using the mouse. Such shortcuts could include (but are not limited to):

- Selecting the next/previous markable
- Selecting the next/previous existing annotation
- Selecting the next/previous data field
- Creating new annotations
- Deleting existing annotations
- Switching to the next document

Why: allowing the user to use keyboard shortcuts instead of the mouse speeds up the annotation process. It makes the software more accessible, since people with certain motor disabilities are incapable of using a mouse, but have alternative means of entering text (e.g. via speech recognition).

Examples: Yedda is usable entirely via keyboard.



(source: Yedda documentation Yang et al. (2018))

Automating annotations

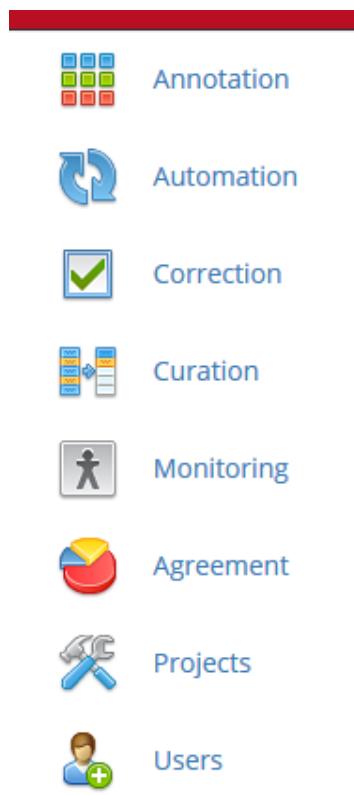
Problem: users seek to automate annotations.

Use when: the users would prefer an option to automate (parts of) the annotation process.

Solution: the software supports the use of macros and/or rulesets which automate repetitive tasks.

Why: enabling the user to automate parts of the annotation workflow speeds up the annotation process and allows the annotator to focus on complex annotation tasks which aren't possible without human intervention. It facilitates integrating the annotation software into a larger annotation pipeline.

Examples: WebAnno offers an automation category.



Annotating via mouse/pointing device

Problem: users want to annotate documents using a mouse/pointing device.

Use when: the users would prefer an option to complete (parts of) the annotation using a mouse or other pointing device.

Solution: the software contains menus which can be operated using a mouse/pointing device.

Why: allowing the user to use a pointing device to complete (parts of) the annotation renders the interaction with the annotation software more intuitive compared to a keyboard-only approach.

Examples: (Webanno)

Annotating via touchscreen

Problem: users want to annotate documents using a touchscreen.

Use when: the users would prefer an option to use the annotation software on a device using touchscreen.

Solution: A: touchscreen annotation via a native application. The software is capable of running on touchscreen devices such as smartphones or tablets. The software is designed to make use of this input option.

B: Touchscreen annotation via a web-based application. The software is split into two sections, a server part and a client part, with the software only required to be installed on the server-side, but being usable by the client via any browser. The interface of the web-application avoids interactions which cannot be used on touch-based (or are difficult to be replicated on them), such as double-clicking, right clicking, etc., or features alternative interactions.

Why: enabling the user to use a touchscreen to complete (parts of) the annotation renders the annotation process more intuitive. This is important to consider for crowd-sourcing annotations, as the number of smartphones and tablets has long surpassed the number of desktop PCs.

Examples: Crowdsource app for Android
docco being served via browser.

4.2 Software Design patterns

In the introductory section of chapter 4 we concluded that ID and SD patterns are useful pattern formats in the development of requirements for annotation software, but with the caveat that the formats were too abstract for the development of patterns about certain core annotation tasks. We deemed the SD pattern format too abstract, because the proposed patterns are based on observations in the area of natural-language annotation tasks, and therefore validity of the proposed patterns outside this context was not considered. SD patterns, which are generally applicable in all areas of software design and are independent from domain specific considerations, would therefore be too restricting for the purposes of this thesis. Abstraction attempts of requirements exclusive to natural-language annotation tasks would cause issues, either by abstracting away the domain-specific requirements, or by introducing these requirements into patterns that appear to be generally applicable, even though these requirements might not exist outside this field.

However, SD patterns are a well established concept in the field of software design, and the structure of these patterns can inform the formulation of a pattern structure more suitable for the needs of this thesis. In table 4.2 we

briefly introduce *only* the sections of the Software Design pattern format by Gamma et al. (1994) which influenced the Annotation pattern format proposed in chapter 5.

Pattern Name and Classification	“The pattern’s name conveys the essence of the pattern succinctly. A good name is vital, because it will become part of your design vocabulary. [...]”
Intent	“A short statement that answers the following questions: What does the design pattern do? What is its rationale and intent? What particular design issue or problem does it address?”
Motivation	“A scenario that illustrates a design problem and how the class and object structures in the pattern solve the problem. The scenario will help you understand the more abstract description of the pattern that follows.”
Consequences	“How does the pattern support its objectives? What are the trade-offs and results of using the pattern? What aspect of system structure does it let you vary independently?”
Implementation	“What pitfalls, hints, or techniques should you be aware of when implementing the pattern? Are there language-specific issues?”
Related Patterns	“What design patterns are closely related to this one? What are the important differences? With which other patterns should this one be used?”

Table 4.2: Structure of the Software Design pattern format by Gamma et al. (1994)

Chapter 5

Annotation patterns

5.1 Proposal for new pattern structure

In this thesis we want to formalise annotation patterns. We observe similarities to software patterns and find that both the ID pattern format and the SD pattern format contain attributes useful or necessary for the description of patterns about annotation tasks. However, these pattern formats are individually ill suited to describe the observed patterns in the core annotation tasks. We propose a new Annotation Pattern format (AP format for short) for this purpose, adapting aspects and terminology of both the ID pattern format and SD pattern format. This renders the new format easily understandable for anyone already familiar with one or both of the previously mentioned formats. The new format also facilitates comparison between the formats, encouraging the adaption of existing patterns into this domain.

The AP format as depicted in section 5.1 offers all the attributes necessary to describe an annotation pattern in its entirety.

Pattern name: a descriptive and unique name that is conducive to identifying and referring to the pattern.

Intent: a description of the goal behind the pattern and the reason for using it.

Use when: a scenario consisting of a problem and a context in which this pattern can be used.

Solution: a proven solution to the problem. This solution describes only the core of the problem, and the designer has the freedom to implement it in many ways. Other patterns may be needed to solve sub problems.

Consequences (optional): a description of the results, side effects, and trade offs caused by using the pattern, if they are known.

Related patterns (optional): a description of, if known, how the pattern interacts with other annotation patterns and discussions of known differences to similar patterns.

Examples: a list of examples illustrating how the pattern has been successfully applied in a real life system. In many examples a screenshot and a short description is included.

Table 5.1: Structure of the Interaction Design pattern format

During the development of the AP format, we compared the terminology and descriptions of Folmer (nd); Nguyen (nd) and Gamma et al. (1994) to determine useful sections for the proposed pattern. We chose to adapt existing terminology and descriptions whenever possible, and decided to use the most descriptive and easily understood terms if similar sections existed in the different pattern formats.

We contribute a list of patterns for annotation tasks in the newly proposed Annotation Pattern format.

5.2 Annotation patterns

We decided to forgo redundant formulations in the "Use when" section of the pattern, as every pattern would include a variation of "the objective is designing an application to annotate textual data in documents, wherein users [...]".

5.2.1 Selecting markables

Intent:

when creating annotations, it is necessary to specify which part of the document the user is trying to annotate.

Use when:

the users need to be enabled to specify which part of the document they want to annotate to complete this task.

Solution:

the software contains support for selecting subsections of a document. Interacting with the markables of a document by highlighting parts of the text visually indicates which part of the text has been selected. This is done via clicking and dragging the mouse cursor. A representation of the selected markables is temporarily stored in memory, e.g. by storing start and end indexes of the selection.

Consequences:

since the pattern requires interaction with a markable to create a selection, it becomes an issue if such an interaction is not possible or non-trivial, e.g. interacting with zero-width-characters.

Since clicking and dragging the mouse cursor does not necessarily follow the reading direction of the text in a document, the start index (where the mouse was pressed) can occur positioned after the end index (where the mouse was released). In the implementation, start and end indexes may require interchanging and, if the direction of the selection matters, the direction is to be stored separately.

Related patterns:

this pattern is often used in combination with the "Creating labels" pattern (see 5.2.2).

Examples:



(WebAnno)

5.2.2 Creating labels

Intent:

creating labels is the most basic functionality of any annotation software. Users will want to annotate a selected markable.

Use when:

users need to be able to create annotations.

Solution:

the software contains support for creating annotations. Interacting with a selection opens a menu to create an annotation for the markables within the selection, or enables the use of a designated button or shortcut for creating annotations.

Consequences:

since the pattern requires a selection to create annotations, there is a problem if a user wants to create an annotation prior to selecting the markables the annotation is intended for. An example would be automated recurring annotations in a live-updating feed, where selection at a given point in time is not possible since the markables do not exist yet. A possible solution might be to allow the creation of annotations without any selection, this requiring, however, an assignment of the annotation to a selection for the annotation to become valid.

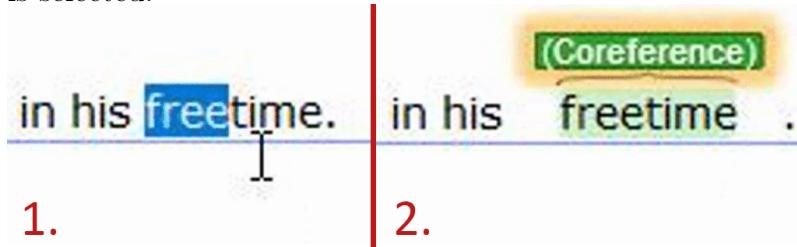
Related patterns:

- This pattern requires the "Selecting markables" pattern (see 5.2.1).

- This pattern is the counter-pattern to the "Deleting annotations" pattern (see 5.2.5).
- This pattern is often used in combination with the "Editing annotations" pattern (see 5.2.4).

Examples:

in WebAnno a label is created upon releasing the left mouse-click while text is selected.



5.2.3 Selecting markables via label

Intent:

at times the user needs to interact with existing annotated markables. For this purpose the ability to select labels is necessary.

Use when:

the users need the ability to select existing annotations of the document as a prerequisite for other actions (see the related patterns section for details).

Solution:

the software contains support for selecting labels.

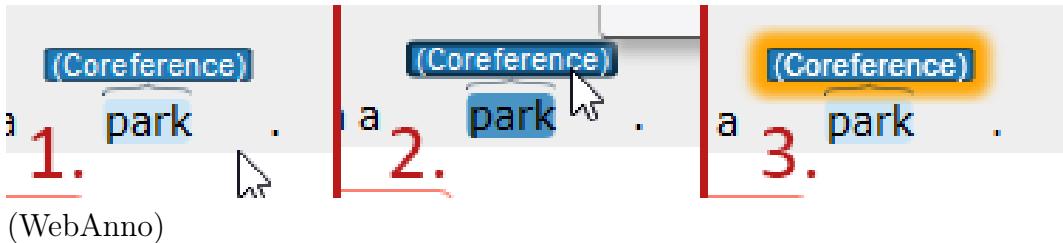
Interacting with a label of an annotated markable by clicking or double-clicking on the label selects the annotation visually represented by the label.

It is suggested to indicate an active selection by visually distinguishing the label of a selected annotation from the label of an annotation that is not currently selected.

Related patterns:

- This pattern requires the "Creating labels" pattern (see 5.2.2).

- This pattern is a prerequisite for the "Editing annotations" pattern (see 5.2.4).

Examples:

5.2.4 Editing annotations

Intent:

the user may need to modify an existing annotation.

Use when:

the users will prefer an option to update and change the properties of markables they have already annotated.

Solution:

the software contains support for editing existing annotations.

Selecting an existing annotation opens a menu (or switches the contents of a form showing the properties of the currently selected element) that allows the user to modify properties of the selected annotation.

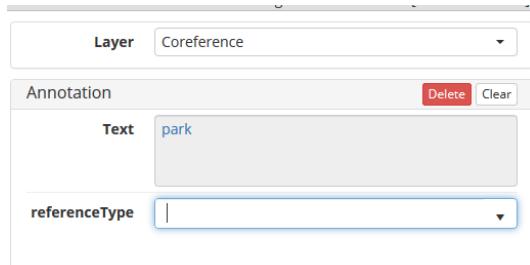
Consequences:

certain types of relations can be restricted to be valid only between annotations that fulfill certain properties (e.g. annotating a certain phenomena). If a relation between two annotations was created, and the annotation is edited afterwards, relations between this annotation and others can potentially become invalid. It is suggested that by confirming any changes to the properties of an annotation, any relations or other structures which are dependent on the annotation receiving an alert regarding the change. This way such relations or other structures can react to the change (e.g. by triggering a validity check, alerting the user to possible changes of these relations, etc.)

Related patterns:

- This pattern requires the "Selecting annotations via label" pattern (see 5.2.3).

Examples:



(WebAnno)

5.2.5 Deleting annotations

Intent:

the user may need to delete an existing annotation.

Use when:

the users require the ability to delete existing annotations.

Solution:

the software contains support for deleting existing annotations.

Selecting an existing annotation opens a menu (or switches the contents of a form showing the properties of the currently selected element), allowing the user to delete the selected annotation.

Consequences:

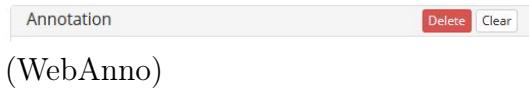
relations are only valid if the annotations they are referring to exist.

If a relation between two annotations was created, and one of the two annotations is deleted afterwards, relations between this annotation and others become invalid. It is suggested that by confirming the deletion of an annotation, any relations or other structures which are dependent on the annotation receive an alert regarding the change. This way these relations or other structures can react to the change (e.g. by triggering a validity check, alerting the user to possible changes of these relations, etc.)

Related patterns:

- This pattern requires the "Selecting annotations via label" pattern (see 5.2.3).

Examples:



5.2.6 Creating directed relations between markables

Intent:

the user may want to annotate the relation between annotated markables of a document, wherein the direction of the relation is of importance.

Use when:

the users require the ability to annotate which annotated markables of the document are related to one another and the ability to differentiate between relations in one direction or the other.

Solution:

the software contains support for creating relations between annotated markables.

Interacting with one annotated markables of a document by clicking and holding on the associated label indicates that this annotated markable was selected as the source of the relation. Then moving the cursor while still holding to another label indicates that a relation between the two annotated markables is being created. Releasing the click over the target label indicates that the annotated markable associated with this label was selected as the target of the relation.

It is suggested to indicate a directional relation between two annotated markables. This is done by displaying a link with an arrowhead between their associated labels, with the link originating at the source label and pointing at the target label. While creating the relation, the arrowhead should point at and follow the cursor until the click is released over the target label.

Consequences:

since the pattern requires two labels to visualize a relation, modification or deletion of one of the labels becomes an issue. In case a label is modified and its visual representation changes, the link between the labels needs to be updated to ensure the arrowhead points to the correct label.

In case a label is modified by changing the properties of the underlying markable, the relation could potentially become invalid. This may for example be the case if relations are only allowed between certain types of annotation and the type of an annotation changed. Invalid relations might need to be deleted, brought to the annotators attention or temporarily disabled.

- For details regarding potential problems consult the "Editing annotations" pattern (see 5.2.4).

In case a label is deleted, the relation becomes invalid.

- For details regarding potential problem consult the "Deleting annotations" pattern (see 5.2.5).

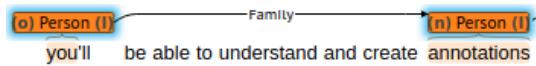
Since labels may be very far apart in the document, it is important to display links between labels in a manner that does not negatively impact the annotation experience.

- A negative example can be found inside section A.3.

There is no upper limit to how many relations one annotated markable can have with other annotated markables. It therefore is important to display links between labels in a manner that does not negatively impact the annotation experience. Such an impact may happen in case of a large amount of relations originating from one label, or several labels in close proximity.

Related patterns:

- This pattern requires the "Selecting annotations via label" pattern (see 5.2.3).
- This pattern is parallel to the "Creating undirected relations" pattern (see 5.2.7).
- This pattern is the counter-pattern to the "Deleting relations" pattern (see 5.2.10).
- This pattern is often used in combination with the "Editing relations" pattern (see 5.2.9).

Examples:

(brat)

5.2.7 Creating undirected relations between markables

Intent:

the user may want to annotate the relation between annotated markables of a document, wherein the direction of the relation is not of importance.

Use when:

the users need to be enabled to annotate which annotated markables of the document are related to one another.

Solution:

the software contains support for creating relations between annotated markables. Interacting with one annotated markables of a document by clicking and holding on the associated label indicates that this annotated markable will be part of the resulting relation. Then, moving the cursor – while still holding – to another label indicates that a relation between the two annotated markables is being created. Releasing the click over the target label finalizes the creation of a relation between the two annotated markables.

It is suggested to indicate an undirected relation between two annotated markables. This is done by displaying a link between their associated labels. While creating the relation, the link should appear between the label that was first clicked and the cursor, until the click is released over the other label.

Consequences:

since the pattern requires two labels to visualize a relation, it becomes an issue if one of the labels is modified or deleted. In case a label is modified and its visual representation changes, the link between the labels needs to be updated to ensure the link is still connected to both labels.

In case a label is modified by changing the properties of the underlying markable, the relation could potentially become invalid, e.g. if relations are only allowed between certain types of annotation and the type of an annotation

changed. Invalid relations might need to be deleted, brought to the annotators attention or temporarily disabled.

- For details regarding source of potential problem consult the "Editing annotations" pattern (see 5.2.4).

In case a label is deleted, the relation becomes invalid.

- For details regarding source of potential problem consult the "Deleting annotations" pattern (see 5.2.5).

Since labels can be very far apart in the document, it is important to display links between labels in a manner that doesn't negatively impact the annotation experience.

- See negative example "brat link across several sentences".

Since there is no upper limit to how many relations one annotated markable can have with other annotated markables, it is important to display links between labels in a manner that doesn't negatively impact the annotation experience in case of a large amount of relations originating from one label, or several labels in close proximity.

Related patterns:

- This pattern requires the "Selecting annotations via label" pattern (see 5.2.3).
- This pattern is parallel to the "Creating directed relations" pattern (see 5.2.6).
- This pattern is the counter-pattern to the "Deleting relations" pattern (see 5.2.10).
- This pattern is often used in combination with the "Editing relations" pattern (see 5.2.9).

Examples:



(brat)

5.2.8 Selecting relations

Intent:

at times the user needs to interact with existing relations between annotated markables. For this purpose the ability to select existing relations is necessary.

Use when:

the users need the ability to select existing relations between annotated markables of the document as a prerequisite for other actions (see the related patterns section for details).

Solution:

the software contains support for selecting existing relations between annotated markables.

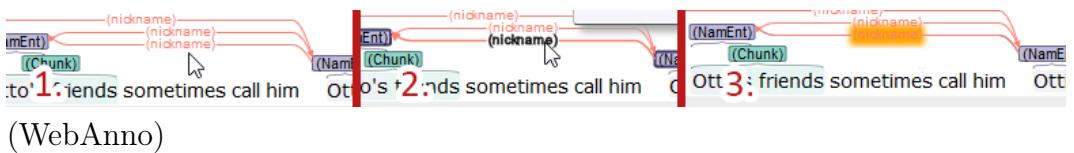
Interacting with a relation between annotated markables of a document by clicking or double-clicking on the link between the associated labels selects the relation visually represented by the link.

It is suggested to indicate an active selection by visually distinguishing the link of a selected relation from the link of a relation that isn't currently selected.

Related patterns:

- This pattern requires the "Creating directed relations between markables" pattern (see 5.2.6) and/or the "Creating undirected relations between markables" pattern (see 5.2.7).
- This pattern is a prerequisite for the "Editing relations" pattern (see 5.2.9).

Examples:



5.2.9 Editing relations

Intent:

at times the user needs to modify an existing relation between annotated markables of a document.

Use when:

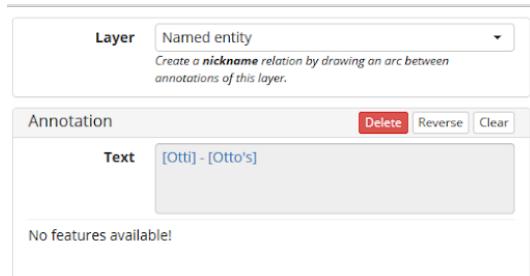
the users would prefer an option to modify existing relations between annotated markables of the document.

Solution:

the software contains support for editing existing relations between annotated markables. Selecting an existing relation between annotated markables opens a menu, or switches the contents of a form showing the properties of the currently selected element, which allows the user to modify properties of the selected relation.

Related patterns:

- This pattern requires the "Selecting relations" pattern (see 5.2.8).

Examples:

(WebAnno)

5.2.10 Deleting relations

Intent:

at times the user needs to delete an existing relation between annotated markables of a document.

Use when:

the users need the ability to delete existing relations between annotated markables of the document.

