INCEpTION User Guide

The INCEpTION Team

Version 0.6.1

Table of Contents

Introduction

This guide summarizes the functionality of INCEpTION from the user's perspective.

It is assumed that you plan to test the INCEpTION standalone version or an already existing server installation of INCEpTION. For information on how to set up INCEpTION for a group of users on a server, please refer to the Administrator Guide.

All materials, including this guide, are available via the INCEpTION homepage.

System Requirements

Table 1. Requirements for users

Browser	Chrome or Safari

Table 2. Requirements to run the standalone version

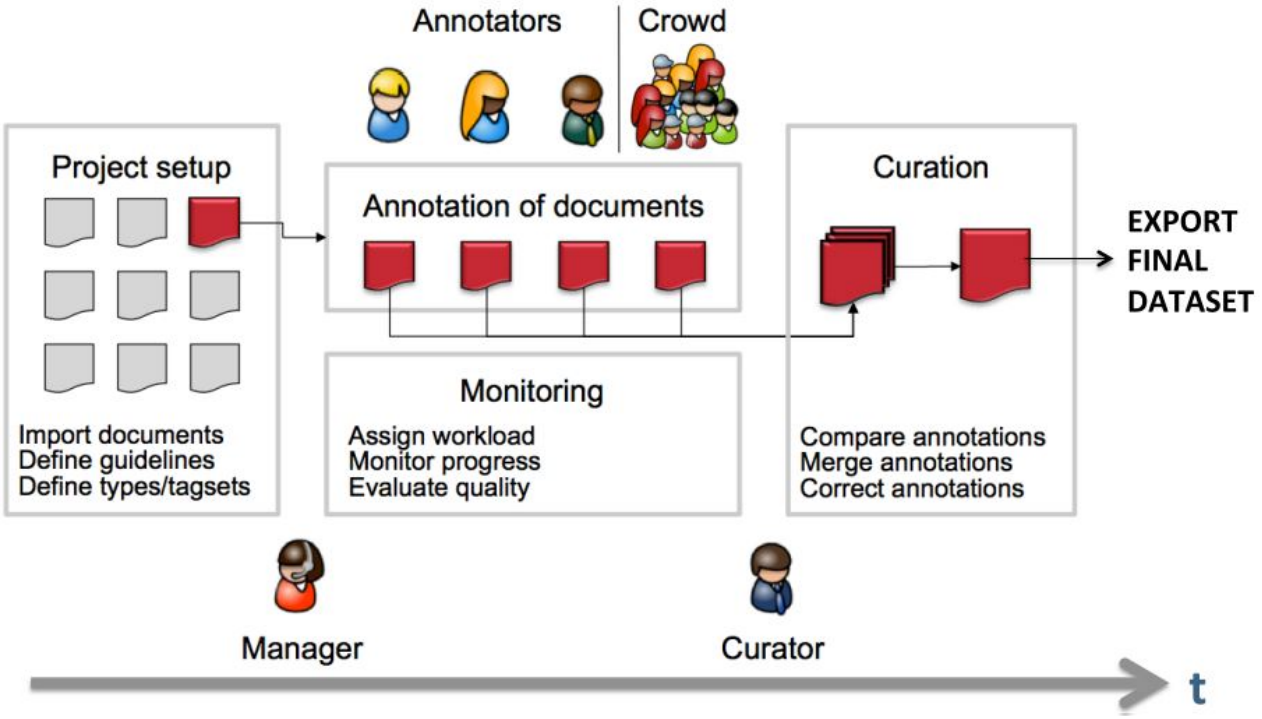
Java Runtime Environment	version 8 or higher

Table 3. Requirements run the server version

Java Runtime Environment	version 8 or higher
Apache Tomcat	version 8.5 or higher (Servlet API 3.1.0)
MySQL Server	version 5 or higher

Workflow

The following image shows an exemplary workflow of an annotation project with INCEpTION.



First, the projects need to be set up. In more detail, this means that users are to be added, guidelines need to be provided, documents have to be uploaded, tagsets need to be defined and uploaded, etc. The process of setting up and administrating a project are explicitly described in Projects.