Solution:

the software contains support for deleting existing relations between annotated markables. Selecting an existing relation between annotated markables opens a menu, or switches the contents of a form showing the properties of the currently selected element, which allows the user to delete the selected relation by pressing a button.

It is suggested to allow the deletion of a selected relation via a keyboard shortcut.

Related patterns:

- This pattern requires the "Selecting relations" pattern (see 5.2.10).

Examples:



5.2.11 Reusing annotations

Intent:

at times the user wants to reuse certain annotations because of their repetitive nature.

Use when:

the users would prefer an option to reuse the properties of previous annotations.

Solution:

Via templates:

The software contains support for creating templates, which the user can fill with information they expect to be able to reuse several times. When interacting with existing annotations or creating new annotations, a menu option allows the user to apply the template to the currently selected markable.

Via copy & paste:

The software contains support for copying & pasting the contents of annotations.

Related patterns:

- This pattern requires the "Selecting annotations via label" pattern (see 5.2.3).

Examples:

Elan (Hellwig et al. (2021); Sloetjes and Wittenburg (2008)) support the copying and pasting of annotations.

Chapter 6

Conclusion and Future Work

In this thesis we proposed a new approach to designing annotation software based on patterns. This approach facilitates the development and improvement of annotation software by examining and guarding against common issues found in existing annotation tools. We defined eleven new Annotation patterns for common annotation tasks, and 23 Interaction Design patterns for common annotation-adjacent tasks within an annotation project, categorized by the different aspects of typical annotation projects. This thesis collected and documented a large number of linguistic phenomena and redefined various annotation units to fix issues caused by the previous definitions.

We selected eleven annotation tools for testing and produced a detailed documentation of the installation process for each of them. We created screenshots of their interfaces and documented the features of nine of the tools in extreme detail. Furthermore, we curated a list of known interface issues and edge cases. During the research phase of this thesis and attempts to formulate patterns based on various edge cases, we also noted that the existing landscape of annotation software is severely lacking in accessibility features. Only a single tool (Yedda) was controllable using only a keyboard, in all other annotation software we tested mouse-interactions were needed, and few tools had options to manipulate the visual representation of annotations in the software, e.g. increasing font size or changing colour schemes.

The patterns introduced in this thesis can act as foundation for future research into ways of making the field of NLP and computer assisted annotation more accessible. Using the proposed patterns as a starting point, existing accessibility UX patterns could be examined for compatibility with common annotation tasks, integrated where possible and expanded where insufficient for the needs of annotators.

Appendix A

Documentation

A.1 Documentation of annotation tasks in the tested software

A detailed documentation of basic annotation tasks in the tested annotation software is attached to the digital copy of this bachelor's thesis under the filename "A1_Appendix_Tested_Software.pdf"

A.2 Documentation of the installation process

A detailed documentation of the installation process for all tested annotation tools is attached to the digital copy of this bachelor's thesis under the filename "A2_Appendix_Installation_Documentation.pdf"

A.3 Documentation of known interface issues in the annotation software

A detailed documentation of known issues and edge-cases in regards to the interface of annotation software is attached to the digital copy of this bachelor's thesis under the filename "A3_Appendix_Interface_Issues.pdf"

Appendix B

List of phenomena

Below we provide a list of phenomena sorted by source:

1. (Ide and Pustejovsky, 2017)

- transcriptions (phonetic features, discourse structures)
- part-of-speech tags
- sense tags
- syntactic analyses
- named entity labels
- semantic role labels
- time and event identification
- co-reference chains
- discourse-level analyses
- translation equivalents
- opinion, sentiment & subjectivity
- factivity
- spatial phenomena
- metaphor
- textual entailment
- dialogue acts
- speech
- biomedical annotations

2. Li et al. (2019)

- Chinese character annotation

3. Pustejovsky and Stubbs (2012)

- syntax
- semantics
- morphology
- phonology
- phonetics
- lexicon
- discourse analysis
- pragmatics
- text structure analysis
- TreeBank tags
- Agent (these should be subitems):
 - Theme/figure
 - Experiencer
 - Source
 - Goal
 - Recipient
 - Patient
 - Instrument
 - Location/ground

4. Palmer and Xue (2012)

- syntactic structure (e.g. treebanking)
- independent semantic classification (e.g. sense tagging)
- semantic relation labeling (e.g. semantic role labeling)
- discourse relations
- temporal relations
- coreference tagging
- opinion tagging

5. Karoui et al. (2017)
 - irony (multilayered: activation types, categories, markers)
6. Bawden et al. (2018)
 - coreference
 - lexical cohesion
 - lexical disambiguation
7. Dipper et al. (2004a)
 - interlinear morphemic transcription
 - morphosyntactic annotation
 - syntactic annotation
 - semantic/pragmatic annotation
8. Bada et al. (2010)
 - part-of-speech tagging
 - tree-banking
 - noun + noun-phrase coreference
 - assertional annotations between concept annotations via relations
 - biomedical terminology
9. Rama et al. (2018)
 - entities (e.g. family, self, index, condition, event)
 - modifier entities (e.g. side, age, negation, amount, temporal)
 - relations
10. Kim and Klinger (2018)
 - emotion
 - entity (e.g. character, event, other)
 - relation (e.g. experiencer, target, cause, co-reference)
11. Zampieri et al. (2017)
 - complicated word identification

12. Bateman et al. (2002)

- rhetorical structure
- layout structure
- navigation structure

13. Hellwig et al. (2018)

- phonetic annotation
- morphologic annotation
- lexical annotation
- verb-argument annotation

14. Musi et al. (2018)

- taxonomic hierarchy of argument schemes
- relations (e.g. support, rebut)

15. Forbes et al. (2018)

- deeply nested event structures
- syntactic dependencies
- undirected relations
- co-reference resolution
- morphological parses

16. Letcher (2013)

- passive clause
- imperative clauses
- interrogative clauses
- complement clause
- relative clause

17. Bamman et al. (2019)

- entities (e.g. people, facilities, geo-political entities, locations, vehicles, organizations)
- events

- co-reference of identity
- copula
- apposition
- quotation (i.e. direct speech, speaker)

Bibliography

- A Survey of NLP Annotation Platforms, N. (2020). A survey of nlp annotation platforms. <https://github.com/alvations/annotate-questionnaire>. 3.1
- ACL Anthology, A. (2021). Acl anthology. <https://www.aclweb.org/anthology/>. 2.1
- Bada, M., Eckert, M., Palmer, M., and Hunter, L. (2010). An overview of the CRAFT concept annotation guidelines. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 207–211, Uppsala, Sweden. Association for Computational Linguistics. 8
- Bamman, D., Popat, S., and Shen, S. (2019). An annotated dataset of literary entities. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2138–2144, Minneapolis, Minnesota. Association for Computational Linguistics. 17
- Bateman, J., Henschel, R., and Delin, J. (2002). A brief introduction to the GeM annotation schema for complex document layout. In *COLING-02: The 2nd Workshop on NLP and XML (NLPXML-2002)*. 12
- Bawden, R., Sennrich, R., Birch, A., and Haddow, B. (2018). Evaluating discourse phenomena in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1304–1313, New Orleans, Louisiana. Association for Computational Linguistics. 6
- Bollmann, M., Petran, F., Dipper, S., and Krasselt, J. (2014). CorA: A web-based annotation tool for historical and other non-standard language data. In *Proceedings of the 8th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)*, pages 86–90, Gothenburg, Sweden. Association for Computational Linguistics. 3.2

BIBLIOGRAPHY

- brat contributors (n.d.a). Annotation comparisons. <https://brat.nlplab.org/configuration.html#visual-configuration>. 4.1.4
- brat contributors (n.d.b). Annotation comparisons. <http://brat.nlplab.org/new-in-v1.3.html#comparison>. 4.1.5
- brat contributors (n.d.c). brat standoff format. <https://brat.nlplab.org/standoff.html>. 3.2.2, 4.1.1
- Burghardt, M. (2012). Usability recommendations for annotation tools. In *6th Linguistic Annotation Workshop - Proceedings of the ACL 2012*, pages 104–112. Association for Computational Linguistics, Jeju. 2.1
- Coviello, E., Chan, A. B., and Lanckriet, G. (2011). Time series models for semantic music annotation. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(5):1343–1359. 2.1
- Dipper, S., Götze, M., and Skopeteas, S. (2004a). Towards user-adaptive annotation guidelines. In *Proceedings of the 5th International Workshop on Linguistically Interpreted Corpora*, pages 23–30, Geneva, Switzerland. COLING. 7
- Dipper, S., Götze, M., and Stede, M. (2004b). Simple annotation tools for complex annotation tasks: an evaluation. 2.2.2, 2.2.2, 2.2.2, 2.2.2
- Eckart de Castilho, R., Mújdríca-Maydt, É., Yimam, S. M., Hartmann, S., Gurevych, I., Frank, A., and Biemann, C. (2016). A web-based tool for the integrated annotation of semantic and syntactic structures. In *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*, pages 76–84, Osaka, Japan. The COLING 2016 Organizing Committee. 3.2
- Federmann, C. (2012). Appraise: An open-source toolkit for manual evaluation of machine translation output. *The Prague Bulletin of Mathematical Linguistics*, 98:25–35. 3.2
- Folmer, E. (n.d.). The glossary of human computer interaction. <https://www.interaction-design.org/literature/book/the-glossary-of-human-computer-interaction/interaction-design-patterns>. 2.1, 4.1, 5.1
- Forbes, A., Lee, K., Hahn-Powell, G., Valenzuela-Escárcega, M. A., and Surdeanu, M. (2018). Text annotation graphs: Annotating complex natural

BIBLIOGRAPHY

- language phenomena. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA). 15
- Gamma, E., Helm, R., Johnson, R., and Vlissides, J. M. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1 edition. 2.1, 4.2, 4.2, 5.1
- GRAHL software design, G. (n.d.). Automation. <https://www.pdfannotator.com/en/>. 4.1.1
- Gühring, T., Linz, N., Theis, R., and Friedrich, A. (2016). Swan: an easy-to-use web-based annotation system. 3.2
- Hellwig, B., Uytvanck, D. V., Hulbosch, M., Somasundaram, A., Tacchetti, M., Geerts, J., and Sloetjes, H. (2021). 2.16. copy and paste annotations. <https://www.mpi.nl/corpus/html/elan/ch02s16.html>. 5.2.11
- Hellwig, O., Hettrich, H., Modi, A., and Pinkal, M. (2018). Multi-layer annotation of the rigveda. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA). 13
- Hirschman, L. (1997). Muc-7 coreference task definition. https://www-nlpir.nist.gov/related_projects/muc/proceedings/co_task.html. 2.2.2
- Ide, N. and Pustejovsky, J. (2017). *Handbook of Linguistic Annotation*. Springer Netherlands, 1 edition. 2.2, 1
- Inc., A. (n.d.). Photoshop. 4.1.1, 4.1.1, 4.1.3
- Jovanovic, J. and Bagheri, E. (2017). Semantic annotation in biomedicine: The current landscape. *Journal of Biomedical Semantics*, 8. 2.1
- Karoui, J., Benamara, F., Moriceau, V., Patti, V., Bosco, C., and Aussenac-Gilles, N. (2017). Exploring the impact of pragmatic phenomena on irony detection in tweets: A multilingual corpus study. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 262–272, Valencia, Spain. Association for Computational Linguistics. 5
- Kiesel, J., Wachsmuth, H., Al-Khatib, K., and Stein, B. (2017). Wat-sl: A customizable web annotation tool for segment labeling. In Blunsom, P., Koller, A., and Lapata, M., editors, *Software Demonstrations at the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 13–16. 3.2

BIBLIOGRAPHY

- Kim, E. and Klinger, R. (2018). Who feels what and why? annotation of a literature corpus with semantic roles of emotions. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1345–1359, Santa Fe, New Mexico, USA. Association for Computational Linguistics. 10
- Klie, J.-C., Bugert, M., Boullosa, B., de Castilho, R. E., and Gurevych, I. (2018). The inception platform: Machine-assisted and knowledge-oriented interactive annotation. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 5–9. Association for Computational Linguistics. 3.2
- Lanfranchi, A., Crooks, K., and Hamang, M. (2013). Clinical coreference annotation guidelines (with excerpts from odie guidelines and modified for sharpn /thyme). http://clear.colorado.edu/compsem/documents/coreference_guidelines.pdf. 2.2.2
- Letcher, N. (2013). Linguistic phenomena annotation guidelines. <http://hdl.handle.net/11343/33329>. 16
- Li, Y., Gerdes, K., and Chuanming, D. (2019). Character-level annotation for chinese surface-syntactic universal dependencies. pages 216–226. 2.2, 2
- Lin, G. (2016). *Building Simple Annotation Tools*. PhD thesis, University of California, San Diego, USA. 2.1
- LLC, S. (2021). How can i save one or more parts of a large dataset? <https://www.stata.com/support/faqs/data-management/save-parts-of-dataset/>. 4.1.1
- Maeda, K., Bird, S., Ma, X., and Lee, H. (2001). The annotation graph toolkit: Software components for building linguistic annotation tools. In *Proceedings of the First International Conference on Human Language Technology Research*. 2.2.2
- Montani, I. and Honnibal, M. (2018). Prodigy: A new annotation tool for radically efficient machine teaching. *Artificial Intelligence*, to appear. 4.1.2
- Müller, C. and Strube, M. (2006). Multi-level annotation of linguistic data with MMAX2. In Braun, S., Kohn, K., and Mukherjee, J., editors, *Corpus Technology and Language Pedagogy: New Resources, New Tools, New Methods*, pages 197–214. Peter Lang, Frankfurt a.M., Germany. 3.2
- Musi, E., Stede, M., Kriese, L., Muresan, S., and Rocci, A. (2018). A multi-layer annotated corpus of argumentative text: From argument schemes to

BIBLIOGRAPHY

- discourse relations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA). 14
- Nakayama, H., Kubo, T., Kamura, J., Taniguchi, Y., and Liang, X. (2018). doccano: Text annotation tool for human. Software available from <https://github.com/doccano/doccano>. 3.2
- Neves, M. and Leser, U. (2012). A survey on annotation tools for the biomedical literature. *Briefings in Bioinformatics*, 15(2):327–340. 2.1, 3.1
- Neves, M. and Seva, J. (2019). An extensive review of tools for manual annotation of documents. *Briefings in Bioinformatics*, 22(1):146–163. 2.1, 3.1, 3.2
- Neves, M. and Seva, J. (2020). Annotationsaurus: A searchable directory of annotation tools. 3.1, 3.2
- Nguyen, P. (n.d.). Pattern design in java. <https://web.csulb.edu/~pnguyen/cecs277/lecnotes/introduction%20to%20pattern.html>. 5.1
- Nixon, L. and Troncy, R. (2014). Survey of semantic media annotation tools for the web: Towards new media applications with linked media. In Presutti, V., Blomqvist, E., Troncy, R., Sack, H., Papadakis, I., and Tordai, A., editors, *The Semantic Web: ESWC 2014 Satellite Events*, pages 100–114, Cham. Springer International Publishing. 2.1, 3.1
- Nordquist, R. (2021). Definition and examples of discourse. <https://www.thoughtco.com/discourse-language-term-1690464>. 2.2
- O’ Mahony, N., Campbell, S., Carvalho, A., Krpalkova, L., Riordan, D., and Walsh, J. (2019). Point cloud annotation methods for 3d deep learning. 2.1
- Palmer, M. and Xue, N. (2012). Linguistic annotation. In Clark, A., Fox, C., and Lappin, S., editors, *The Handbook of Computational Linguistics and Natural Language Processing*, volume 1 of 1, chapter 10, pages 238–270. Wiley-Blackwell, <https://www.wiley.com/en-us/The+Handbook+of+Computational+Linguistics+and+Natural+Language+Processing-p-9781118347188>, 1 edition. 4
- Pustejovsky, J. and Stubbs, A. (2012). *Natural Language Annotation for Machine Learning*. O'Reilly Media, Inc. 3.2.1, 3

BIBLIOGRAPHY

- Rama, T., Brekke, P., Nytrø, Ø., and Øvrelid, L. (2018). Iterative development of family history annotation guidelines using a synthetic corpus of clinical text. In *Proceedings of the Ninth International Workshop on Health Text Mining and Information Analysis*, pages 111–121, Brussels, Belgium. Association for Computational Linguistics. 9
- Reidsma, D., Jovanovic, N., and Hofs, D. (2004). *Designing Annotation Tools based on Properties of Annotation Problems*. Number 45 in CTIT TR-04. Centre for Telematics and Information Technology (CTIT), Netherlands. Imported from CTIT. 2.1
- Sloetjes, H. and Wittenburg, P. (2008). Annotation by category: ELAN and ISO DCR. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA). 5.2.11
- Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S., and Tsujii, J. (2012). brat: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations Session at EACL 2012*, Avignon, France. Association for Computational Linguistics. 3.2
- TEI (n.d.). Introducing the guidelines. <https://tei-c.org/support/learn/introducing-the-guidelines/>. 2.1
- The University of Sheffield, U. (n.d.). Performance evaluation of language analysers. <https://gate.ac.uk/releases/gate-6.0-build3764-ALL/doc/tao/splitch10.html>. 4.1.5
- The WebAnno Team, W. (n.d.). Automation. https://webanno.github.io/webanno/releases/2.3.1/docs/user-guide.html#sect_automation. 4.1.2
- Toxboe, A. (n.d.). Ui patterns. <http://ui-patterns.com/>. 2.1
- van Gompel, M. (2019). A survey of nlp annotation platforms. <https://raw.githubusercontent.com/proycon/folia/master/docs/folia.pdf>. 3.2.2
- van Gompel, M., Başar, E., Neumann, A., and van der Klis, M. (n.d.). Flat - folia linguistic annotation tool. 3.2
- van Gompel, M. and Reynaert, M. (2013). Folia: A practical xml format for linguistic annotation à a descriptive and comparative study. *Computational Linguistics in the Netherlands Journal*, 3:63–81. 3.2

BIBLIOGRAPHY

- Watanabe, T. and Wolf, D. F. (2019). Instance segmentation as image segmentation annotation. *CoRR*, abs/1902.05498. 2.1
- Yang, J., Zhang, Y., Li, L., and Li, X. (2018). Yedda: A lightweight collaborative text span annotation tool. 3.2, 4.1.6
- Zampieri, M., Malmasi, S., Paetzold, G., and Specia, L. (2017). Complex word identification: Challenges in data annotation and system performance. In *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017)*, pages 59–63, Taipei, Taiwan. Asian Federation of Natural Language Processing. 11
- Zhao, J., Glueck, M., Breslav, S., Chevalier, F., and Khan, A. (2017). Annotation graphs: A graph-based visualization for meta-analysis of data based on user-authored annotations. <https://www.autodesk.com/research/publications/annotation-graphs>. 2.2.2

Table of contents

[Table of contents](#)

[Overview](#)

[Appraise](#)

[brat](#)

[Markables](#)

[Implementation](#)

[Label boxes](#)

[Single label](#)

[Default](#)

[Highlighted](#)

[Double clicked](#)

[Not logged in](#)

[Logged in \(annotator view\)](#)

[Single token, multi label](#)

[Default](#)

[Highlighted](#)

[Double-clicked](#)

[Not logged in](#)

[Logged in \(annotator view\)](#)

[Multi label](#)

[Default](#)

[Highlighted](#)

[Double-clicked](#)

[Not logged in](#)

[Logged in \(annotator view\)](#)

[Creation](#)

[Highlights](#)

[Overlapping spans of different lengths](#)

[Default](#)

[Highlighted](#)

[Relations](#)

[Implementation](#)

[Binary relation](#)

[Equivalence relation](#)

[Directed](#)

[Single-label to single-label](#)

[Default](#)

[Highlighted](#)
[Double-clicked](#)
[Single-label to multi-label](#)
[Separate labels/not overlapping](#)
[Stacked labels/overlapping](#)
[Multi-label to multi-label](#)

[Undirected](#)

[Single-label to single-label](#)
[Default](#)
[Highlighted](#)
[Double-clicked](#)
[Single-label to multi-label](#)
[Separate labels/not overlapping](#)
[Stacked labels/overlapping](#)

[Creation](#)

[Data formats](#)

[Documents](#)
[Annotations](#)

[doccano](#)

[Markables](#)
[Implementation](#)
[Label boxes](#)
[Highlights](#)
[Non overlapping](#)
[Default](#)
[Highlighted](#)
[Double clicked](#)
[Overlapping](#)

[Relations](#)

[Data formats](#)
[Documents](#)
[Annotations](#)

[FoLiA / FLAT](#)

[Markables](#)
[Label-boxes](#)
[Highlights](#)
[Annotation focus](#)
[Single highlight](#)
[Default](#)
[Highlighted](#)
[Clicked](#)

[Relations](#)

[Data formats](#)

[Documents](#)

[Annotations](#)

[INCEpTION](#)

[Data formats](#)

[Documents](#)

[Annotations](#)

[MMAX2](#)

[Markables](#)

[Label-boxes](#)

[Highlights](#)

[Single highlight](#)

[Default](#)

[Highlighted](#)

[Clicked](#)

[Multi highlight](#)

[Relations](#)

[Directed](#)

[Undirected](#)

[Default](#)

[Highlighted](#)

[Clicked](#)

[Data formats](#)

[Documents](#)

[Annotations](#)

[Swan](#)

[Markables](#)

[Label-boxes](#)

[Single label-box/single span type](#)

[Single label \(conflicting terminology: property\)](#)

[Default](#)

[Highlighted](#)

[Clicked](#)

[Multi property](#)

[Two properties](#)

[Three properties](#)

[Five properties](#)

[Multi label-box/ multi span types](#)

[Same length](#)

[Different lengths](#)

[Highlights](#)

[Relations](#)

[Single to single](#)

[Directed](#)

[Default](#)

[Highlighted](#)

[Clicked](#)

[Click on link](#)

[Click on link label](#)

[Click on first label](#)

[Click on second label](#)

[Undirected](#)

[Two links, different directions](#)

[Default](#)

[Highlighted](#)

[Clicked](#)

[Click on first label](#)

[Clicked on first label then first link label](#)

[Clicked on first label then second link label](#)

[Click on second label](#)

[Clicked on second label then first link label](#)

[Clicked on second label then second link label](#)

[Multi to multi](#)

[Default](#)

[Highlighted](#)

[Clicked](#)

[Data formats](#)

[Documents](#)

[Annotations](#)

[WAT-SL](#)

[Markables](#)

[Label-boxes](#)

[Highlights](#)

[Single highlight](#)

[Default](#)

[Highlighted](#)

[Clicked](#)

[Multi highlight](#)

[Relations](#)

[Data formats](#)

[Documents](#)

[Annotations](#)

[WebAnno](#)

[Important note](#)

[Markables](#)

[Implementation](#)

[Label boxes](#)

[Single label](#)

[Default](#)

[Highlighted](#)

[Double-clicked](#)

[Multi label](#)

[Default](#)

[Highlighted](#)

[Double-clicked](#)

[Highlights](#)

[Relations](#)

[Directed](#)

[Single-label to single-label](#)

[Default](#)

[Highlighted](#)

[Double-clicked](#)

[Multi-label to multi-label](#)

[Default](#)

[Highlighted](#)

[Double-clicked](#)

[Undirected](#)

[Note](#)

[Creation](#)

[Directed](#)

[Undirected](#)

[Data formats](#)

[Documents](#)

[Annotations](#)

[YEDDA](#)

[Markables](#)

[Label-boxes](#)

[Highlights](#)

[Single highlight](#)

[Default](#)

[Highlighted](#)

[Clicked](#)
[Multi highlight](#)
[Relations](#)
[Data formats](#)
[Documents](#)
[Annotations](#)

Overview

Screenshots of various annotation tools and how they handle basic annotation units, tasks and features.

Annotation features

	brat	doccano	flat	INCEPTION	MMAX2	Swan	WAT-SL	WebAnno	YEDDA
Markables	True	True	True	True	True	True	True	True	True
Single property (stored as label)	True	False	False	True	False	True	False	True	False
Multi property (stored as label)	True	False	False	True	False	True	False	True	False
Highlights as separate property storage	False	True	True	False	True	False	True	False	True
Single property (stored as highlight)	True	True	True	True	True	False	True	True	True
Multi property (stored as highlight)	True	False	True	True	False	False	False	True	False
Multi property (combined)	True	False	True	True	False	True	False	True	False
Relations	True	False	True	True	True	True	False	True	False
Directed	True	False	True	True	True	True	False	True	False
Undirected	True	False	True	False	True	False	False	False	False
Chains	nan	nan	nan	nan	nan	nan	nan	True	nan

Information communicated on mouse-over

Information communicated on mouse-over	brat	doccano	flat	INCEPTION	MMAX2	Swan	WAT-SL	WebAnno	YEDDA
Information communicated on mouse-over	nan	nan	nan	nan	nan	nan	nan	nan	nan
ID	True	False	False	True	False	False	True	True	False
Content	True	False	True	True	False	True	False	True	False
Type	True	False	True	True	False	True	False	True	False
Error message	True	False	nan	True	False	False	False	True	False
Customizable	False	False	True	False	False	False	False	False	False

*due to FLAT's customizability, it is possible, but not mandatory, to include Error messages in tooltips.

Interface/display features

	brat	doccoano	flat	INCEPTION	MMAX2	Swan	WAT-SL	WebAnno	YEDDA
Single label	True	False	False	True	False	True	False	True	False
Multi label	True	False	False	True	False	True	False	True	False
Additional information on mouse-over (label)	True	False	False	True	False	True	False	True	False
Additional information on single click (label)	False	False	False	False	False	True	False	False	False
Additional information on double click (label)	True	False	False	True	False	False	False	True	False
Single highlight	True	True	True	True	True	False	True	True	True
Multi highlight	True	False	True	True	False	False	False	True	False
Highlights independent from label boxes	False	True	True	False	True	False	True	False	True
Additional information on mouse-over (highlight)	True	False	True	True	False	False	True	True	False
Additional information on single click (highlight)	False	False	True	False	True	False	True	False	False
Additional information on double click (highlight)	True	False	False	True	False	False	False	True	False
Overlapping highlights of differing lengths	True	False	True	True	False	False	False	True	False
Relations	True	False	True	True	True	True	False	True	False
Additional information on mouse-over (relation)	True	False	nan	True	False	False	False	True	False
Additional information on single click (relation)	False	False	nan	False	True	True	False	False	False
Additional information on double click (relation)	True	False	nan	True	False	False	False	True	False

*Due to failure to create relations with FLAT, we weren't able to test and document relation tooltips.

Data formats

	brat	doccoano	flat	INCEPTION	MMAX2	Swan	WAT-SL	WebAnno	YEDDA
Document formats	nan	nan	nan	nan	nan	nan	nan	nan	nan
(import) plaintext support	TRUE	TRUE	FALSE	TRUE	False	TRUE	TRUE	TRUE	TRUE
(import) JSON support	FALSE	TRUE	FALSE	FALSE	False	FALSE	FALSE	FALSE	FALSE
(import) CSV support	FALSE	TRUE	FALSE	FALSE	False	FALSE	FALSE	FALSE	FALSE
(import) XML support	FALSE	FALSE	FALSE	TRUE	True	FALSE	FALSE	FALSE	FALSE
(import) other format	FALSE	TRUE	TRUE	TRUE	False	FALSE	FALSE	TRUE	False
Noteworthy import formats	nan	CoNLL	FoLiA	WebAnno TSV	nan	nan	nan	WebAnno TSV	nan
import format type	nan	plaintext based	XML-based	plaintext based	nan	nan	nan	plaintext based	nan
Standoff / separate annotation format	TRUE	TRUE	Mix	TRUE	True	TRUE	TRUE	TRUE	TRUE
Annotation formats	nan	nan	nan	nan	nan	nan	nan	nan	nan
(export) plaintext support	FALSE	FALSE	FALSE	TRUE	False	FALSE	FALSE	TRUE	FALSE
(export) JSON support	FALSE	TRUE	FALSE	FALSE	False	FALSE	FALSE	FALSE	FALSE
(export) CSV support	FALSE	FALSE	FALSE	FALSE	False	FALSE	TRUE	FALSE	FALSE
(export) XML support	FALSE	FALSE	FALSE	TRUE	True	TRUE	FALSE	FALSE	FALSE
(export) other format	TRUE	FALSE	FALSE	TRUE	False	TRUE	TRUE	TRUE	TRUE
Noteworthy export formats	.ann	nan	FoLiA	WebAnno TSV	nan	UIMA XMI	.ann	WebAnno TSV	.ann
Export format type	plaintext based	nan	XML-based	plaintext based	nan	XML-based	plaintext based	plaintext based	plaintext based

These overviews visualize the contents of the .csv files in which we marked down observed properties of the tested annotation software.

The visualizations were created using a custom python script, with the goal of allowing the combination of different .csv files into one table for easy data exploration.

The script was developed for internal use only, with the intended purpose of aiding the discovery of patterns (in the common word sense, not software pattern) across the different tools.

Example of visual exploration:

This table is created by combining the .csv files containing the data of “annotation features” and “data formats”, and filters for “tools which allow the annotation of multiple overlapping properties”

In this example one observable pattern might be that none of the tools with the ability to annotate overlapping properties seem to support JSON or CSV files for both import and export.

Subjective observations about the tested software:

brat: straightforward

WebAnno: similar to brat, but needlessly complicated for undirected relations

Appraise: broken installation

doccano: missing most features

INCEpTION: identical to WebAnno in terms of how basic functionality is handled

FoLiA/flat: very hard to install

Swan: straightforward, some limited functionality

WAT-SL: missing most features

MMAX2: unintuitive nightmare

yedda: missing most features & visually displeasing

Note about installation instructions

We documented the installation process of each software we tested.

This documentation can be found in the “A2_Appendix_Installation_Documentation.pdf” file attached to the digital copy of the thesis.

Note about screenshot attribution

All screenshots included in this document were created by the author of this thesis, unless specified otherwise.

Appraise

<https://github.com/cfedermann/Appraise>

The installation instructions on the github were incorrect at the time of testing, and the official website www.appraise.cf was not reachable during the testing phase.

The github was updated on 06.05.2021 and the website seems to be reachable again, which was only noticed during the proof-reading phase of this thesis (15.05.2021). An archived version of the github, as available during the research phase of this thesis, can be found here: <https://web.archive.org/web/20201101143415/https://github.com/cfedermann/Appraise>

Due to the incorrect installation instructions, we were unable to get the software running and test it.

brat

Markables

Implementation

Source of screenshot and quote: <https://brat.nlplab.org/standoff.html>

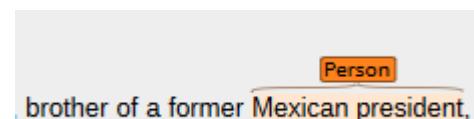
T1	Organization	0 4	Sony
T3	Organization	33 41	Ericsson
T3	Country	75 81	Sweden

“Each entity annotation has a unique ID and is defined by type (e.g. Person or Organization) and the span of characters containing the entity mention (represented as a “start end” offset pair).”

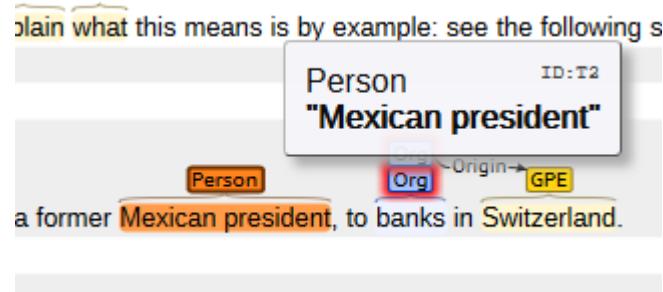
Label boxes

Single label

Default

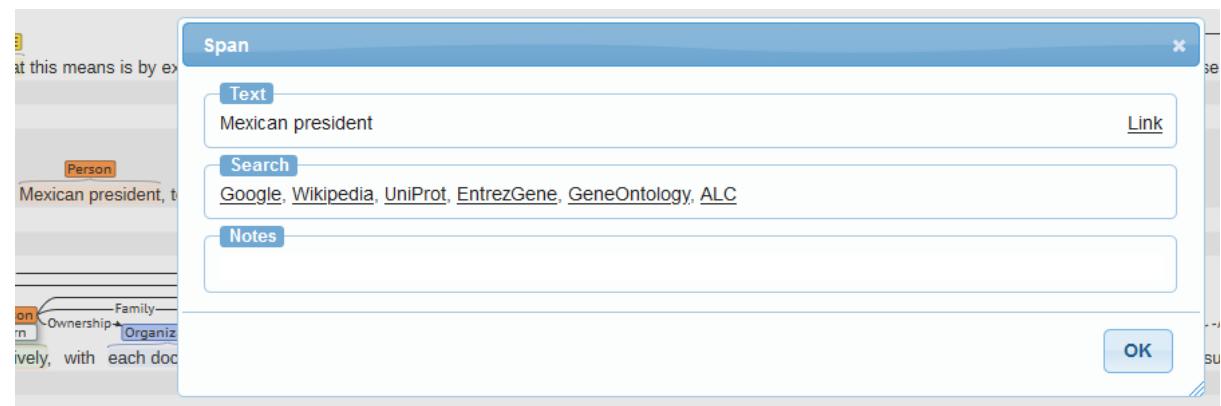


Highlighted

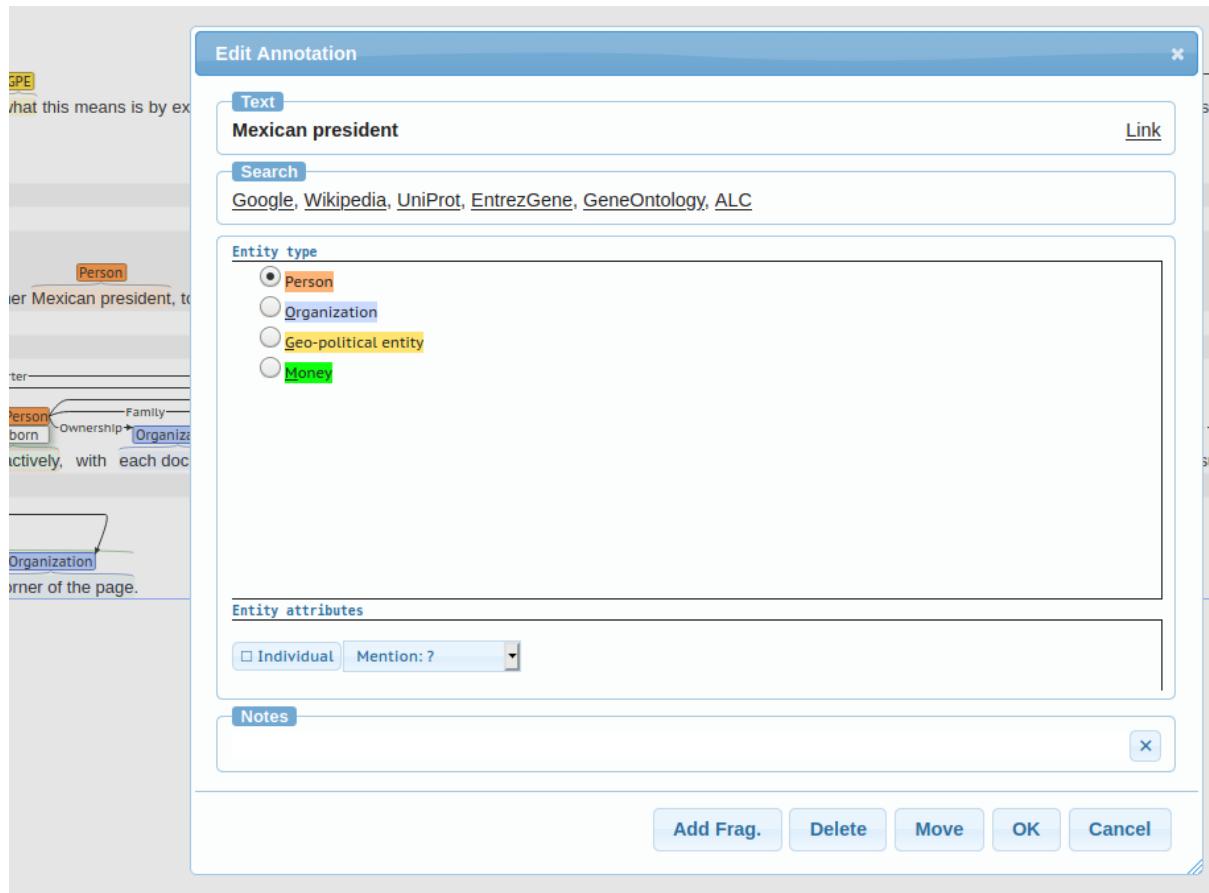


Double clicked

Not logged in



Logged in (annotator view)



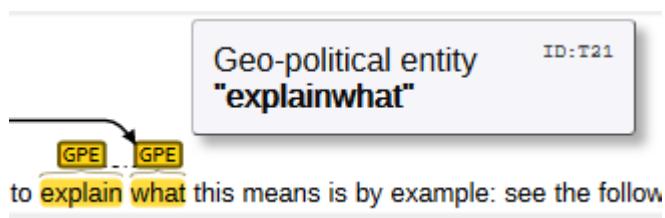
Single token, multi label

Default



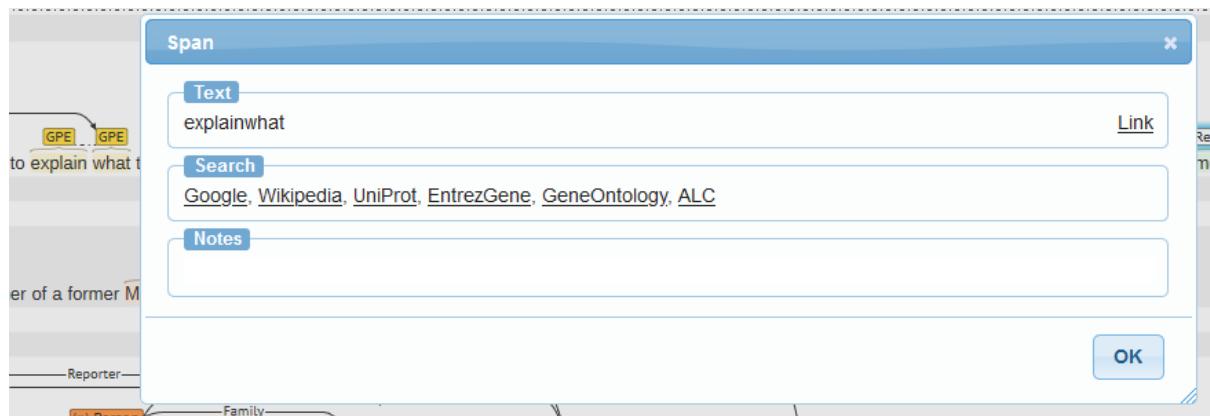
"explain what" is one token, it is unclear why the label is split.

Highlighted

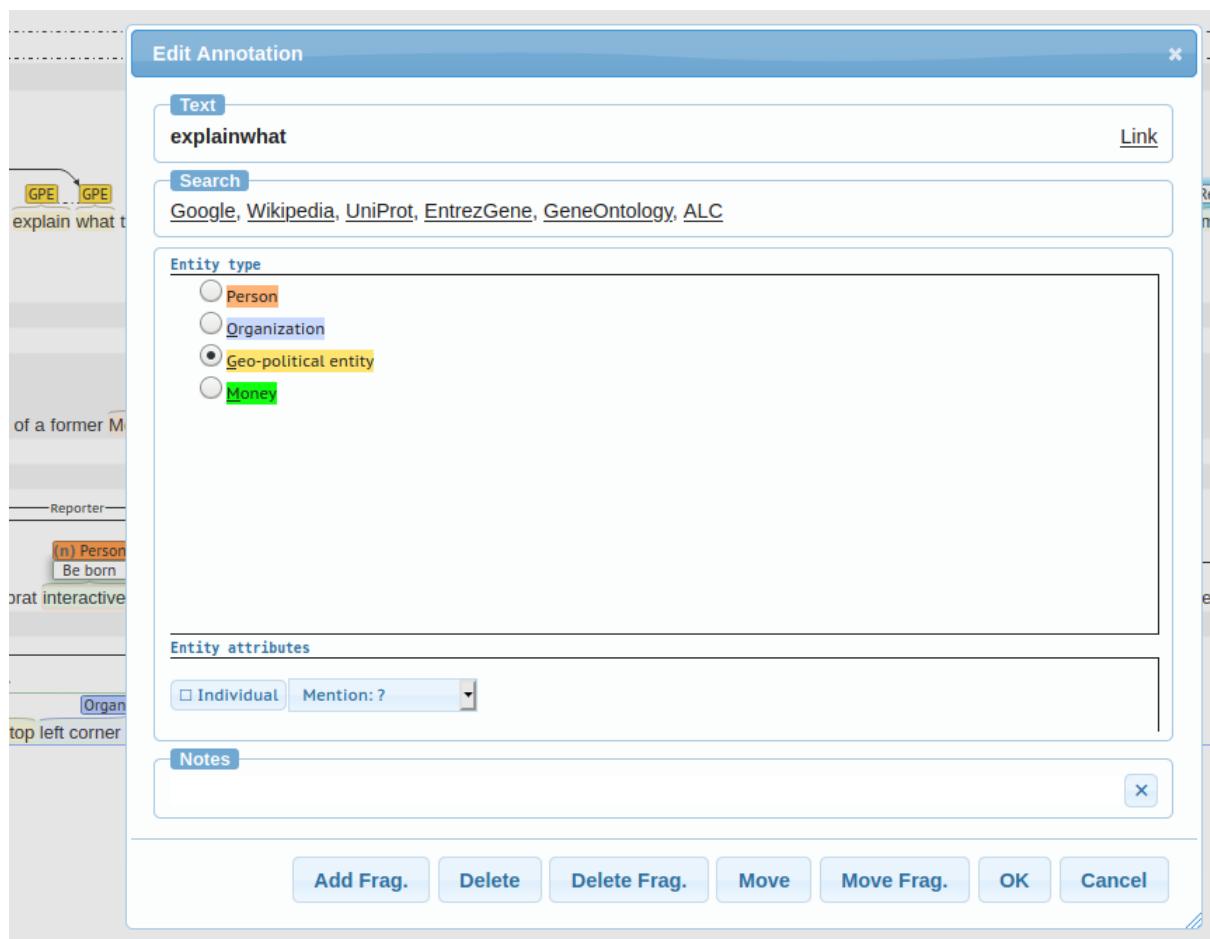


Double-clicked

Not logged in



Logged in (annotator view)



Multi label

Default



Highlighted

A screenshot of the Brat annotation interface. A specific annotation is highlighted with a red background. The text in the annotation is: "Welcomebrat". An error message is displayed: "AnnotationError: Error: Person cannot contain Organization (T70)". The cursor is positioned over the bottom 'Person' label in the annotation. The interface includes a navigation bar at the top with links like '/editable/tutorials/news/000-introduction' and a sidebar on the left.