After the setup of a project, the users who were assigned with the task of annotation annotate the documents according to the guidelines. The task of annotation is further explained in Annotation. The work of the annotators is managed and controlled by monitoring. Here, the person in charge has to assign the workload. For example, in order to prevent redundant annotation, documents which are already annotated by several other annotators and need not be annotated by another person, can be blocked for others. The person in charge is also able to follow the progress of individual annotators. All these tasks are demonstrated in Monitoring in more detail. The person in charge should not only control the quantity, but also the quality of annotation by looking closer into the annotations of individual annotators. This can be done by logging in with the credentials of the annotators.

After at least two annotators have finished the annotation of the same document by clicking on **Done**, the curator can start his work. The curator compares the annotations and corrects them if needed. This task is further explained in Curation.

The document merged by the curator can be exported as soon as the curator clicked on **Done** for the document. The extraction of curated documents is also explained in Projects.

Installation

Run as Java application

All-in-one version which does not require a database server or servlet container to be set up.

By default, INCEpTION creates and uses an embedded database. It is not recommended to use the application in such a configuration for production use. Instead, please use a database server when using it in production. For more information, please refer to the Administrator Guide.

Get the stand-alone JAR from the downloads page and start it simply with a **double-click** in your file manager. The application stores its data in a folder called `.inception` (_dot inception) within your home folder,

Optional configuration

Alternatively, you can start INCEpTION from the command line, in particular if you wish to provide it with additional memory (here 1 GB) or if you want it to store its data in a different folder.

```
java -Xmx1g -Dinception.home=/my/inception/home -jar inception-app-standalone-XXX.jar
```

Mind to replace `/my/inception/home` with path of a folder where the application can store its data.

By default the server starts on port 8080 and you can access it via a browser at `http://localhost:8080` after you started it. You can add the parameter `-Dserver.port=9999` at the end of the command line to start the server on port 9999 (or choose any other port).

INCEpTION uses Spring Boot. If you need to set additional parameters of the embedded webserver of the stand-alone version, please refer to the Spring Boot embedded container documentation.

Upgrade

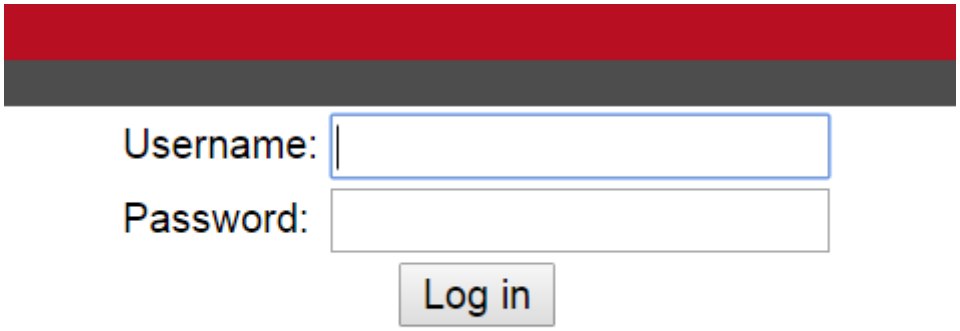
This section describes how to upgrade the standalone version of INCEpTION using an embedded database. For further information on how to upgrade INCEpTION, in particular the WAR version when using a MySQL database or older versions of INCEpTION, please refer to the Administrator Guide.

Before any upgrade, make a copy of your INCEpTION home folder. If INCEpTION is configured to use an external database, e.g. MySQL, make a backup of this database as well. See the Administrator Guide for further information.

Logging in

Upon opening the application in the browser, the login screen opens. Please enter your credentials to proceed.

When INCEpTION is started for the first time, a default user called **admin** with the password **admin** is automatically created. Be sure to change the passwort for this user after logging in (see User Management).



Menu bar

At the top of the screen, there is always a menu bar visible which allows a quick navigation within the application. It offers the following items:

- **Home** - always takes you back to the main menu.
- **Help** - opens the integrated help system in a new browser window.
- **Username** - shows the name of the user currently logged in. If the administrator has allowed it, this is a link which allows accessing the current user's profile, e.g. to change the password.
- **Log out** - logs out of the application.
- **Timer** - shows the remaining time until the current session times out. When this happens, the browser is automatically redirected to the login page.

Main Menu

After login, you will be presented with the overview screen. This screen can be reached at any time from within the GUI by clicking on the **Home** link in the left upper corner.