In the screenshot above, the cursor is on bottom label box “Person”.

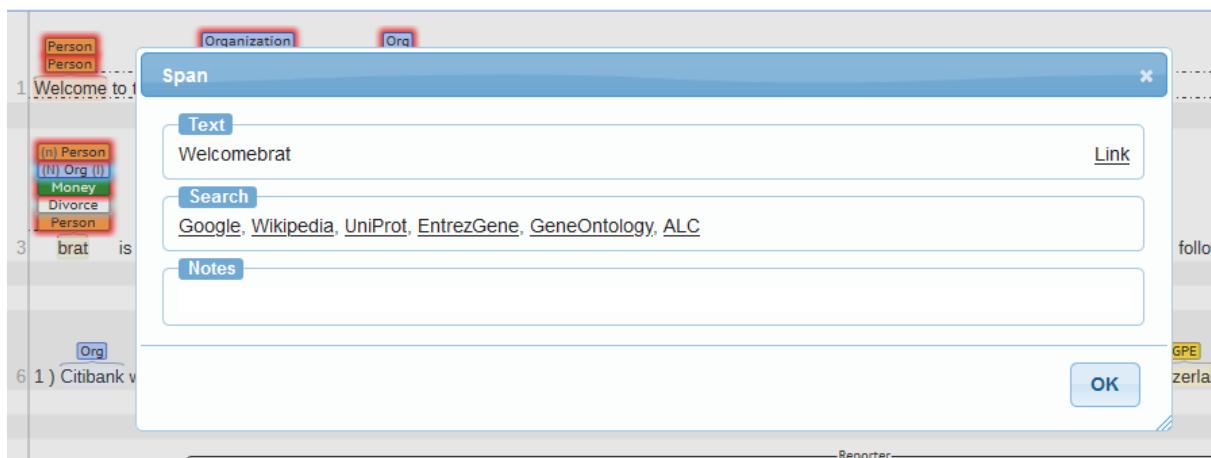
A screenshot of the Brat annotation interface. A specific annotation is highlighted with a yellow background. The text in the annotation is: "brat". An error message is displayed: "AnnotationIncomplete: Incomplete: exactly 2 Person arguments required for event". The cursor is positioned over the second-to-bottom 'Divorce' label in the annotation. The interface includes a navigation bar at the top with links like '/editable/tutorials/news/000-introduction' and a sidebar on the left.

In the screenshot above the cursor is on second-to-bottom label box “Divorce”

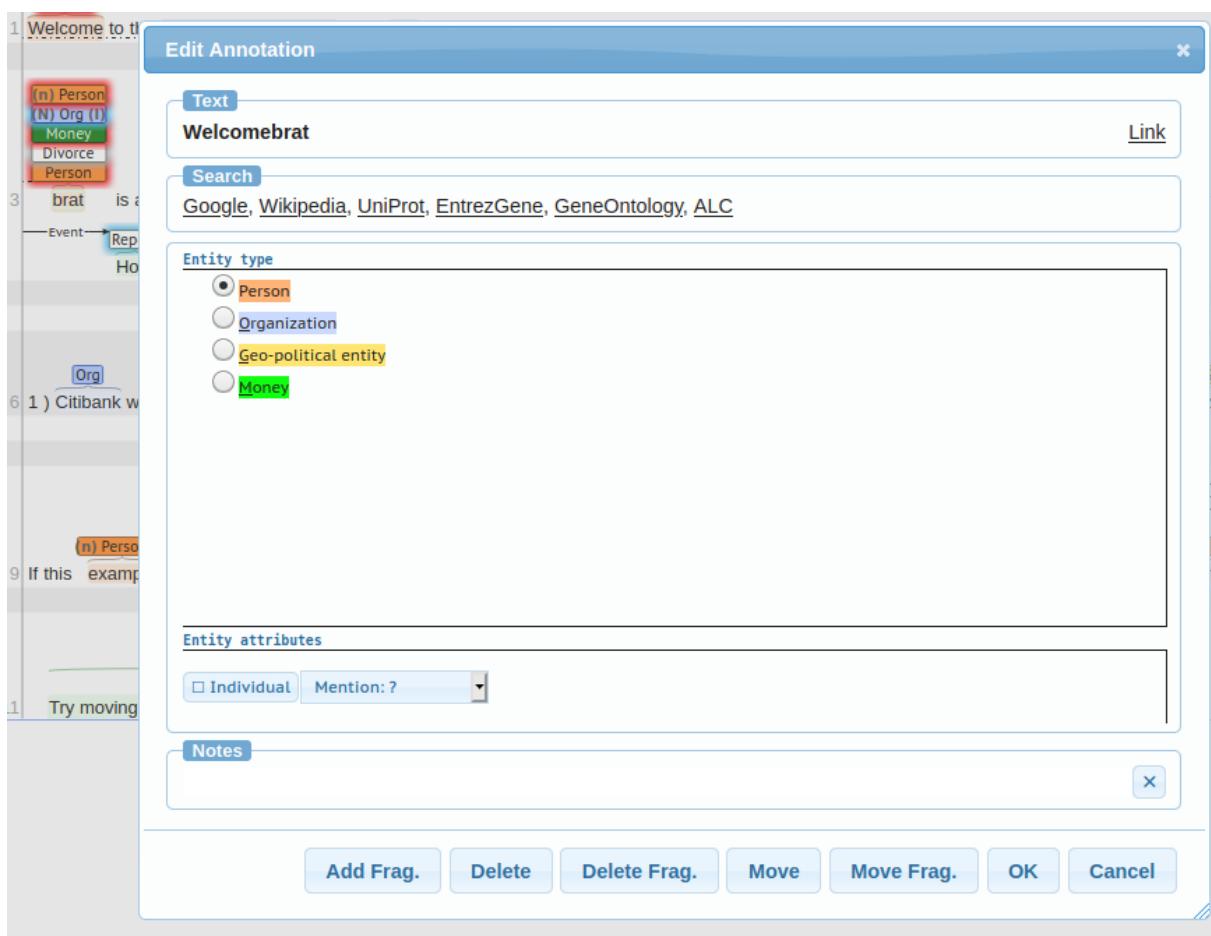
Double-clicked

Clicked on bottom label box “Person”.

Not logged in



Logged in (annotator view)



Creation

In brat creating label annotations works in three different ways:

1. Double clicking a word in the text. This creates a new annotation for the entire word.
2. Highlighting an arbitrary length of text by clicking+dragging. This creates a new annotation for the entire selected span.

3. Manually editing the stand off .ann file.

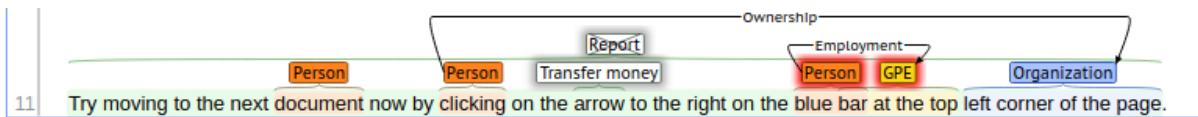
Highlights

In brat annotations are always a combination of highlights and label boxes, so there are no examples of “just highlights” - see the section for label boxes for examples.

One noteworthy observation is how brat handles overlapping text spans of different lengths:

Overlapping spans of different lengths

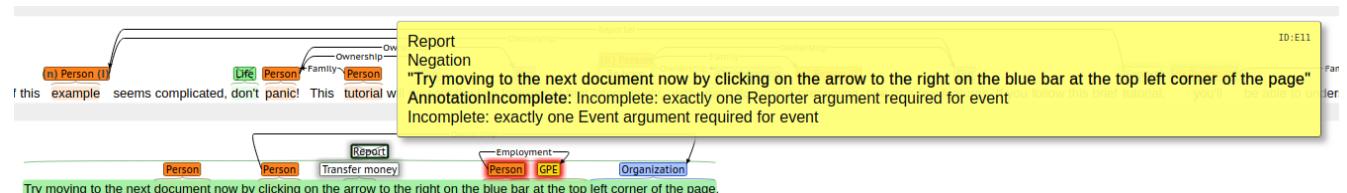
Default



The text is highlighted in various background colours. In the default view, where no specific annotation is highlighted, shorter annotations cancel out the colour of the annotation they are overlapping.

This only becomes obvious once specific annotations are highlighted, revealing the actual length of the text span:

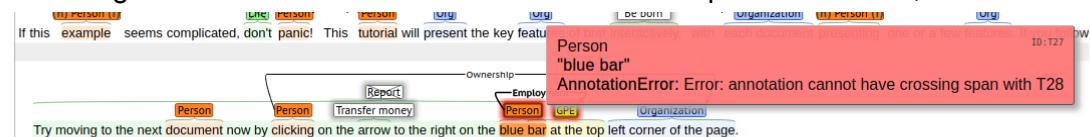
Highlighted



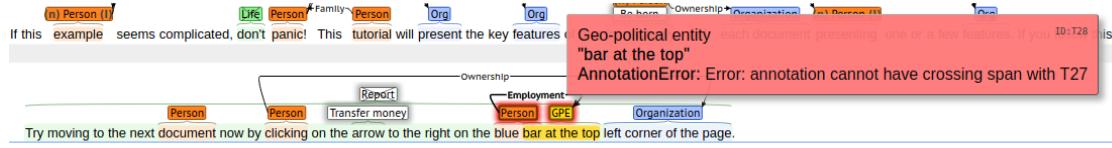
In the screenshot above the crossed-out “Report” annotation is highlighted, revealing that the annotation actually spans the entire length of the sentence.



In the screenshot above the “Transfer money” annotation on the word “arrow” is highlighted, revealing an annotation of the same colour as the “Report” annotation, but of different length



In the screenshot above the “Person” annotation on the words “blue bar” is highlighted.



In the screenshot above the “GPE” annotation on the words “bar at the top” is highlighted, revealing that this annotation in the default view was both overlapping and therefore obscuring the highlight of the “Report” annotation, while at the same time being partially overlapped and therefore obscured by the preceding “Person” annotation on the words “blue bar”.

Relations

Implementation

Source of the two screenshots and two quotes: <https://brat.nlplab.org/standoff.html>

Binary relation

R1	Origin Arg1:T3 Arg2:T4
----	------------------------

“Binary relations have a unique ID and are defined by their type (e.g. Origin, Part-of) and their arguments. “

Equivalence relation

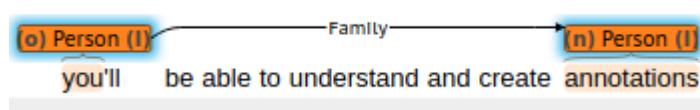
T1	Organization 0 43	International Business Machines Corporation
T2	Organization 45 48	IBM
T3	Organization 52 60	Big Blue
*	Equiv T1 T2 T3	

“Equivalence relations are symmetric and transitive relations that define sets of annotations to be equivalent in some sense (e.g. referring to the same real-world entity). Such relations can be represented in a compact way as a SPACE-separated list of the IDs of the equivalent annotations. “

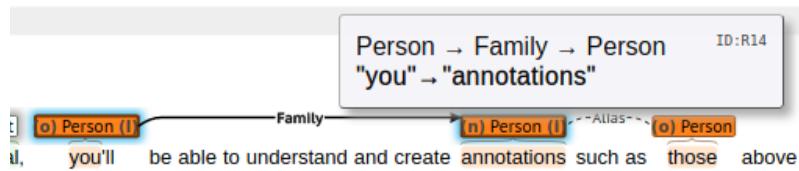
Directed

Single-label to single-label

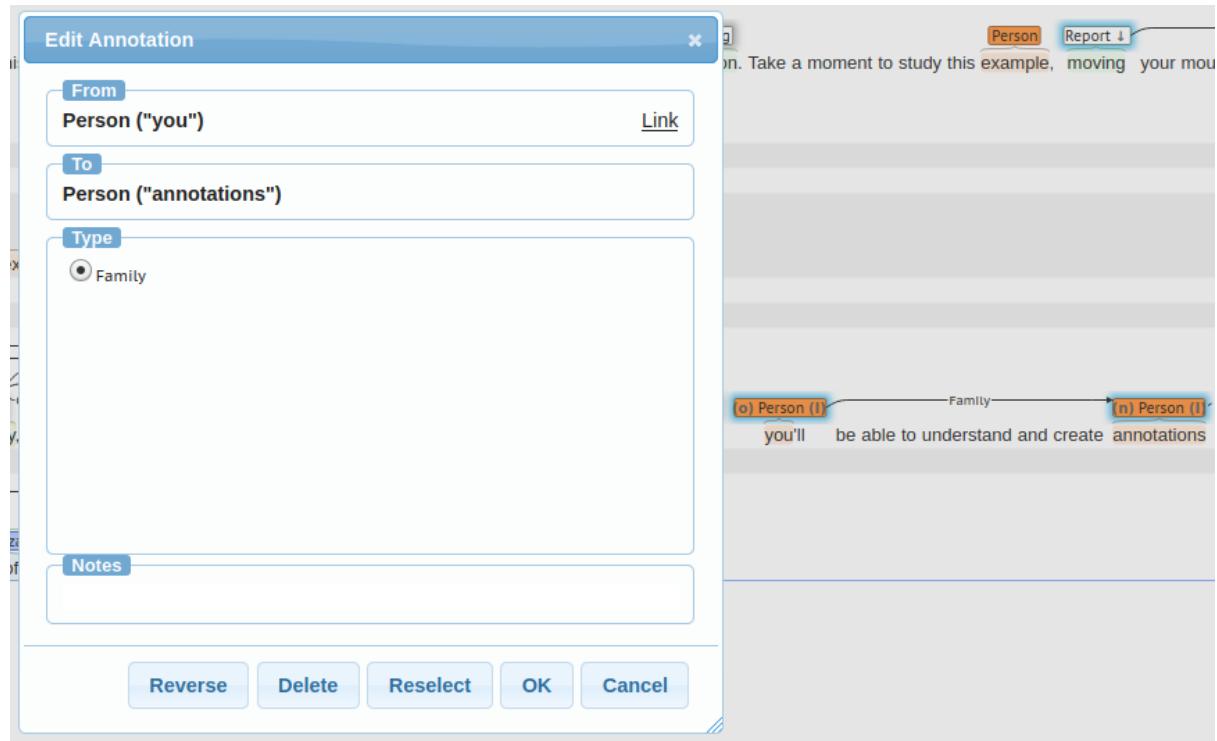
Default



Highlighted



Double-clicked



Note: double-clicking relations only works when logged in

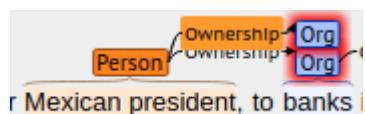
Single-label to multi-label

Separate labels/not overlapping

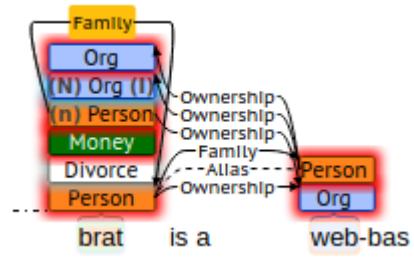


As highlighting and double-clicking show the same behaviour as if there was only one relation, we decided to not produce duplicate screenshots.

Stacked labels/overlapping



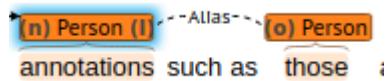
Multi-label to multi-label



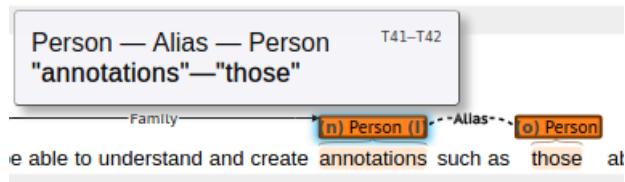
Undirected

Single-label to single-label

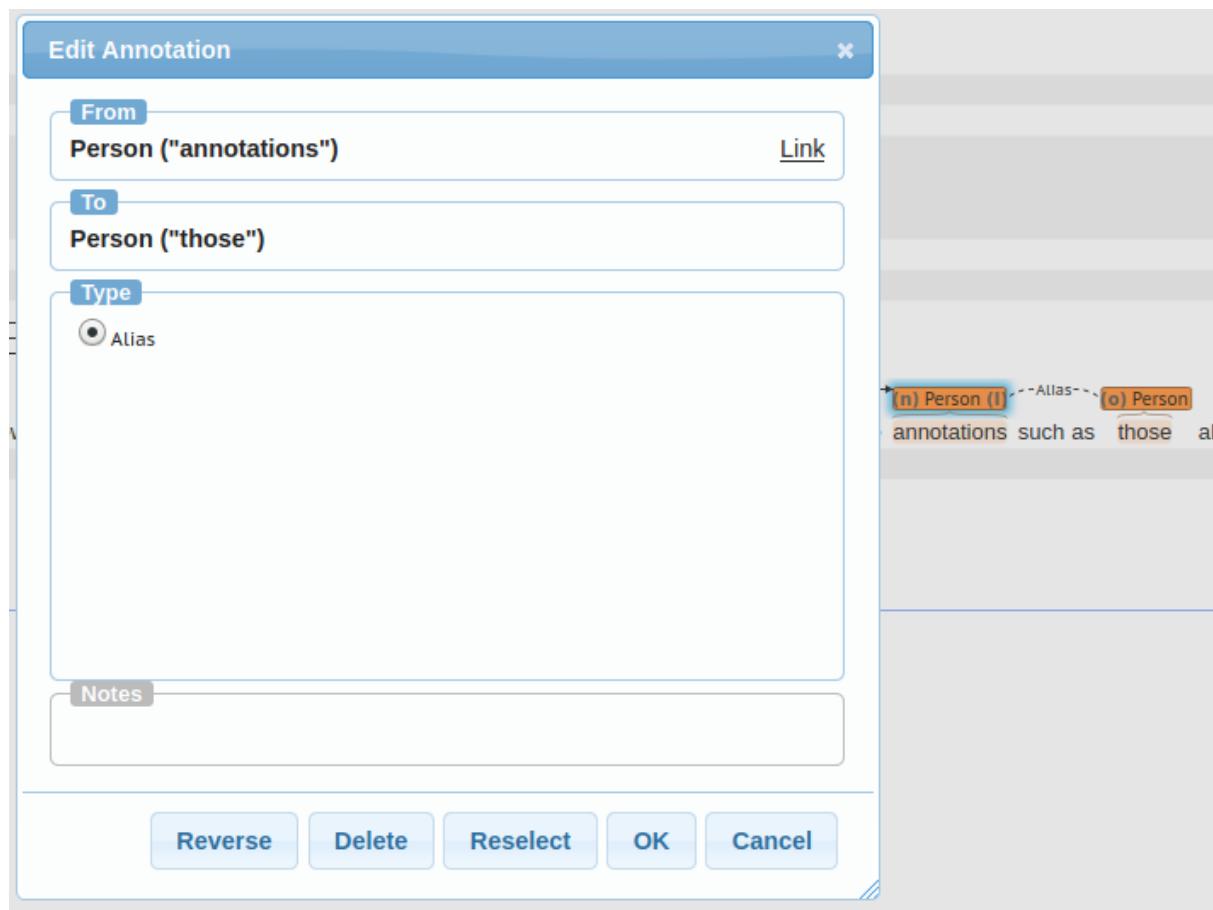
Default



Highlighted

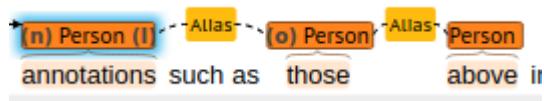


Double-clicked



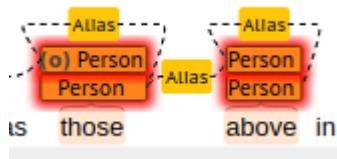
Single-label to multi-label

Separate labels/not overlapping



In this example there is an undirected connection between “annotations” and “above” - however, brat chooses to display this relation like this.

Stacked labels/overlapping



Creation

Relations in brat are created by click-dragging from one label-box to another.
The direction of the relation is always from the label-box clicked on to the label-box where the mouse-click is released.

Data formats

<https://brat.nlplab.org/standoff.html>

Documents

Documents are stored in .txt format.