Here, you can navigate to one of the currently seven options:

- Annotation - The page to perform annotations
- Curation - Compare and merge annotations from multiple users (only for *curators*)
- [sect_correction] - Correcting automatic annotation (under development)
- [sect_automation] - Creating automatically annotated data
- Projects - Set up or change annotation projects (only for *administrators*)
- Monitoring - Allows you to see the projects, their progress and change document status (only for *administrators* and *curators*)
- User Management - Allows you to manage the rights of users

Please click on the functionality you need. The individual functionalities will be explained in further chapters.

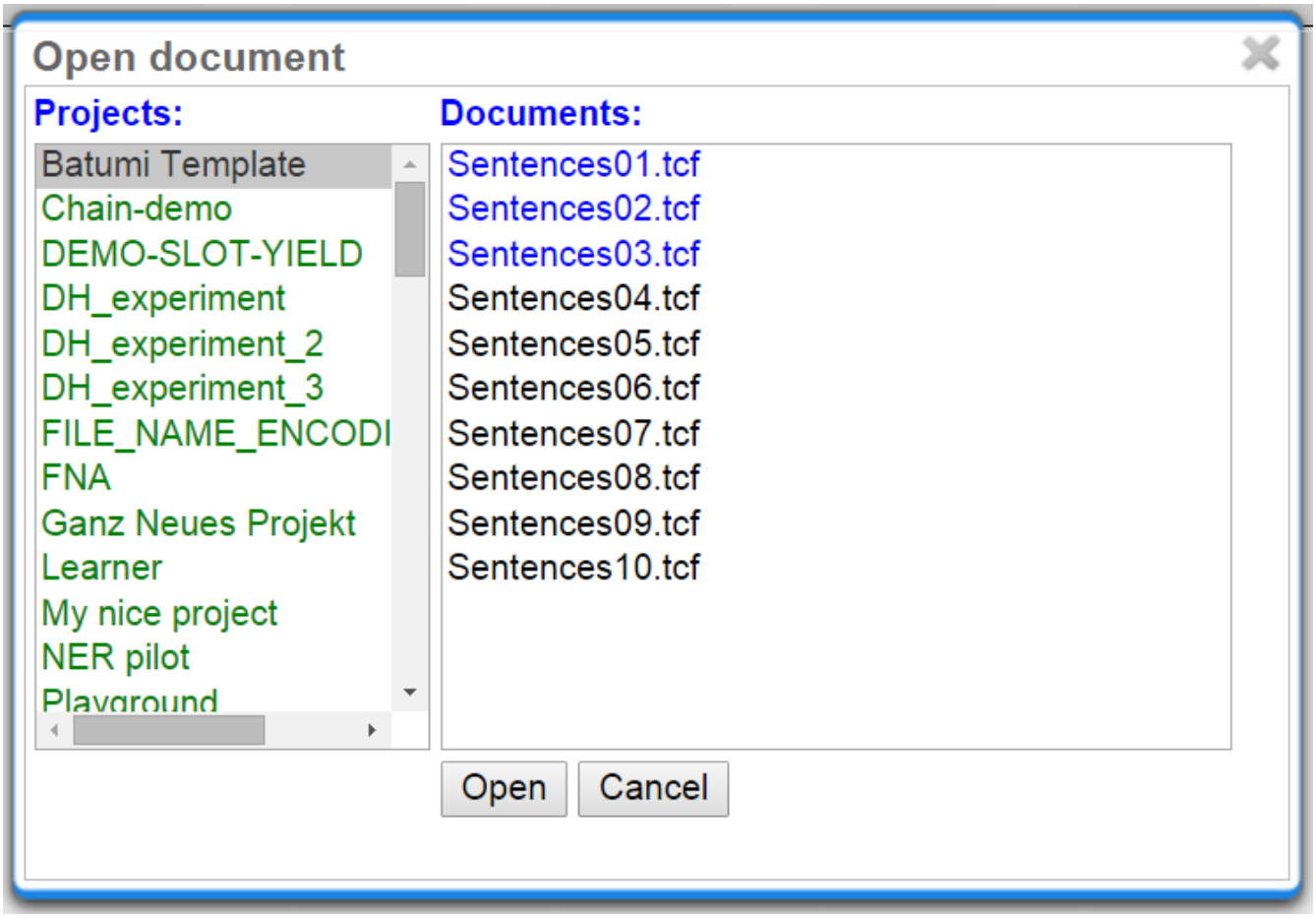
Annotation

This functionality is only available to **annotators**, **project managers**, and **administrators**. Annotators and project managers only see projects in which they hold the respective roles.

The annotation screen allows to view text documents and to annotate them.

Opening a Document

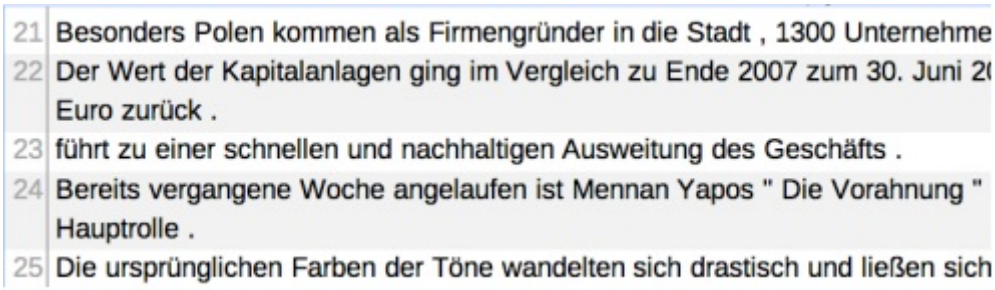
When navigating to the **Annotation** page, a dialogue opens that allows you to select a project, and a document within the project. If you want to open a different project or document later, click on **Open** to open the dialog.



Projects appear as folders, and contain the documents of the project. Double-click on a document to open it for annotation. Document names written in black show that the document has not been opened by the current user, blue font means that it has already been opened, whereas red font indicates that the document has already been marked as **done**.

Navigation

Sentence numbers on the left side of the annotation page show the exact sentence numbers in the document.



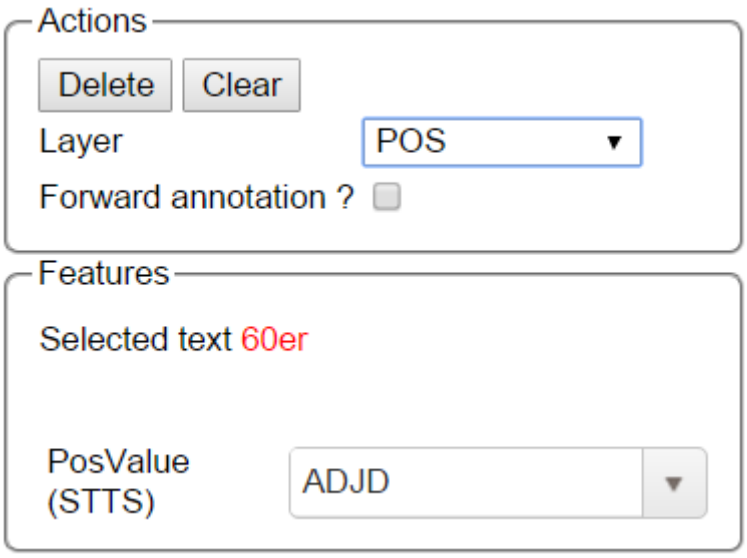
The arrow buttons **first page**, **next page**, **previous page**, **last page**, and **go to page** allow you to navigate accordingly. The **Prev.** and **Next** buttons in the **Document** frame allow you to go to the previous or next document on your project list. You can also use the following keyboard assignments in order to navigate only using your keyboard.

Table 4. Navigation key bindings

Key	Action
HOME	jump to first sentence
END	jump to last sentence
PAGE DOWN	move to the next page, if not in the last page already
PAGE UP	move to previous page, if not already in the first page
SHIFT+PAGE DOWN	go to next document in project, if available
SHIFT+PAGE UP	go to previous document in project, if available

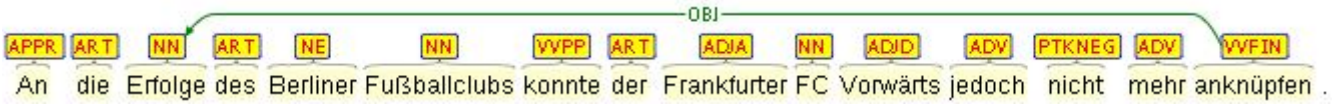
A click on the **Help** button displays the Guidelines for the tool and **The Annotator's Guide to NER-Annotation**. When you are finished with annotating or curating a document, please click on the **Done** button, so that the document may be further processed. If the button above the **Done** is a cross symbol, it means the documents have already been finished. If the symbol has a tick, it is still open.