Annotations

Annotations are stored in .ann standoff format, separate from the actual document.
.ann is a plaintext-based format.

doccano

Markables

Implementation

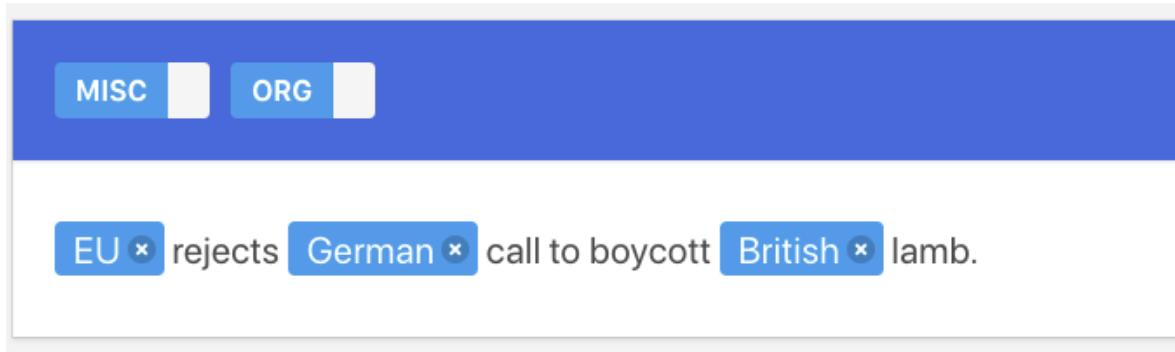
<https://github.com/doccano/doccano/wiki/Import-and-Export-File-Formats>

Source of the following screenshot and quote is the GitHub-User “BrambleXu” in the following issue discussion:

<https://github.com/doccano/doccano/issues/113>

(archived:

<https://web.archive.org/web/20200920134252/https://github.com/doccano/doccano/issues/113>



```
{  
"doc_id": 15, "text": "EU rejects German call to boycott British lamb.",  
"labels": [[0, 2, "ORG"], [11, 17, "MISC"]],  
"meta": {}  
}
```

Label boxes

Doccano doesn't seem to feature label boxes, all annotations are on-text (highlights).

Highlights

Non overlapping

Default

Barack Hussein Obama II

Highlighted

Not an option.

Double clicked

Not an option.

Overlapping

Not supported.

Relations

Not supported.

Data formats

Source of the following screenshot:

<https://github.com/doccano/doccano/wiki/Import-and-Export-File-Formats>

Documents

Documents can be imported as different formats, depending on the annotation task.

Task x format is as follows:

	Plain	CSV	JSON	CoNLL
Text Classification	<input type="radio"/>	<input type="radio"/> (single label)	<input type="radio"/>	X
Sequence Labeling	<input type="radio"/>	X	<input type="radio"/>	<input type="radio"/>
Seq2seq	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	X

http://mwetoolkit.sourceforge.net/PHITE.php?sitesig=MWE&page=MWE_070_File_types&subpage=MWE_010_CONLL

CoNLL appears to be a plaintext-based, tab-separated file format.

Not to be confused with CoNLL-X or CoNLL-U.

Annotations

Annotations can be exported in JSON format.

FoLiA / FLAT

Markables

Label-boxes

Highlights

Annotation focus

FLAT allows the user to change the “Annotation Focus”, highlighting different parts of the annotation.

This is an example of the “Part-of-Speech” focus in one example project:

The screenshot shows a block of Dutch text from a historical document. The text is annotated with various colors: purple for 'Nova Zembla', green for 'Novgorod', yellow for 'West-Europeanen', red for '16e eeuw', and blue for 'Hugh Willoughby'. The text discusses the discovery of Nova Zembla by Russian traders in the 11th or 12th century, and the subsequent expeditions by Willem Barentsz and Jacob van Heemskerck in 1596 and 1597, respectively. It also mentions the first Western visitor, Hugh Willoughby, and the 'Nova Zembla-effect'.

The available variations of “Annotation Focus” however are defined per document.

Single highlight

Default

I go to the bank.

Highlighted

I go to the **bank**.

This screenshot shows a detailed annotation interface for the word 'bank'. The top bar indicates the source is 'Word/Token * example.p.1.s.1.w.5 * WORD'. The 'Text' section shows the URL 'https://raw.githubusercontent.com/proycon/folia/master/setdefinitions/text.foliaset.ttl'. The 'Semantic Sense' section shows 'wordnet' and 'r_n-5919' with a 'Confidence 100%' and a 'Description: financial institution'. The 'Domain' section shows 'topics' and 'finance'. The entire interface has a light blue background.

Clicked



Relations

https://flat.readthedocs.io/en/latest/user_guide.html#adding-new-annotations

It is possible to add relations in FLAT, unfortunately we weren't able to reproduce it in practice.

Data formats

<https://proycon.github.io/folia/>

Appendix to "Requirements engineering for natural-language annotation tasks" Bachelor's Thesis by
Vincent Söllner, Bauhaus-Universität Weimar, Matriculation Number 118764

Documents

Documents can only be imported in the FoLiA format.

FoLiA is an XML-based format that combines standoff and inline annotation.

Annotations

Annotations can be exported in FoLiA format.

INCEpTION

<https://webanno.github.io/webanno/info/>

WebAnno 3 III (2018-now)

In this phase, the development of WebAnno is mainly driven by the requirements of the INCEpTION project. The principal goal is the modularization and modernization of the architecture to allow re-using parts of it in INCEpTION.

It appears as if INCEpTION was built on the foundation of WebAnno and doesn't differ in its basic building blocks, but only in higher functionality. We weren't able to observe differences in how markables and relations are treated when compared to WebAnno, nor any differences in the user interface in relation to the annotation tasks we seek to document in this file.

As a result, we decided to not include duplicated screenshots in this document, and instead refer to the [WebAnno section](#) (we tested WebAnno before INCEpTION, which is why the screenshots are included there, despite INCEpTION coming earlier in the lexicographic order).

Data formats

https://inception-project.github.io/releases/0.19.3/docs/user-guide.html#sect_formats

Documents

Documents can be imported in a variety of formats.

Available formats at the time of testing:

Appendix A: Formats

- [CoNLL 2000](#)
- [CoNLL 2002](#)
- [CoNLL 2003](#)
- [CoNLL 2006](#)
- [CoNLL 2009](#)
- [CoNLL 2012](#)
- [CoreNLP CoNLL-like format](#)
- [CoNLL-U](#)
- [IMS CWB \(aka VRT\)](#)
- [Inline XML](#)
- [LAPPS Interchange Format](#)
- [NLP Interchange Format](#)
- [Perseus Ancient Greek and Latin Dependency Treebank 2.1 XML](#)
- [WebLicht TCF](#)
- [TEI P5 XML](#)
- [Plain Text](#)
- [UIMA Binary CAS](#)
- [UIMA CAS XMI](#)
- [WebAnno TSV 1](#)
- [WebAnno TSV 2](#)
- [WebAnno TSV 3.x](#)

Annotations

Annotations can be exported in a variety of formats

MMAX2

Surprisingly MMAX2 was updated in 2020 - not feature-wise but to support a new java version.o

MMAX2 is incredibly complex and features a user interface with multiple floating windows and multiple tabs in each window.

Due to this complexity, unfamiliarity with the software and the time constraints of the thesis, it is possible that, despite our best efforts, some information relevant to the annotation task was missed and is therefore not included in the screenshots.

Markables

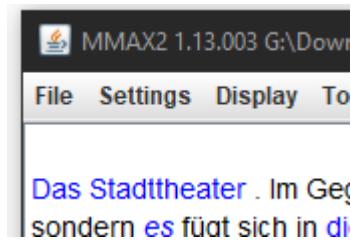
Label-boxes

Not supported.

Highlights

Single highlight

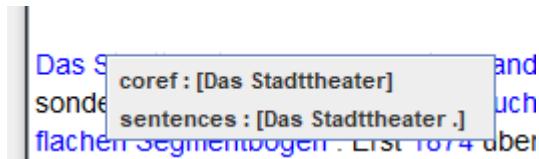
Default



Highlighted

Not supported.

Clicked



Multi highlight

Not supported.

Relations

Directed

Directed relations are supported according to the documentation:

<http://mmax2.net/mmax2quickstart.pdf>

However, we weren't able to figure out the creation process of such directed relations, and were thus unable to document the creation process.

Undirected

Default

Undirected relations don't show up unless a label belonging to the chain/bridge/link is selected.

Highlighted

Not supported.

Clicked

Das Stadttheater . Im Gegensatz zu anderen Städten steht das Heidelberger Stadttheater nicht an herausgehobener Stelle , sondern es fügt sich in die Straßenflucht ohne vorplatz ein . Der Haupteingang zeigt noch das alte Arkadenmotiv mit den flachen Segmentbögen . Erst 1874 übernahm die Stadt das bis dahin von einer privaten Initiative getragene Theater . Es wurde in der Folge stark verändert . Nach dem Innen-Umbau von 1880 durch Hermann Behagel gestaltete Fritz Haller 1924 das Haus erneut um . Trotz Aufstockung und Verbreiterung im Stil des Neoklassizismus wurden aber Teile der Straßenfassade erhalten . 1990 wurde seitlich ein gläsernes Foyer von Rudolf Biste und Kurt Gerling angebaut .

Data formats

<http://mmax2.net/mmax2quickstart.pdf>

Documents

Documents are stored in XML format.

Annotations

Annotations are stored in XML format, one XML file per markable level.

Swan

Markables

Label-boxes

Swan provides the option to assign multiple labels to a single span, and the option to stack multiple span types.

This conflicts with our way of looking at labels and label boxes. The interchangeable use of 'label' and 'property' also becomes an issue.

Single label-box/single span type

Single label (conflicting terminology: property)

Default

the Aegean to

Highlighted

the Aegean toward the horizon
Type: span1 | Labels: label1

Clicked

ss the Aegean toward the horizon and see the faint silhouette of land. Their curiosity pushed them to

label1 | 'Aegean'

The screenshot shows a user interface for text annotation. At the top, there is a text input field containing the sentence: "ss the Aegean toward the horizon and see the faint silhouette of land. Their curiosity pushed them to". A pink highlight box is placed over the word "Aegean". To the right of the text, a small box displays the label "label1 | 'Aegean'". Below the text input, there is a horizontal bar with the label "labe lhl1".

Annotations

SPAN TYPE	labelSet1 (multiple allowed)	labelSet2 (multiple allowed)
span1	label1	lbl1
span2	label2	lbl2
	label3	lbl3

not sure

[Remove annotation](#)

Multi property

Two properties

Default:

the Aegean to

Highlighted:

he Aegean toward the horizon and se
Type: span1 | Labels: label1 label2

Clicked:

la la
oss the Aegean toward the horizon and see the faint silhouette of land. Their curiosity pushed them to

label label | 'Aegean'

The screenshot shows a user interface for annotation. At the top, there are two purple boxes: one labeled "la la" and another labeled "label label | 'Aegean'". Below this is a horizontal bar with a pink segment labeled "labe lbl1". The main area is titled "Annotations". It contains three sections: "SPAN TYPE" with "span1" and "span2"; "labelSet1 (multiple allowed)" with "label1", "label2", and "label3"; and "labelSet2 (multiple allowed)" with "lbl1" and "lbl2". There is also a "not sure" checkbox and a "Remove annotation" button. A "Project navigation" bar is at the bottom.

Three properties

Default:

la Aegean t

Highlighted:

s the Aegean toward the horizon and see

Type: span1 | Labels: label1 label2 label3

Clicked:

oss the Aegean toward the horizon and see the faint silhouette of land. Their curiosity pushed them to

lab lab lab | 'Aegean'

The screenshot shows a user interface for annotation, identical to the previous one but with more highlighted text. The top purple box "la la" now highlights "oss the Aegean toward the horizon and see the faint silhouette of land. Their curiosity pushed them to". The middle purple box "label label | 'Aegean'" highlights "lab lab lab | 'Aegean'". The main annotations section shows the same three sets: SPAN TYPE (span1, span2), labelSet1 (label1, label2, label3), and labelSet2 (lbl1, lbl2). The "not sure" checkbox and "Remove annotation" button are present. A "Project navigation" bar is at the bottom.

Five properties

Default:

; the Aegean to

Highlighted:

he Aegean toward the horizon and see the faint :

Type: span1 | Labels: label1 label2 label3 lbl1 lbl2

label1

Clicked:

he Aegean toward the horizon and see the faint silhouette of land. Their curiosity pushed them to

la la la lb lb | 'Aegean'

Annotations

SPAN TYPE	labelSet1 (multiple allowed)	labelSet2 (multiple allowed)
span1	label1	lbl1
span2	label2	lbl2
	label3	

not sure

[Remove annotation](#)

Multi label-box/ multi span types

Same length

lbl1
labe
he Aegean tc

Different lengths

label1
lbl1
labe
; the Aegean toward the h

Granularity seems to be word-size

Highlights

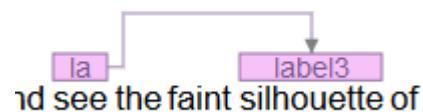
Not supported.

Relations

Single to single

Directed

Default



Highlighted

Not an option.

Clicked

Click on link

Not an option.

Click on link label

This option only becomes available after clicking on one of the labels linked.



and see the faint silhouette of la. Their curiosity pushed them to

label3 | 'silhouette'
label3 | 'see'

Annotations

Remove link

LINK TYPE
linkType1
(select one)

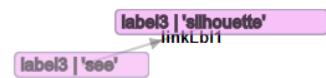
linkLb1
linkLb2

label3 'silhouette'
label3 'see'

Click on first label



Click on second label

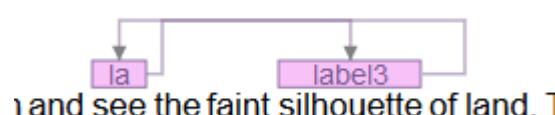


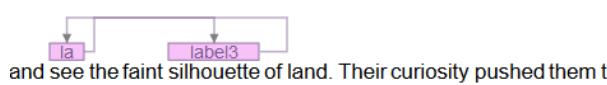
Undirected

Not supported.

Two links, different directions

Default





and see the faint silhouette of land. Their curiosity pushed them to

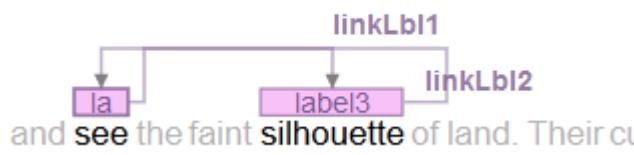
label3 | 'see' label3 | 'silhouette'

Highlighted

Not supported.

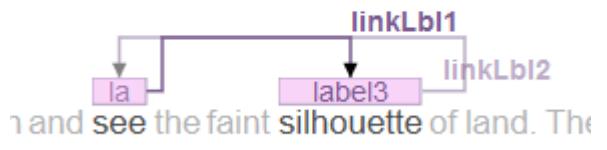
Clicked

Click on first label



and see the faint silhouette of land. Their cu

Clicked on first label then first link label



and see the faint silhouette of land. The

and see the faint silhouette of land. Their curiosity pushed them to

label3 | 'see' label3 | 'silhouette'

Annotations

[Remove link](#)

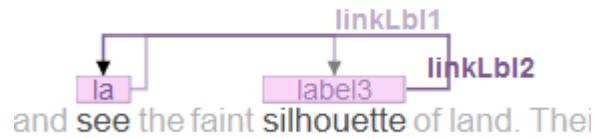
LINK TYPE

linkType1
(select one)

<input checked="" type="radio"/> linkLbI1
<input type="radio"/> linkLbI2

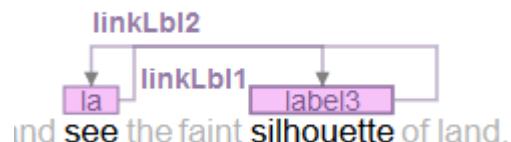
[Cancel](#) [Save](#)

Clicked on first label then second link label

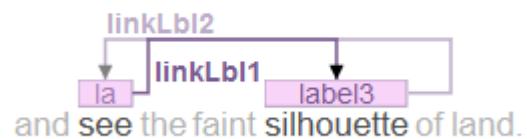


label3 | 'see'
label3 | 'silhouette'
label3 | 'land'

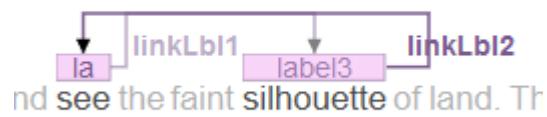
Click on second label



Clicked on second label then first link label

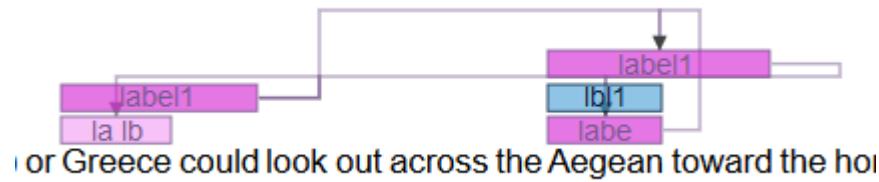


Clicked on second label then second link label



Multi to multi

Default



or Greece could look out across the Aegean toward the horizon ε

Highlighted

Not supported.

Clicked



ey) or Greece could look out across the Aegean toward the horizon ε

Data formats

<https://github.com/annefried/swan/wiki/Projects>

Documents

Documents are stored in plaintext .txt format.

Annotations

Annotations can be exported in stand-off XML or UIMA XMI.

WAT-SL

Markables

Label-boxes

Not supported.

Highlights

Single highlight

Default

PHYSICIAN HOSPITAL DISCHARGE SUMMARY 

Provider: Ken Cure, MD 

Highlighted

PHYSICIAN HOSPITAL DISCHARGE SUMMARY 

Provider: Ken Cure, MD  segment 0

Clicked

Clicks only register on the symbol right to the segment/span/highlight.

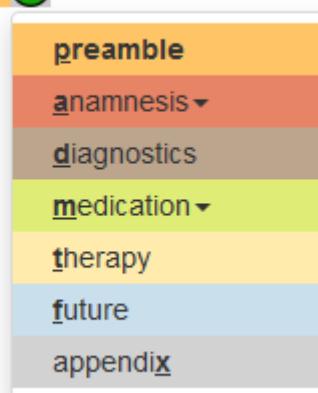
PHYSICIAN HOSPITAL DISCHARGE SUMMARY 

Provider: Ken Cure, MD 

Patient: Patient H Sample 

Provider's Pt ID: 6910828 Sex: Female 

Attachment Control Number: XA728302 



Multi highlight

Not supported.

Relations

Not supported.

Data formats

<https://www.aclweb.org/anthology/E17-3004.pdf>

Documents

Documents are stored in plaintext format.

Annotations

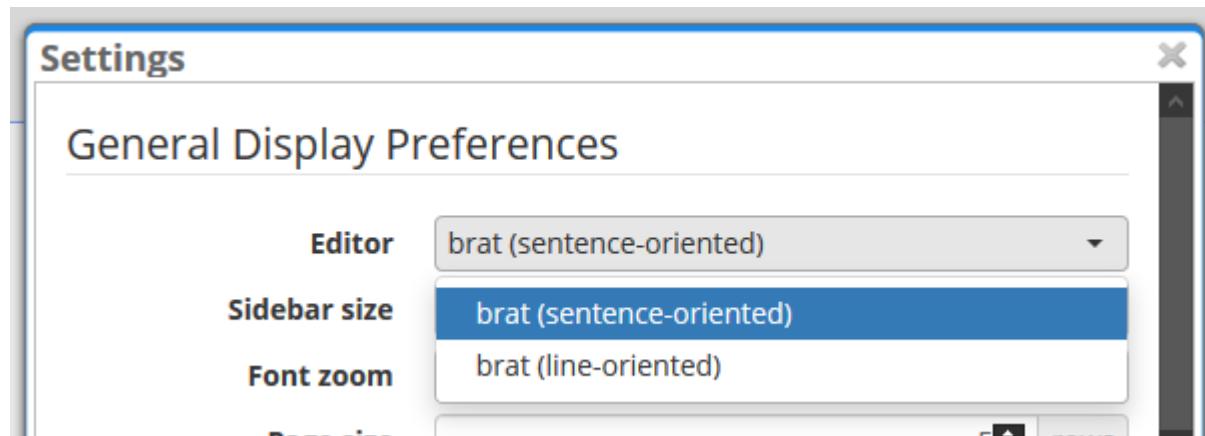
Annotations can be exported in CSV or .ann format.

WebAnno

Important note

WebAnno seems to allow customization of the editor itself.

By default it appears to utilize a brat-based editor:



As a result, many of the behaviours of WebAnno are identical to brat.

We will only document cases in which WebAnno differs from brat.

Markables

Implementation

Source of the following two screenshots and quotes:

https://webanno.github.io/webanno/releases/3.4.5/docs/user-guide.html#sect_formats

WebAnno supports various formats.

It appears as if the WebAnno TSV 3.2 File Format is the default format.

https://webanno.github.io/webanno/releases/3.4.5/docs/user-guide.html#sect_webannotsv

Example: Token position

1-2	4-8	Haag
-----	-----	------

“Token annotation starts with a sentence-token number marker followed by the begin-end offsets and the token itself, separated by a TAB characters.”