Annotation of spans works by selecting the span, or double-clicking on a word. This activates the **Actions**-box on the right, where you can choose a layer. One can also type in the initial letters and chose the needed layer. After having chosen a layer, the drop-down menu inside the **Features**-box displays the features you can use during the annotation. The tag can be selected out of the drop-down menu inside the **Features**-box which contains the tags of the chosen layer.



To change or delete an annotation, double-click on the annotation (span or link annotations). The **Actions**-box is now activated. Changes and Deletions are possible via the respective buttons.

Link annotations (between POS tags) are created by selecting the starting POS-tag, then dragging the arrow to connect it to its target POS tag. All possible targets are highlighted.



Creating annotations

The **Layer** box in the right sidebar shows the presently active layer span layer. To create a span annotation, select a span of text or double click on a word.

If a relation layer is defined on top of a span layer, clicking on a corresponding span annotation and dragging the mouse creates a relation annotation.

One an annotation has been created or if an annotation is selected, the **Annotation** box shows the features of the annotation.

The result of changing the active layer in the **Layer** box while an annotation is selected depends on the **Remember layer** setting. If this setting is disabled, changing the active layer causes the currently selected annotation to be deleted and replaced with an annotation of the selected layer. In this mode, it is necessary to unselect the current annotation by pressing the **Clear** button before an annotation on another layer can be created. If **Remember layer** is enabled, changing the active layer has no effect on the currently selected annotation.

The definition of layers is covered in Section Layers.

Spans

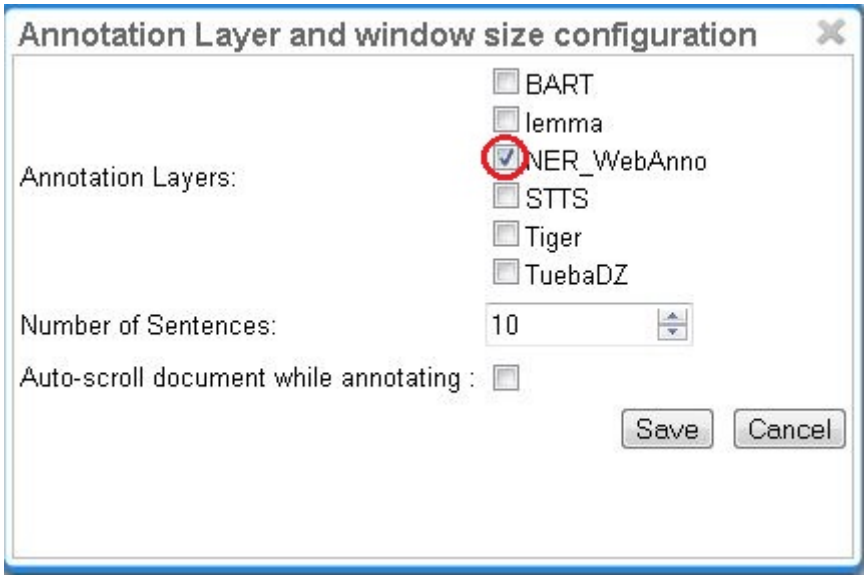
To create an annotation over a span of text, click with the mouse on the text and drag the mouse to create a selection. When you release the mouse, the selected span is activated and highlighted in orange. The annotation detail editor is updated to display the text you have currently selected and

to offer a choice on which layer the annotation is to be created. As soon as a layer has been selected, it is automatically assigned to the selected span. To delete an annotation, select a span and click on **Delete**. To deactivate a selected span, click on **Clear**.

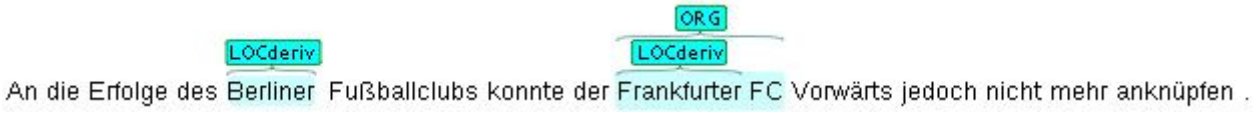
Depending on the layer behavior configuration, spans annotations can have any length, can overlap, can stack, can nest, and can cross sentence boundaries.

Example

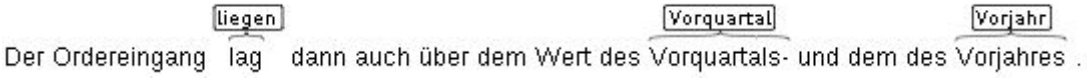
For example, for NE annotation, select the options as shown below (red check mark):



NE annotation can be chosen from a tagset and can span over several tokens within one sentence. Nested NE annotations are also possible (in the example below: "Frankfurter" in "Frankfurter FC").



Lemma annotation, as shown below, is freely selectable over a single token.



POS can be chosen over one token out of a tagset.

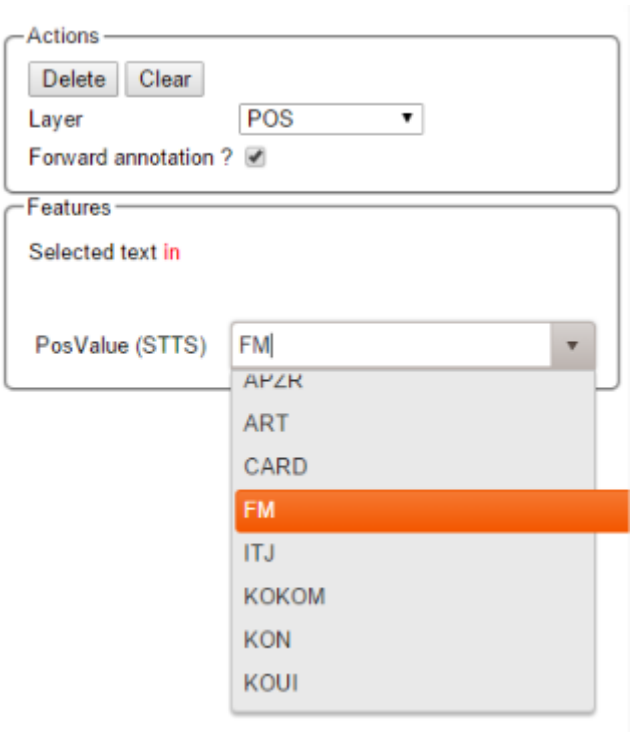
Zero-width spans

To create a zero-length annotation, hold **SHIFT** and click on the position where you wish to create the annotation. To avoid accidental creations of zero-length annotations, a simple single-click triggers no action by default. The **lock to token** behavior cancels the ability to create zero-length annotations.

A zero-width span between two tokens that are directly adjacent, e.g. the full stop at the end of a sentence and the token before it (`end.`) is always considered to be **at the end of the first token** rather than at the beginning of the next token. So an annotation between `d` and `.` in this example would rendered at the right side of `end` rather than at the left side of `.`.

Forward annotation

To improve the speed of POS-annotation, select **forward annotation** in the **Actions** box on the left side of your screen. This allows you to select POS-tags via the keys of your keyboard. Pushing a key several times successively proposes every POS-tag starting with the respective letter inside the **Features** box. Pressing a key whose letter does not represent the beginning of any tag leads to the first tag in the tagset. Once a POS-tag has been selected, pushing **space** and **Enter** keys automatically assigns the POS-tag to the token in focus and the next token can be annotated as described. Note that the **Enter** key will not work for the **Safari** browser. Also the **Forward annotation** works only for span annotations with 1) **tagset** and 2) a layer with only one **feature**.



Co-reference annotation can be made over several tokens within one sentence. A single token sequence has several co-ref spans simultaneously.

Relations

To create a relation annotation, click on a span annotation and drag the mouse to another span annotation. While you drag, an arc is drawn. It is not possible to create arbitrary relation annotations. In order to create one, a corresponding relation layer needs to be defined between the source and target spans.

Depending on the layer behavior configuration, relation annotations can stack, can cross each other, and can cross sentence boundaries.

Self-looping relations

To create a relation from a span to itself, press the **SHIFT** key before starting to drag the mouse and hold it until you release the mouse button.

To abort the creation of an annotation, hold the **CTRL** key when you release the mouse button.

Currently, there can be at most one relation layer per span layer. Relations between spans of different layers are not supported.

Not all arcs displayed in the annotation view are belonging to chain or relation layers. Some are induced by Link Features.

When moving the mouse over an annotation with outgoing relations, the info popup includes the **yield** of the relations. This is the text transitively covered by the outgoing relations. This is useful e.g. in order to see all text governed the head of a particular dependency relation. The text may be abbreviated.