Example: Sub-token positions

1-3	9-14	plays
1-3.1	9-13	play
1-3.2	13-14	s

“Sub-token representations are affixed with a . and a number starts from 1 to N.”

Label boxes

Single label

Default

Identical to brat.

Highlighted

Identical to brat.

Double-clicked

The screenshot shows a text editor with the following content:

example document I created.

1d Anna met in a park.

2d friends sometimes call him Otti .

Annotations:

- Annotation for "Anna": Layer: Named entity, Text: Anna, value: (nickname)
- Annotation for "Otti": Layer: Named entity, Text: Otti, value: (nickname)
- Annotations for "met in a park": Layer: Chunk, Text: (Chu), value: (InFTec)
- Annotations for "friends sometimes call him": Layer: Chunk, Text: (Chunk), value: (Sur)
- Annotations for "Otti": Layer: Chunk, Text: (Chunk), value: (Mus)

Right sidebar:

- Layer:** Named entity
- Annotation:** Delete Clear
- Text:** Anna
- value:** (nickname)

Multi label

Default

Identical to brat.

Highlighted

Identical to brat.

Double-clicked

The screenshot shows a text editor with the following content:

1 This is an example document I created.

2 Otto and Anna met in a park.

3 Otto's friends sometimes call him Otti .

4 Otto works in IT , is a surgeon and composes music in his freetime.

Annotations:

- Annotation for "Otto": Layer: Named entity, Text: Otto, value: (name)
- Annotation for "Anna": Layer: Named entity, Text: Anna, value: (name)
- Annotation for "Otti": Layer: Named entity, Text: Otti, value: (name)
- Annotations for "met in a park": Layer: Chunk, Text: (Chu), value: (InFTec)
- Annotations for "friends sometimes call him": Layer: Chunk, Text: (Chunk), value: (Sur)
- Annotations for "works in IT": Layer: Chunk, Text: (Chu), value: (InFTec)
- Annotations for "surgeon": Layer: Chunk, Text: (Chunk), value: (Sur)
- Annotations for "composes music": Layer: Chunk, Text: (Chunk), value: (Mus)

Right sidebar:

- Layer:** Information Technician
- Annotation:** Delete Clear
- Text:** Otto
- value:** (name)

No features available!

Highlights

Identical to brat.

Relations

Directed

Single-label to single-label

Default

Identical to brat.

Highlighted

Identical to brat.

Double-clicked



Multi-label to multi-label

Default

Identical to brat.

Highlighted

Identical to brat.

Double-clicked

Identical to “single-label to single-label” behaviour.

Undirected

Note

WebAnno doesn't have undirected relations, instead it has “chains”.

Chains seem to have some extra settings which relations do not have.

Unfortunately, the usage of chains turned out to be complicated to a degree that we weren't able to create custom chains, and even the attempt of recreating an existing chain relation that comes shipped with WebAnno by default (Coreference) failed.

For this reason we have to refer to the official documentation in regards to chain behaviour:

https://webanno.github.io/webanno/releases/3.6.5/docs/user-guide.html#_chains

Table 2. Chain behavior

Linked List	Condition	Result
disabled	the two spans are already in the same chain	nothing happens
disabled	the two spans are in different chains	the two chains are merged
enabled	the two spans are already in the same chains	the chain will be re-linked such that a chain link points from the source to the target span, potentially creating new chains in the process.
enabled	the two spans are in different chains	the chains will be re-linked such that a chain link points from the source to the target span, merging the two chains and potentially creating new chains from the remaining prefix and suffix of the original chains.

Source of screenshot: WebAnno documentation

Creation

Directed

Identical to brat.

Undirected

Unable to reproduce.

Data formats

Source of the following screenshot:

https://webanno.github.io/webanno/releases/3.4.5/docs/user-guide.html#sect_formats

Documents

Documents can be imported in a variety of formats.

Table 18. Supported annotation formats

Format	Read	Write	Custom Layers	Description
CoNLL 2000	yes	yes	no	POS, chunks
CoNLL 2002	yes	yes	no	Named entities
CoNLL 2006	yes	yes	no	Lemma, POS, dependencies (basic)
CoNLL 2009	yes	yes	no	Lemma, POS, dependencies (basic)
CoNLL-U	yes	yes	no	Lemma, POS, dependencies (basic & enhanced), surface form
Plain text	yes	yes	no	No annotations
TCF	yes	no	no	Lemma, POS, dependencies (basic), coreference, named entities
TEI CPH dialect	yes	no	no	
WebAnno TSV 1	yes	no	no	
WebAnno TSV 2	yes	no	yes	token, multiple token, and arc annotations supported. No chain annotation is supported. no sub-token annotation is supported
WebAnno TSV 3	yes	yes	yes	
Binary	yes	yes	yes	UIMA Binary CAS
XMI	yes	yes	yes	UIMA XMI CAS

Annotations

Annotations can be exported in a variety of formats

YEDDA

Markables

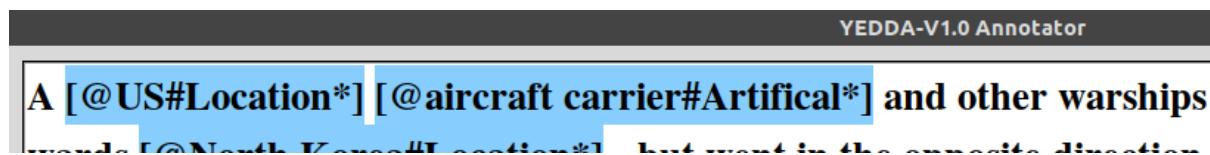
Label-boxes

Not supported.

Highlights

Single highlight

Default



Highlighted

Not supported.

Clicked

Not supported.

Multi highlight

Not supported.

Relations

Not supported.

Data formats

<https://arxiv.org/pdf/1711.03759.pdf>

Documents

Documents are stored in plaintext .txt format.

Annotations

Annotations can be exported in stand-off .ann format.

Table of contents

[Table of contents](#)

[Installation environment](#)

[Disclaimer](#)

[brat](#)

[Github](#)

[Docker container](#)

[Installation documentation](#)

[Standalone installation](#)

[Docker](#)

[Short instructions](#)

[Cora](#)

[Installation documentation \(failed\)](#)

[doccoano](#)

[Docker container](#)

[Installation documentation](#)

[FoLiA/flat](#)

[Docker container](#)

[Installation documentation](#)

[INCEpTION](#)

[Docker container](#)

[Installation documentation](#)

[MMAX2](#)

[Docker container](#)

[Installation documentation](#)

[Swan](#)

[Docker container](#)

[Installation documentation](#)

[WAT-SL](#)

[Docker container](#)

[Installation documentation](#)

[WebAnno](#)

[Docker container](#)

[Installation documentation](#)

[YEDDA](#)

[Docker container](#)
[Installation documentation](#)

Installation environment

We installed the annotation software on a desktop computer running Ubuntu 18.04 inside a virtual machine.

In cases where installation failed, we retried installation on a laptop running Ubuntu 16.04 natively.

Docker & Docker Compose are installed in both environments.

In cases where the installation failed in both environments, we tried to identify if a webdemo of the software was available. If this also failed, we stopped testing the software.

If a docker container of the software was available, we judged whether installation with a docker container would be easier than the official installation process and decided which installation to use on a case-by-case basis.

Disclaimer

We documented the installation process as a mix of comments, screenshots and a protocol of terminal commands (indicated by lines starting with "\$"), following the official installation instructions at the time of testing.

This file is included as an appendix to the thesis for the sake of completeness, and to document solutions to installation issues we encountered during testing.

This installation documentation is supposed to be understood as complementary reading to official documentation and attempt to support reproducibility.

We strongly encourage the use of the official and up-to-date documentation over static third party instructions.

brat

Github

<https://github.com/nlplab/brat>

Docker container

Unofficial:

<https://github.com/ddevaraj/docker-brat>

Visit the URL for installation instructions by the maintainer of the docker container.

Installation documentation

Standalone installation

Docker

Short instructions

1. Clone the git repository <https://github.com/ddevaraj/docker-brat>
2. \$ cd docker-brat
 - Brat login requires username, password and admin email to be mentioned by the user. These arguments are specified in the dockerfile. User is required to provide the values while building the dockerfile.
- 3.
4. \$ docker build --build-arg username=<Username> --build-arg password=<Password> --build-arg admin_email=<email ID> -t brat .
5. \$ docker run --name brat_instance -p 80:80 -d brat
6. Open the browser and enter the address :
http://localhost:80/brat-v1.3_Crunchy_Frog/

Cora

<https://github.com/comphist/cora>

Installation documentation (failed)

1. Download prebuilt from <https://github.com/comphist/cora/releases>
2. Extract
3. Install Apache:
4. \$ sudo apt update
5. \$ sudo apt install apache2
6. Note: if Cora is supposed to be accessed from a different computer, firewall rules have to be edited for apache - since we intend to use Cora via localhost it isn't necessary in this case.
7. Edit permissions for apache document folder:
8. \$ sudo chmod o+w /var/www/html
9. Copy contents of 'www' folder from the cora release into the /var/www/html folder
10. Install PHP (5.5+ or 7+, We'll be using version 7.2):
11. \$ sudo apt install php libapache2-mod-php
12. \$ sudo apt-get install php-mysql
13. Install PHP extensions:
14. \$ sudo apt-get install php-xml
15. Note: php-xml contains the needed dom extension
16. \$ sudo apt-get install php-json
17. Note: libxml is included in php 7 by default

18. Restart apache:
19. \$ sudo service apache2 restart
20. Install MySQL:
21. \$ sudo apt-get install mysql-server
22. Open <localhost:port>/db/configure_db.php in your browser

The installation “succeeded” insofar that we were able to open Cora in the browser. Unfortunately we weren’t able to log into the software using the default admin:admin user/password combination and were thus unable to use the software for annotation. This behaviour was observed in both installation environments. We believe that this issue might be due to a fault during the MySQL installation. However, we were unable to determine the exact problem.

doccano

Docker container

official:

<https://github.com/dooccano/dooccano>

Installation documentation

The installation of doccano via docker was initially unsuccessful. Thankfully we were able to identify the issue and managed to install the software after some troubleshooting. This documentation will include the steps of the failed installation attempt as strike-through text to help anyone who might run into the same issue.

1. Clone the git repository
2. \$ cd doccano
3. ~~\$ docker build -t dooccanoimage .~~
4. ~~Wait quite a bit~~
5. ~~Result: “Successfully tagged dooccanoimage:latest”~~
6. ~~\$ docker images~~
~~(to check if image exists)~~
7. ~~\$ docker run dooccanoimage~~

For production environment:

3. \$ docker-compose -f docker-compose.prod.yml up

Error:

```
vincent@UbuntuDesktop:~/Documents/annotation/dooccano/dooccano$ docker-compose -f docker-compose.prod.yml up
ERROR: Version in "./docker-compose.prod.yml" is unsupported. You might be seeing this error
because you're using the wrong Compose file version. Either specify a supported version (e.g "2.2" or "3.3") and place your service definitions under the `services` key, or omit the `version` key and place your service definitions at the root of the file to use version 1.
For more on the Compose file format versions, see https://docs.docker.com/compose/compose-file/
```

Fix:

<https://github.com/bigbluebutton/greenlight/issues/228#issuecomment-545919537>

grtlinux commented on 24 Oct 2019

ref: 10up/wp-local-docker#58 (comment)

I had faced the same issue on ubuntu vm.
I did the following steps:

```
$ sudo apt-get remove docker-compose
$ sudo curl -L "https://github.com/docker/compose/releases/download/1.23.2/docker-compose-$(uname -s)-$(uname -m)" -o
/usr/local/bin/docker-compose
$ sudo chmod +x /usr/local/bin/docker-compose
$ sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

I hope this will resolve this issue.
Mine is working

Error:

```
Successfully built 8c47a64f2129
Successfully tagged dooccano_nginx:latest
WARNING: Image for service nginx was built because it did not already exist. To rebuild this image you must use `docker-compose build` or `docker-compose up --build`.
Creating dooccano_frontend_1 ... done
Creating dooccano_postgres_1 ... done
Creating dooccano_backend_1 ... done
Creating dooccano_nginx_1 ... error

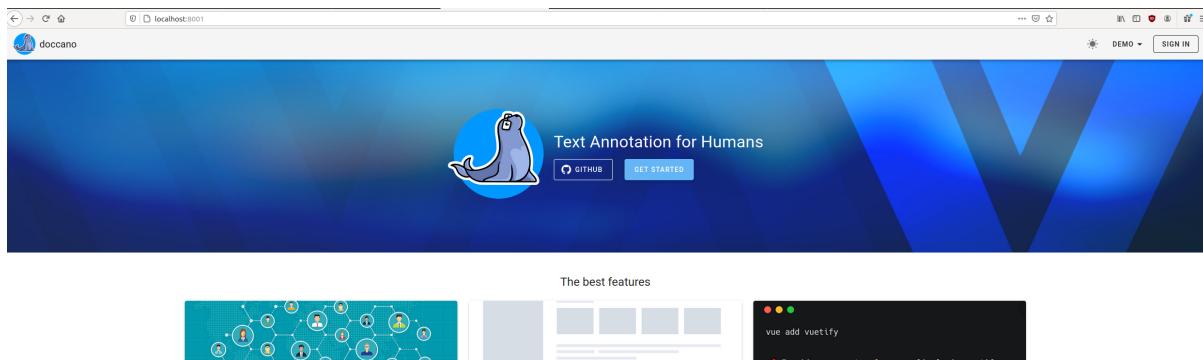
ERROR: for dooccano_nginx_1 Cannot start service nginx: driver failed programming external connectivity on endpoint dooccano_nginx_1 (6c12011d0067d7b57a49934f35d07d4d245521bf25cf81911c0afdc88085df7c): Error starting userland proxy: listen tcp 0.0.0.0:80: bind: address already in use
ERROR: for nginx Cannot start service nginx: driver failed programming external connectivity on endpoint dooccano_nginx_1 (6c12011d0067d7b57a49934f35d07d4d245521bf25cf81911c0afdc88085df7c): Error starting userland proxy: listen tcp 0.0.0.0:80: bind: address already in use
ERROR: Encountered errors while bringing up the project.
```

Fix:

change the ports in the compose file (docker-compose.prod.yml):

```
nginx:
  build: ./nginx
  volumes:
    - www:/var/www/html:ro
    - static_volume:/static
  ports:
    - 8001:80
  depends_on:
    - backend
  networks:
    - network-frontend
```

4. Open dooccano via localhost:port (port being the port designated in the file)



FoLiA/flat

Docker container

No official or unofficial docker container of FLAT exists as of time of writing.

We instead suggest the usage of [LaMachine](#), which attempts to increase the ease of use for various NLP tools, FLAT and foliadocserve included. LaMachine appears to be actively maintained and comes with a more detailed documentation.

We unfortunately weren't aware of LaMachine during the testing phase of this thesis, and were thus unable to use it and document the installation process.

Installation documentation

https://flat.readthedocs.io/en/latest/installation_guide.html

1. \$ mkdir FoLiA
2. \$ cd FoLiA
3. \$ virtualenv --python=python3 env
 - a. if virtualenv isn't installed yet: \$ sudo apt install virtualenv
 - b. repeat step 3
4. \$. env/bin/activate
5. \$ pip install FoLiA-Linguistic-Annotation-Tool
 - a. Error:

```

vincent@UbuntuDesktop:~/Documents/annotation/FoLiA-Linguistic-Annotation-Tool$ pip install FoLiA-Linguistic-Annotation-Tool
Collecting FoLiA-Linguistic-Annotation-Tool
  Downloading https://files.pythonhosted.org/packages/34/69/e31ad3c5d937ee373e3ab0c04a61f754b10a913989e51d53279ae04dfe4b7/FoLiA-Linguistic-Annotation-Tool-0.9.1.tar.gz (321kB)
    100% |████████████████████████████████| 321kB 4.9MB/s
Collecting folia==2.2.1 (from FoLiA-Linguistic-Annotation-Tool)
  Downloading https://files.pythonhosted.org/packages/bb/55/f6b672c19b50915a59176900dc462ab0491e7194fad2eb566d701fc6482/FoLiA-2.3.0.tar.gz (173kB)
    100% |████████████████████████████████| 173kB 4.9MB/s
  Complete output from command python setup.py egg_info:
  Traceback (most recent call last):
    File "<string>", line 1, in <module>
      File "/tmp/pip-build-BUnuFv/folia/setup.py", line 34, in <module>
        long_description=read('README.rst'),
      File "/tmp/pip-build-BUnuFv/folia/setup.py", line 19, in read
        return open(os.path.join(os.path.dirname(__file__), fname),'r',encoding='utf-8').read()
  TypeError: 'encoding' is an invalid keyword argument for this function

  -----
  Command "python setup.py egg_info" failed with error code 1 in /tmp/pip-build-BUnuFv/folia/

```

b. `Command "python setup.py egg_info" failed with error code 1 in /tmp/pip-build-BUnuFv/folia/`

c. Fix 1:

d. `$ sudo pip install --upgrade setuptools`

e. If that didn't work

f. Fix 2:

g. `$ pip install --upgrade pip`

6. Continue with flat configuration from here:

https://flat.readthedocs.io/en/latest/installation_guide.html#flat-configuration

7. Download settings.py from

<https://raw.githubusercontent.com/proycon/flat/master/settings.py>

8. install sqlite3

a. `$ sudo apt-get install sqlite3`

9. `$ sqlite3 Database.db`

a. `$.databases`

b. Copy path to database and enter into settings.py

c. `$.quit`

10. Copy path to document folder into settings.py

11. Configure the rest of of settings.py for your specific needs

12. Once done, comment out this part of settings.py:

```

#####
# PROTECTION AGAINST LAZINESS, GENERAL SLOPPINESS, OVERCONFIDENCE, AND DOCUMENTATION READING IMPAIRMENTS
#####

```

```

#Remove this when done configuring:
raise Exception("settings.py hasn't been configured yet!!")

```

13.

14. Copy path to folder containing settings.py

15. `$ export PYTHONPATH=/your/settings/path/`

16. `$ export DJANGO_SETTINGS_MODULE=settings`

17. `$ django-admin migrate --run-syncdb`

a. Ff django isn't installed, install via

b. `$ sudo apt-get install python-django`

c. If django-admin isn't installed, install via

d. `$ sudo apt install python-django-common`

e. You might get this error:

```

vincent@UbuntuDesktop:~/Documents/annotation/FoLiA$ django-admin migrate --run-syncdb
Traceback (most recent call last):
  File "/usr/bin/django-admin", line 21, in <module>
    management.execute_from_command_line()
  File "/usr/lib/python2.7/dist-packages/django/core/management/__init__.py", line 364, in execute_from_command_line
    utility.execute()
  File "/usr/lib/python2.7/dist-packages/django/core/management/__init__.py", line 308, in execute
    settings.INSTALLED_APPS
  File "/usr/lib/python2.7/dist-packages/django/conf/__init__.py", line 56, in __getattr__
    self._setup(name)
  File "/usr/lib/python2.7/dist-packages/django/conf/__init__.py", line 41, in _setup
    self._wrapped = Settings(settings_module)
  File "/usr/lib/python2.7/dist-packages/django/conf/__init__.py", line 110, in __init__
    mod = importlib.import_module(self.SETTINGS_MODULE)
  File "/usr/lib/python2.7/importlib/_init__.py", line 37, in import_module
    __import__(name)
  File "/home/vincent/Documents/annotation/FoLiA/settings.py", line 13, in <module>
    import flat
f. ImportError: No module named flat

```

- g. This happens if you accidentally left your python3 environment, re-enter it via
- h. \$. env/bin/activate
- i. Repeat

Foliadocserve is also needed to run FLAT, something that the documentation fails to mention

1. \$. env/bin/activate
2. \$ pip install foliadocserve

To run FLAT:

```

$ . env/bin/activate
$ export PYTHONPATH=/your/settings/path/
$ export DJANGO_SETTINGS_MODULE=settings
$ django-admin runserver

```

In a second terminal:

```

$ . env/bin/activate
$ foliadocserve -d /path/to/document/root -p 8080 --git

```

INCEpTION

Docker container

Since INCEpTION is delivered as a java standalone file, packing it into a Docker container would make the installation more complicated rather than simpler.

Installation documentation

<https://inception-project.github.io/downloads/>

1. Download INCEpTION
2. Start .jar by double-clicking
3. Run via localhost:8080

MMAX2

Docker container

The installation is very straightforward, packing it into a Docker container would make the installation more complicated rather than simpler.

Installation documentation

1. \$ git clone <https://github.com/nlpAThits/MMAX2>
2. \$ cd MMAX2
3. \$ chmod +x ./mmax2.sh
4. ./mmax2.sh

Swan

Docker container

Official (broken):

<https://github.com/annefried/swan>

Installation documentation

The installation failed, the dockerfile seems to contain errors (broken dependency).

Attempted installation steps:

1. Clone the git repository
2. cd into swan directory
3. Try to build dockerfile

\$ docker build -t swanimage .

Explanation: -t names the image on successful build, in this case “swanimage”, “.” is the current directory, where docker daemon will look for the dockerfile.

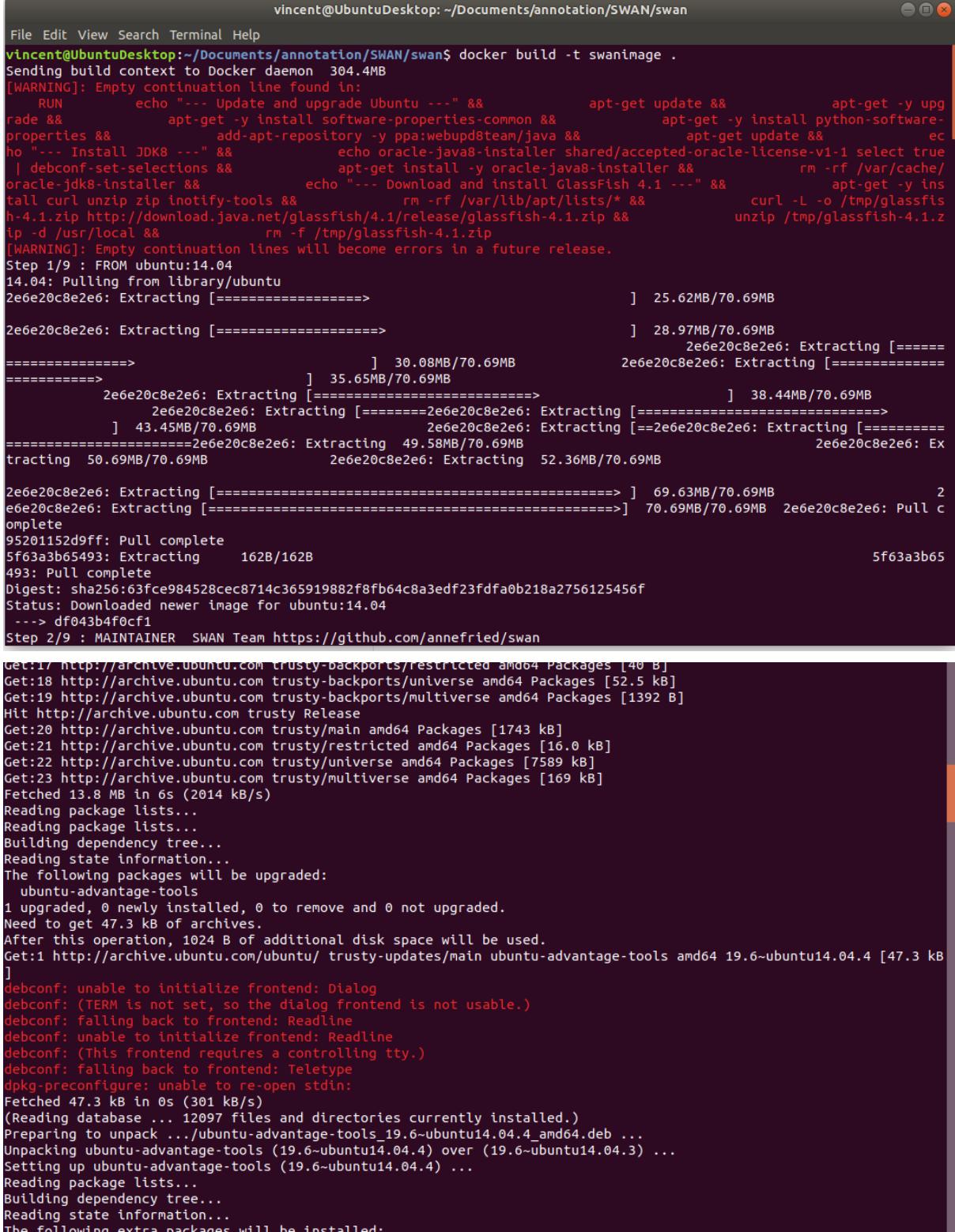
Error:

Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post http://%2Fvar%2Frun%2Fdocker.sock/v1.40/build?buildargs=%7B%7D&cachefrom=%5B%5D&cgroupparent=&cpuperiod=0&cpuquota=0&cpusetcpus=&cpusetmems=&cpushares=0&dockerfile=Dockerfile&labels=%7B%7D&memory=0&memswap=0&networkmode=default&rm=1&session=p31spqkvzpycsbtlsduco6fi&shmsize=0&t=swanimage&target=&ulimits=null&version=1: dial unix /var/run/docker.sock: connect: permission denied

Fix: This means we accidentally skipped this step in the installation:

<https://docs.docker.com/engine/install/linux-postinstall/>

Warning during installation:



The terminal window shows the following Docker build command and its output:

```
vincent@UbuntuDesktop:~/Documents/annotation/SWAN/swan$ docker build -t swanimage .
Sending build context to Docker daemon 304.4MB
[WARNING]: Empty continuation line found in:
    RUN      echo "--- Update and upgrade Ubuntu ---" &&          apt-get update &&          apt-get -y upg
rade &&      apt-get -y install software-properties-common &&          apt-get -y install python-software-
properties &&      add-apt-repository -y ppa:webupd8team/java &&          apt-get update &&          ec
ho --- Install JDK8 --- &&      echo oracle-java8-installer shared/accepted-oracle-license-v1-1 select true
| debconf-set-selections &&      apt-get install -y oracle-java8-installer &&          rm -rf /var/cache/
oracle-jdk8-installer &&      echo "--- Download and install GlassFish 4.1 ---" &&          apt-get -y ins
tall curl unzip zip inotify-tools &&      rm -rf /var/lib/apt/lists/* &&          curl -L -o /tmp/glassfis
h-4.1.zip http://download.java.net/glassfish/4.1/release/glassfish-4.1.zip &&          unzip /tmp/glassfish-4.1.z
ip -d /usr/local &&      rm -f /tmp/glassfish-4.1.zip
[WARNING]: Empty continuation lines will become errors in a future release.
Step 1/9 : FROM ubuntu:14.04
14.04: Pulling from library/ubuntu
2e6e20c8e2e6: Extracting [=====] 25.62MB/70.69MB
2e6e20c8e2e6: Extracting [=====] 28.97MB/70.69MB
2e6e20c8e2e6: Extracting [=====] 30.08MB/70.69MB
2e6e20c8e2e6: Extracting [=====] 35.65MB/70.69MB
2e6e20c8e2e6: Extracting [=====] 43.45MB/70.69MB
2e6e20c8e2e6: Extracting [=====] 49.58MB/70.69MB
2e6e20c8e2e6: Extracting [=====] 52.36MB/70.69MB
2e6e20c8e2e6: Extracting [=====] 69.63MB/70.69MB
2e6e20c8e2e6: Extracting [=====] 70.69MB/70.69MB
2e6e20c8e2e6: Pull complete
5f63a3b65493: Pull complete
Digest: sha256:63fce984528cec8714c365919882f8fb64c8a3edf23fd8a0b218a2756125456f
Status: Downloaded newer image for ubuntu:14.04
--> df043b4f0cf1
Step 2/9 : MAINTAINER SWAN Team https://github.com/annefried/swan
```

Below the Docker logs, an apt-get update command is shown:

```
Get:1 http://archive.ubuntu.com trusty-backports/restricted amd64 Packages [40 B]
Get:18 http://archive.ubuntu.com trusty-backports/universe amd64 Packages [52.5 kB]
Get:19 http://archive.ubuntu.com trusty-backports/multiverse amd64 Packages [1392 B]
Hit http://archive.ubuntu.com trusty Release
Get:20 http://archive.ubuntu.com trusty/main amd64 Packages [1743 kB]
Get:21 http://archive.ubuntu.com trusty/restricted amd64 Packages [16.0 kB]
Get:22 http://archive.ubuntu.com trusty/universe amd64 Packages [7589 kB]
Get:23 http://archive.ubuntu.com trusty/multiverse amd64 Packages [169 kB]
Fetched 13.8 MB in 6s (2014 kB/s)
Reading package lists...
Reading package lists...
Building dependency tree...
Reading state information...
The following packages will be upgraded:
  ubuntu-adantage-tools
1 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 47.3 kB of archives.
After this operation, 1024 B of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu/ trusty-updates/main ubuntu-adantage-tools amd64 19.6~ubuntu14.04.4 [47.3 kB]
debconf: unable to initialize frontend: Dialog
debconf: (TERM is not set, so the dialog frontend is not usable.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Fetched 47.3 kB in 0s (301 kB/s)
(Reading database ... 12097 files and directories currently installed.)
Preparing to unpack .../ubuntu-adantage-tools_19.6~ubuntu14.04.4_amd64.deb ...
Unpacking ubuntu-adantage-tools (19.6~ubuntu14.04.4) over (19.6~ubuntu14.04.3) ...
Setting up ubuntu-adantage-tools (19.6~ubuntu14.04.4) ...
Reading package lists...
Building dependency tree...
Reading state information...
The following extra packages will be installed.
```

```

Need to get 5126 kB of archives.
After this operation, 30.8 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu/ trusty-updates/main libglib2.0-0 amd64 2.40.2-0ubuntu1.1 [1059 kB]
Get:2 http://archive.ubuntu.com/ubuntu/ trusty/main libdbus-glib-1-2 amd64 0.100.2-1 [74.1 kB]
Get:3 http://archive.ubuntu.com/ubuntu/ trusty-updates/main libxml2 amd64 2.9.1+dfsg1-3ubuntu4.13 [573 kB]
Get:4 http://archive.ubuntu.com/ubuntu/ trusty/main sgml-base all 1.26+nmu4ubuntu1 [12.5 kB]
Get:5 http://archive.ubuntu.com/ubuntu/ trusty-updates/main libgirepository-1.0-1 amd64 1.40.0-1ubuntu0.2 [85.6 kB]
Get:6 http://archive.ubuntu.com/ubuntu/ trusty-updates/main gir1.2-glib-2.0 amd64 1.40.0-1ubuntu0.2 [124 kB]
Get:7 http://archive.ubuntu.com/ubuntu/ trusty/main iso-codes all 3.52-1 [2073 kB]
Get:8 http://archive.ubuntu.com/ubuntu/ trusty-updates/main libglib2.0-data all 2.40.2-0ubuntu1.1 [116 kB]
Get:9 http://archive.ubuntu.com/ubuntu/ trusty-updates/main python-apt-common all 0.9.3.5ubuntu3 [15.0 kB]
Get:10 http://archive.ubuntu.com/ubuntu/ trusty-updates/main python3-apt amd64 0.9.3.5ubuntu3 [139 kB]
Get:11 http://archive.ubuntu.com/ubuntu/ trusty/main python3-dbus amd64 1.2.0-2build2 [82.1 kB]
Get:12 http://archive.ubuntu.com/ubuntu/ trusty-updates/main python3-gi amd64 3.12.0-1ubuntu1 [154 kB]
Get:13 http://archive.ubuntu.com/ubuntu/ trusty/main shared-mime-info amd64 1.2-0ubuntu3 [415 kB]
Get:14 http://archive.ubuntu.com/ubuntu/ trusty/main xml-core all 0.13+nmu2 [23.3 kB]
Get:15 http://archive.ubuntu.com/ubuntu/ trusty/main python3-pycurl amd64 7.19.3-0ubuntu3 [47.5 kB]
Get:16 http://archive.ubuntu.com/ubuntu/ trusty/main xz-utils amd64 5.1.1alpha+20120614-2ubuntu2 [78.8 kB]
Get:17 http://archive.ubuntu.com/ubuntu/ trusty-updates/main unattended-upgrades all 0.82.1ubuntu2.5 [25.6 kB]
Get:18 http://archive.ubuntu.com/ubuntu/ trusty-updates/main python3-software-properties all 0.92.37.8 [19.2 kB]
Get:19 http://archive.ubuntu.com/ubuntu/ trusty-updates/main software-properties-common all 0.92.37.8 [9384 B]
debconf: unable to initialize frontend: Dialog
debconf: (TERM is not set, so the dialog frontend is not usable.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Fetched 5126 kB in 3s (1624 kB/s)

Setting up libglib2.0-0:amd64 (2.40.2-0ubuntu1.1) ...
No schema files found: doing nothing.
Setting up libdbus-glib-1-2:amd64 (0.100.2-1) ...
Setting up libxml2:amd64 (2.9.1+dfsg1-3ubuntu4.13) ...
Setting up sgml-base (1.26+nmu4ubuntu1) ...
Setting up libgirepository-1.0-1 (1.40.0-1ubuntu0.2) ...
Setting up gir1.2-glib-2.0 (1.40.0-1ubuntu0.2) ...
Setting up iso-codes (3.52-1) ...
Setting up libglib2.0-data (2.40.2-0ubuntu1.1) ...
Setting up python-apt-common (0.9.3.5ubuntu3) ...
Setting up python3-apt (0.9.3.5ubuntu3) ...
Setting up python3-dbus (1.2.0-2build2) ...
Setting up python3-gi (3.12.0-1ubuntu1) ...
Setting up shared-mime-info (1.2-0ubuntu3) ...
Setting up xml-core (0.13+nmu2) ...
Setting up python3-pycurl (7.19.3-0ubuntu3) ...
Setting up xz-utils (5.1.1alpha+20120614-2ubuntu2) ...
update-alternatives: using /usr/bin/xz to provide /usr/bin/lzma (lzma) in auto mode
Setting up unattended-upgrades (0.82.1ubuntu2.5) ...
debconf: unable to initialize frontend: Dialog
debconf: (TERM is not set, so the dialog frontend is not usable.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
Processing triggers for ureadahead (0.100.0-16) ...
Setting up python3-software-properties (0.92.37.8) ...
Setting up software-properties-common (0.92.37.8) ...
Processing triggers for libc-bin (2.19-0ubuntu6.15) ...
Processing triggers for sgml-base (1.26+nmu4ubuntu1) ...
Reading package lists...
Building dependency tree...
Reading state information...

```

```

Suggested packages:
  python-doc python-tk python-apt-dbg python-gtk2 python-vte python-apt-doc
  libcurl4-gnutls-dev python-pycurl-dbg python2.7-doc binutils binfmt-support
The following NEW packages will be installed:
  libpython-stdlib libpython2.7-minimal libpython2.7-stdlib python python-apt
  python-minimal python-pycurl python-software-properties python2.7
  python2.7-minimal
0 upgraded, 10 newly installed, 0 to remove and 0 not upgraded.
Need to get 3940 kB of archives.
After this operation, 17.0 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu/ trusty-updates/main libpython2.7-minimal amd64 2.7.6-8ubuntu0.5 [308 kB]
Get:2 http://archive.ubuntu.com/ubuntu/ trusty-updates/main python2.7-minimal amd64 2.7.6-8ubuntu0.5 [1186 kB]
Get:3 http://archive.ubuntu.com/ubuntu/ trusty-updates/main libpython2.7-stdlib amd64 2.7.6-8ubuntu0.5 [1872 kB]
Get:4 http://archive.ubuntu.com/ubuntu/ trusty/main libpython-stdlib amd64 2.7.5-5ubuntu3 [7012 B]
Get:5 http://archive.ubuntu.com/ubuntu/ trusty-updates/main python2.7 amd64 2.7.6-8ubuntu0.5 [197 kB]
Get:6 http://archive.ubuntu.com/ubuntu/ trusty/main python-minimal amd64 2.7.5-5ubuntu3 [27.5 kB]
Get:7 http://archive.ubuntu.com/ubuntu/ trusty/main python amd64 2.7.5-5ubuntu3 [134 kB]
Get:8 http://archive.ubuntu.com/ubuntu/ trusty-updates/main python-apt amd64 0.9.3.5ubuntu3 [141 kB]
Get:9 http://archive.ubuntu.com/ubuntu/ trusty/main python-pycurl amd64 7.19.3-0ubuntu3 [47.9 kB]
Get:10 http://archive.ubuntu.com/ubuntu/ trusty-updates/universe python-software-properties all 0.92.37.8 [19.7 kB]
debconf: unable to initialize frontend: Dialog
debconf: (TERM is not set, so the dialog frontend is not usable.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Fetched 3940 kB in 2s (1560 kB/s)
Selecting previously unselected package libpython2.7-minimal:amd64.
(Reading database ... 13007 files and directories currently installed.)
Preparing to unpack .../libpython2.7-minimal_2.7.6-8ubuntu0.5_amd64.deb ...
Unpacking libpython2.7-minimal:amd64 (2.7.6-8ubuntu0.5) ...
Selecting previously unselected package python-software-properties.
Preparing to unpack .../python-software-properties_0.92.37.8_all.deb ...
Unpacking python-software-properties (0.92.37.8) ...
Processing triggers for mime-support (3.54ubuntu1.1) ...
Setting up libpython2.7-minimal:amd64 (2.7.6-8ubuntu0.5) ...
Setting up python2.7-minimal (2.7.6-8ubuntu0.5) ...
Linking and byte-compiling packages for runtime python2.7...
Setting up libpython2.7-stdlib:amd64 (2.7.6-8ubuntu0.5) ...
Setting up libpython-stdlib:amd64 (2.7.5-5ubuntu3) ...
Setting up python2.7 (2.7.6-8ubuntu0.5) ...
Setting up python-minimal (2.7.5-5ubuntu3) ...
Setting up python (2.7.5-5ubuntu3) ...
Setting up python-apt (0.9.3.5ubuntu3) ...
Setting up python-pycurl (7.19.3-0ubuntu3) ...
Setting up python-software-properties (0.92.37.8) ...
gpg: keyring '/tmp/tmpf22kv_dm/secrec.gpg' created
gpg: keyring '/tmp/tmpf22kv_dm/pubring.gpg' created
gpg: requesting key EEA14886 from hkp server keyserver.ubuntu.com
gpg: /tmp/tmpf22kv_dm/trustdb.gpg: trustdb created
gpg: key EEA14886: public key "Launchpad VLC" imported
gpg: no ultimately trusted keys found
gpg: Total number processed: 1
gpg:           imported: 1  (RSA: 1)
OK
Ign http://archive.ubuntu.com trusty InRelease
Get:1 http://ppa.launchpad.net trusty InRelease [15.5 kB]
Hit http://archive.ubuntu.com trusty-updates InRelease
Hit http://archive.ubuntu.com trusty-backports InRelease
Hit http://archive.ubuntu.com trusty Release.gpg
Hit http://archive.ubuntu.com trusty-updates/main amd64 Packages
Get:2 http://ppa.launchpad.net trusty/main amd64 Packages [20 B]
Hit http://security.ubuntu.com trusty-security InRelease
Hit http://archive.ubuntu.com trusty-updates/restricted amd64 Packages
Hit http://archive.ubuntu.com trusty-updates/universe amd64 Packages
Hit http://archive.ubuntu.com trusty-updates/multiverse amd64 Packages
Hit http://archive.ubuntu.com trusty-backports/main amd64 Packages
Hit http://security.ubuntu.com trusty-security/main amd64 Packages

```

```
Hit http://archive.ubuntu.com trusty/universe amd64 Packages
Hit http://security.ubuntu.com trusty-security/universe amd64 Packages
Hit http://archive.ubuntu.com trusty/multiverse amd64 Packages
Hit https://esm.ubuntu.com trusty-infra-security InRelease
Hit http://security.ubuntu.com trusty-security/multiverse amd64 Packages
Hit https://esm.ubuntu.com trusty-infra-updates InRelease
Hit https://esm.ubuntu.com trusty-infra-security/main amd64 Packages
Hit https://esm.ubuntu.com trusty-infra-updates/main amd64 Packages
Fetched 15.5 kB in 1s (8042 B/s)
Reading package lists...
--- Install JDK8 ---
Reading package lists...
Building dependency tree...
Reading state information...
E: Unable to locate package oracle-java8-installer
The command '/bin/sh -c echo "--- Update and upgrade Ubuntu ---" && apt-get update && apt-get upgrade && apt-get -y install software-properties-common && apt-get update && apt-get update && apt-get -y install python-software-properties && add-apt-repository -y ppa:webupd8team/java && apt-get update && echo "--- Install JDK8 ---" && echo oracle-java8-installer shared/accepted-oracle-license-v1-select true | debconf-set-selections && apt-get install -y oracle-java8-installer && rm -rf /var/cache/oracle-jdk8-installer && echo "--- Download and install GlassFish 4.1 ---" && apt-get -y install curl unzip zip inotify-tools && rm -rf /var/lib/apt/lists/* && curl -L -o /tmp/glassfish-4.1.zip http://download.java.net/java/glassfish/4.1/release/glassfish-4.1.zip && unzip /tmp/glassfish-4.1.zip -d /usr/local && rm -f /tmp/glassfish-4.1.zip' returned a non-zero code: 100
vincent@UbuntuDesktop:~/Documents/annotation/SWAN/swans$
```

We identified the problem as the following in the Dockerfile:

```
!UN echo "--- Update and upgrade Ubuntu ---" && \
apt-get update && \
apt-get -y upgrade && \
apt-get -y install software-properties-common && \
apt-get -y install python-software-properties && \
add-apt-repository -y ppa:webupd8team/java && \
apt-get update && \

echo "--- Install JDK8 ---" && \
echo oracle-java8-installer shared/accepted-oracle-license-v1-1 select true | debconf-set-selections && \
apt-get install -y oracle-java8-installer && \
rm -rf /var/cache/oracle-jdk8-installer && \
```

Java 8 on linux is fundamentally broken after some changes in licensing agreements from Oracle, and it is no longer possible to install Oracle-JDK8 via apt-get.

Any and all Dockerfiles that try to install java8 as a dependency this way will automatically fail

There still are ways to manually install java8, but those don't mesh with the automated build approach of docker

There's also a replacement of Oracle-JDK8 in the form of OpenJDK8, but this doesn't help if developers don't use it in their docker files.

WAT-SI

Docker container

<https://github.com/webis-de/wat>

An official Dockerfile is included in the GitHub

Installation documentation

<https://github.com/webis-de/wat/releases/tag/2.0.0>

Appendix to “Requirements engineering for natural-language annotation tasks” Bachelor’s Thesis by
Vincent Söllner, Bauhaus-Universität Weimar, Matriculation Number 118764

1. Download wat.jar
2. Download example-project.zip
3. Extract zip file, enter folder
4. \$ java -jar <path-to>/wat.jar [<port> [<base-path>]]
Default port is 2112

WebAnno

Docker container

Since WebAnno is delivered as a java standalone file, packing it into a Docker container would make the installation more complicated rather than simpler.

Installation documentation

<https://webanno.github.io/webanno/downloads/>

1. Download INCEpTION
2. Start .jar by double-clicking
3. Run via localhost:8080

YEDDA

Docker container

Installation documentation

<https://github.com/jiesutd/YEDDA>

1. \$ git clone <https://github.com/jiesutd/YEDDA.git>
2. \$ cd YEDDA
3. \$ python YEDDA.py
 - a. If multiple python versions are installed, that should be python2.7
4. Error:

```
vincent@UbuntuDesktop:~/Documents/annotation/WAT-SL/example-project/YEDDA$ python YEDDA.py
Traceback (most recent call last):
  File "YEDDA.py", line 9, in <module>
    from Tkinter import *
  File "/usr/lib/python2.7/lib-tk/Tkinter.py", line 42, in <module>
    raise ImportError, str(msg) + ', please install the python-tk package'
ImportError: No module named _tkinter, please install the python-tk package
a. vincent@UbuntuDesktop:~/Documents/annotation/WAT-SL/example-project/YEDDA$ 
```

- a.
- b. Fix:
- c. \$ sudo apt-get update -y

Appendix to “Requirements engineering for natural-language annotation tasks” Bachelor’s Thesis by
Vincent Söllner, Bauhaus-Universität Weimar, Matriculation Number 118764

- d. \$ sudo apt-get install -y python-tk
- e. repeat from 3.

Table of contents

[Table of contents](#)

[Introduction](#)

[Note about attribution](#)

[Interface elements](#)

[Label-boxes](#)

[Attributes:](#)

[Problem:](#)

[Examples:](#)

[Highlights](#)

[Attributes:](#)

[Problems:](#)

[Examples:](#)

[Overlapping highlights:](#)

[Zero width annotation:](#)

[Links](#)

[Attributes:](#)

[Problems:](#)

[Link-to-link visualization](#)

[Overlapping link identification](#)

[Visual clutter](#)

[Cross sentence boundary annotation](#)

[Large amounts of annotations](#)

[Discontinuous structure annotation](#)

[Word alignment/cross document annotation example](#)

[Interesting finds:](#)

Introduction

This document features observations about issues in the user interfaces of various annotation tools, as well as proposed solutions to some of these issues. This includes issues in annotation software which we didn't test during the research for this thesis, but found third party reports of.

Most of these problems are annotation project specific or only appear in very niche applications. As a result, typically not enough examples of an issue exist to formulate a pattern based on the issue, or if it is possible, the usefulness of such a pattern would be called into question.

The intended purpose of this document is to inform about UI and UX issues in annotation software and serve as a starting point for QA testing of known edge cases.

Note about attribution

This document utilizes a large amount of screenshots by third parties.

To avoid repetition, a URL before a screenshot is used as short-hand to indicate the source of the image.

Interface elements

Label-boxes

Attributes:

- Float over/under annotated text
- Can be stacked
- Ordered in some way
- Coloured
- Clickable/function as button to access annotation details
- Are anchor-points of relations

Problem:

Label-boxes represent an annotated markable, but markables can be discontinuous. It is not trivial to display this non-continuous property with a label box.

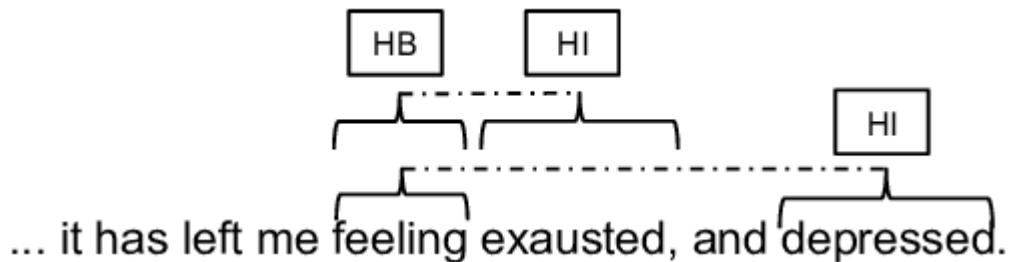
Potential solutions:

- Multiple boxes that highlight on click to indicate shared identity.
- One label-box with different patterns to differentiate between "belongs to span" and "doesn't belong to span".

Examples:

1. <https://github.com/nplab/brat/issues/362>

2. https://www.researchgate.net/figure/A-discontinuous-overlapping-annotation-using-the-extended-BIO-format_fig1_306396469



3. <http://brat.nlplab.org/new-in-v1.3.html>

1 According to some definitions, Panama has territory in both North and South America.

Highlights

Attributes:

- Overlap annotated text
 - Coloured
 - Typically cannot be stacked
 - Typically reserved for text spans of arbitrary length

Problems:

- It is non-trivial to display overlapping spans and proves to be a problem both visually and from a user perspective, because selecting existing annotations for editing becomes tedious.
 - What about spans over zero-length characters?

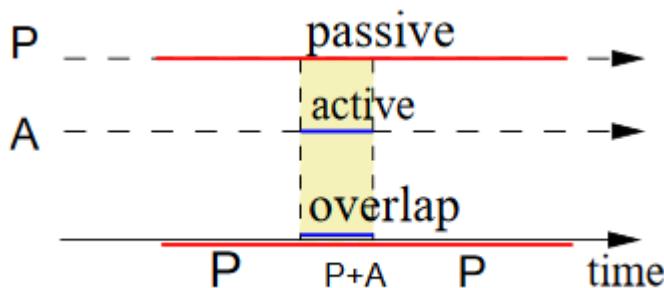
Examples:

Overlapping highlights:

1. <https://stackoverflow.com/questions/56909052/highlight-overlapping-spans-of-text>

2019/06/13, Jim, Alex, Dunedin, 184.3, 93.4
 2019/06/14, William, Tiger, Durango, 154.3, 73.4
 2019/06/15, John, Spider, Dramaqueen, 144.3, 83.4
 2019/06/12, Sarah, Dobbin, Druidtown, 154.3, 63.4
 2019/06/12, Jenny, Andrea, Daniels, 124.3, 73.4

2. <https://perso.limsi.fr/mareuil/publi/788.pdf>



use 2.

3. <https://www.elastic.co/guide/en/elasticsearch/plugins/current/mapper-annotated-text-highlighter.html>

"However, if the search term overlaps the span of an existing annotation it would break the markup formatting so the original annotation is removed in favour of a new annotation with just the search hit information in the results."

Zero width annotation:

1. <https://www.aclweb.org/anthology/C18-1217.pdf>

(ZWNJ is a zero width unicode character, position visualized via a red marker here)

- ZWNJ between two roots or stems of X-Y compounding e.g. اِنْتَاهِيٌّ
- ZWNJ between two roots or stems of X-e-Y compounding e.g. عُلَمَاءٌ،
- ZWNJ before and after ء in X-o-Y compounding e.g. نَفَرَ وَضَبَطَ

Links

Attributes:

- Lines or arrows
- Typically from label-box to label-box
- Weighted links can have label-boxes

Problems:

Link-to-link visualization

Use cases where link-to-link connections are wanted are thinkable. It is non-trivial to visualize or implement this, if the underlying assumption is that links require a label-box as anchor.

Potential solution: Display a label-box over every link, this label-box could then be used as anchor for new links. However, this introduces new problems, such as visual clutter, as in most cases such label-boxes over links aren't required.

Overlapping link identification

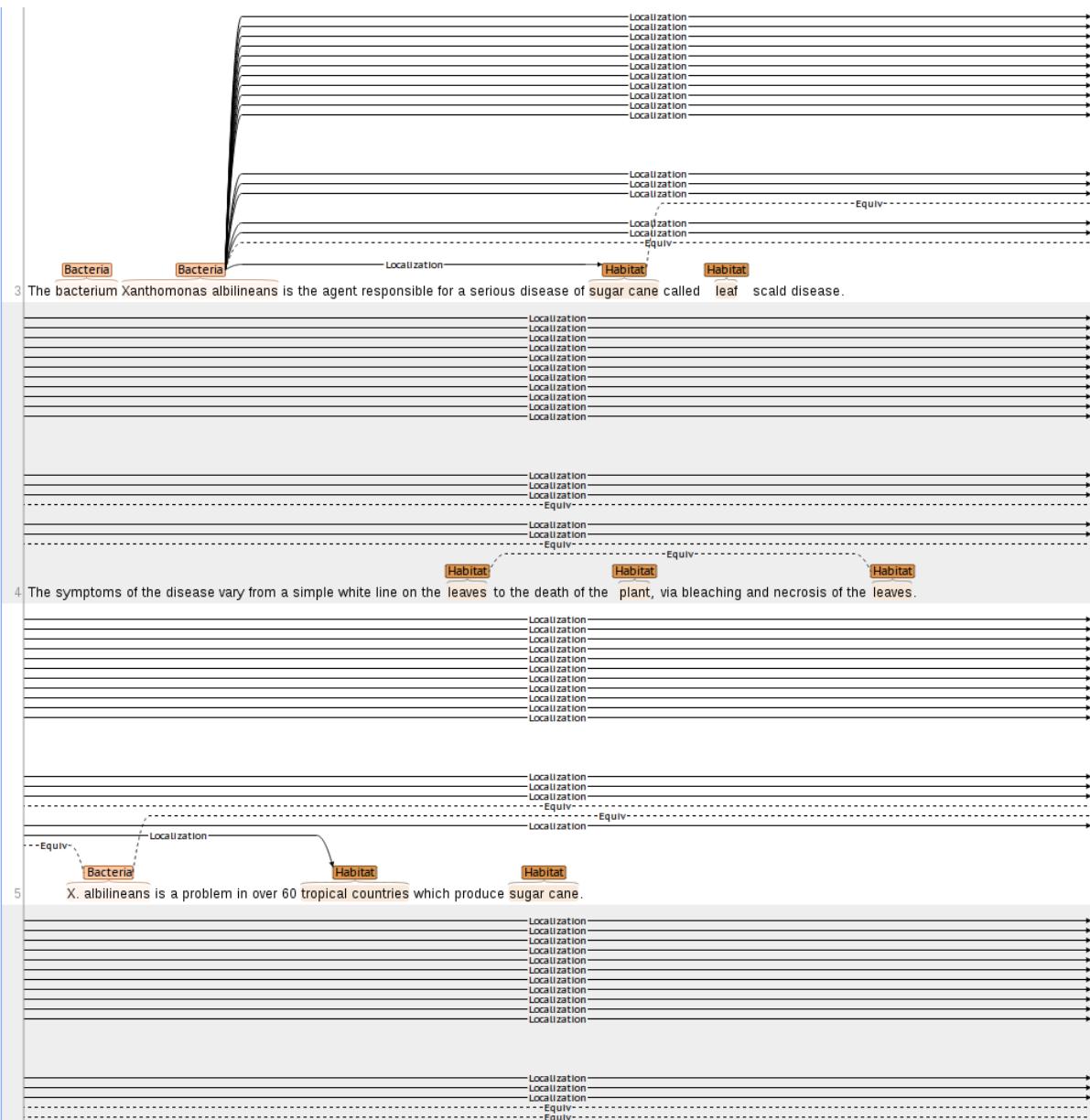
Overlapping links exist (e.g. to visualize two different relations between two labels). It must be possible for users to easily distinguish between the links (or rather between the relations they represent).

Visual clutter

Large amounts of links cause visual clutter, especially if links follow along the lines of text, and the anchoring label-boxes are several sentences apart.

Cross sentence boundary annotation

<https://github.com/nplab/brat/issues/949>



(current version of brat)

Potential solution: Allow links to run diagonally across the screen.

<https://www.aclweb.org/anthology/W18-4702.pdf>



(Modified BRAT release)

Large amounts of annotations

<https://www.aclweb.org/anthology/W15-0303.pdf>

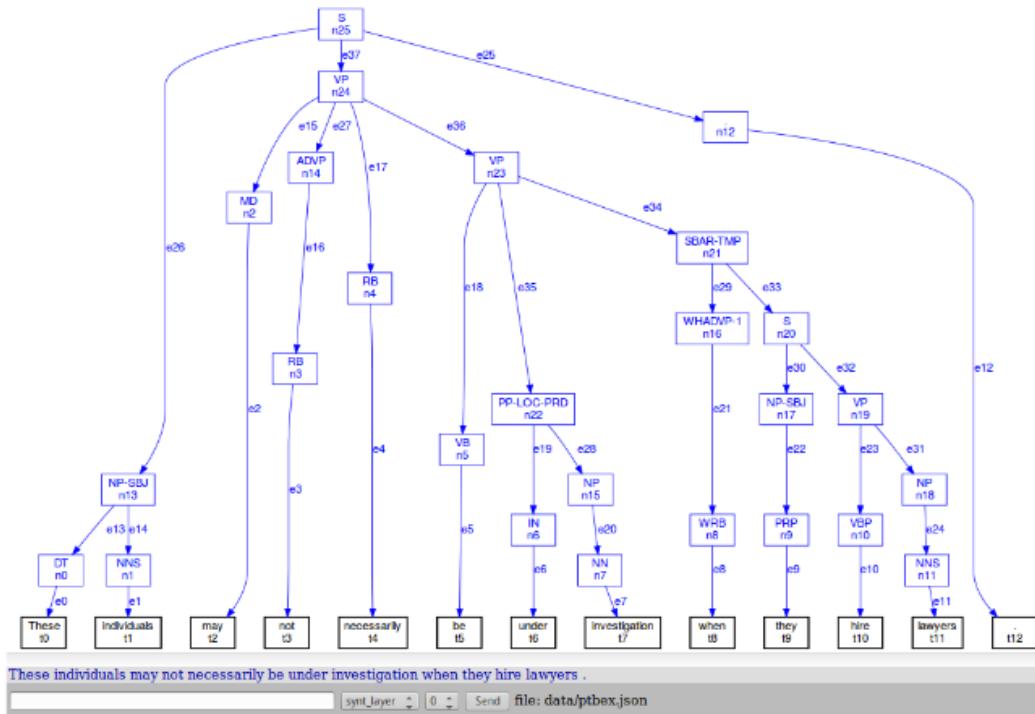


Figure 1: A structurally annotated sentence in GraphAnno

Potential solution: Local zoom/highlighting

<https://www.aclweb.org/anthology/W15-0303.pdf>

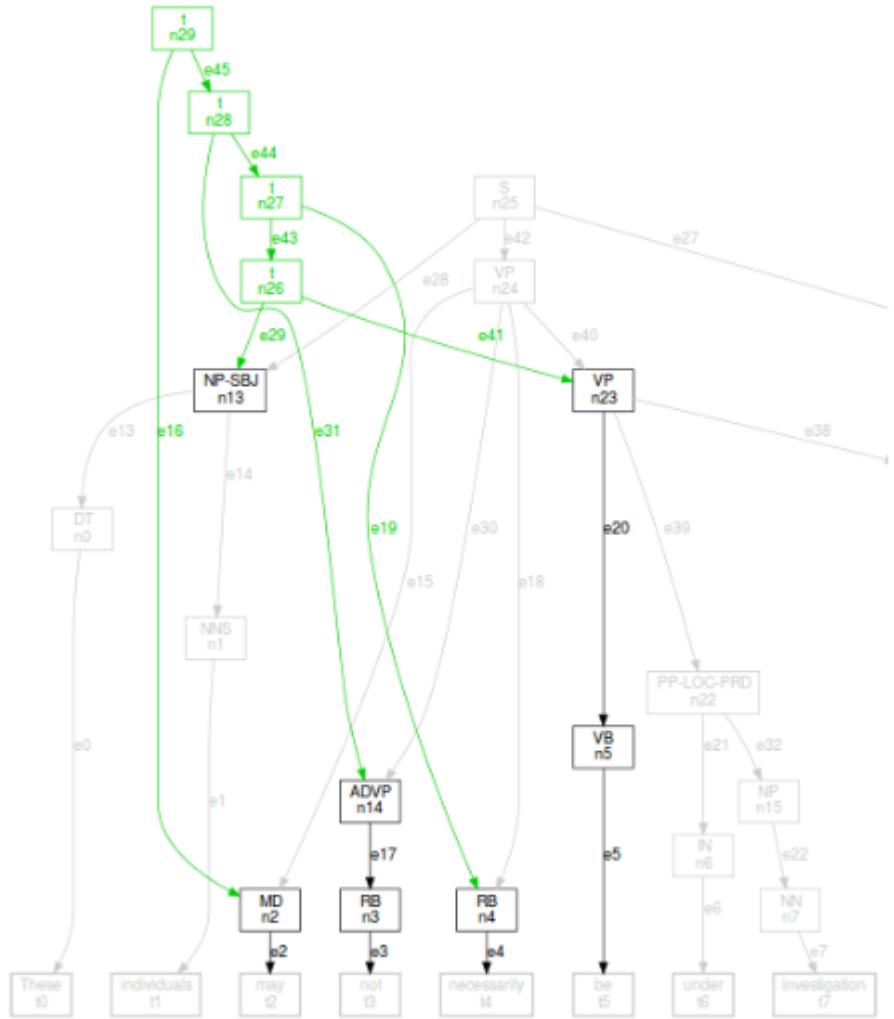


Figure 3: Hiding the structural level

Discontinuous structure annotation

As an extension of the label-box problem with discontinuous markables, it is unclear how to visualize a link between a discontinuous markable and another markable.

Potential solution:

Wolfgang Maier aus Göppingen wrote a PhD thesis on the topic of parsing discontinuous structures:

https://publikationen.uni-tuebingen.de/xmlui/bitstream/handle/10900/47069/pdf/dissertation_maier.pdf?sequence=1&isAllowed=y

Word alignment/cross document annotation example

Links between annotations in different documents (e.g. for alignment tasks such as translation) can exist and need to be displayed.

<https://langsci-press.org/catalog/book/103> chapter 4

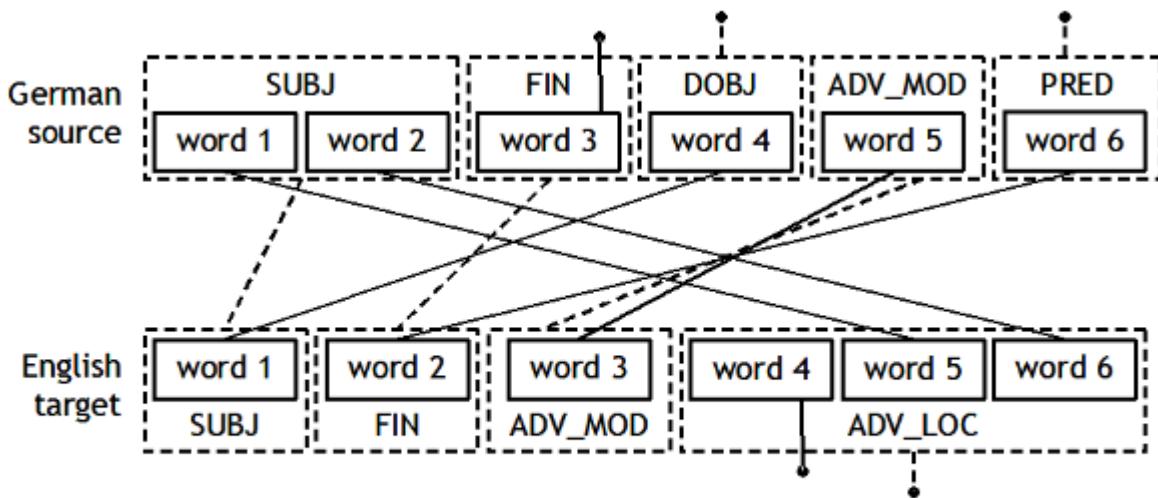


Figure 1: Alignment of grammatical functions and words in sentence 3

Interesting finds:

<https://www.springerprofessional.de/advanced-user-interfaces-for-semantic-annotation-of-complex-rela/15862862>

https://link.springer.com/chapter/10.1007/978-3-319-93581-2_11