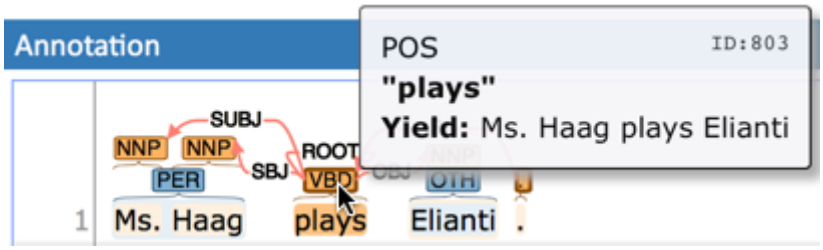


Figure 1. Example of the yield of a dependency relation

Chains

A chain layer includes both, span and relation annotations, into a single structural layer. Creating a span annotation in a chain layer basically creates a chain of length one. Creating a relation between two chain elements has different effects depending on whether the **linked list** behavior is enabled for the chain layer or not. To enable or disable the **linked list** behaviour, go to **Layers** in the **Projects Settings** mode. After choosing **Coreference**, **linked list** behaviour is displayed in the checkbox and can either be marked or unmarked.

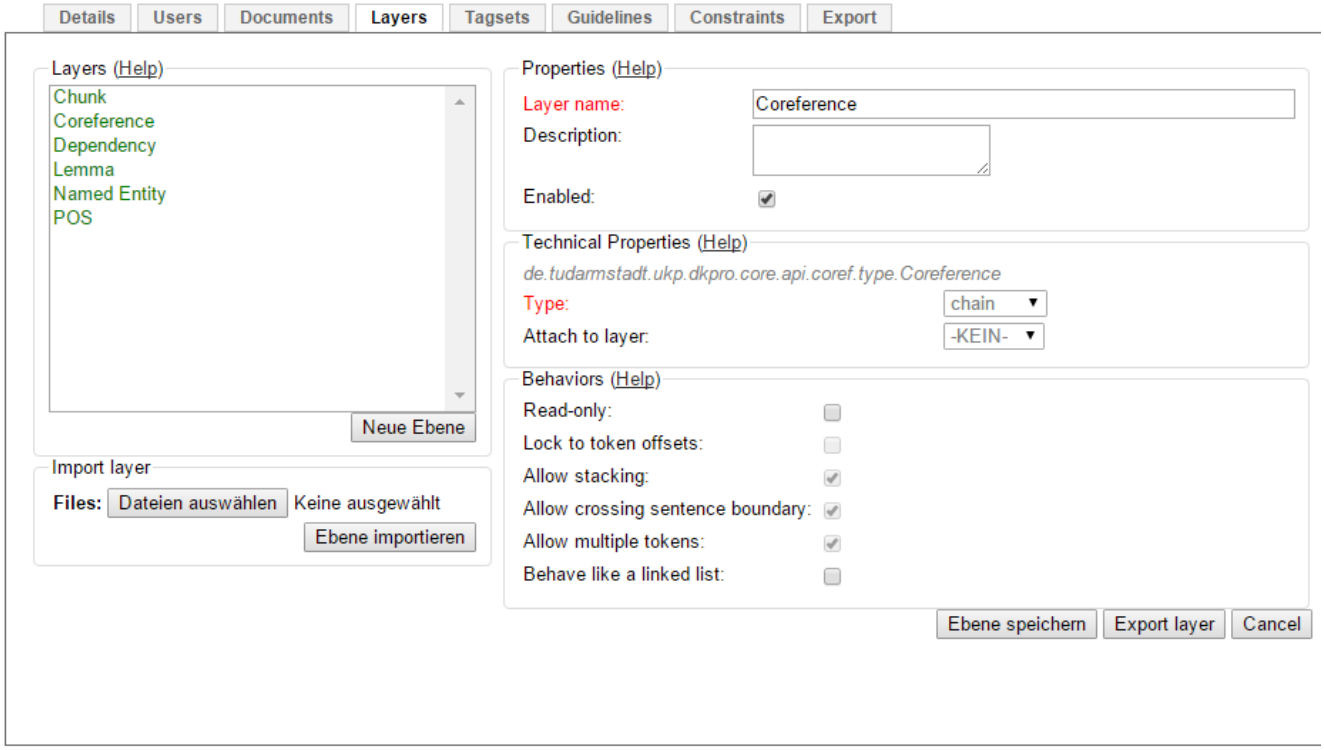


Figure 2. Configuration of a chain layer in the project settings

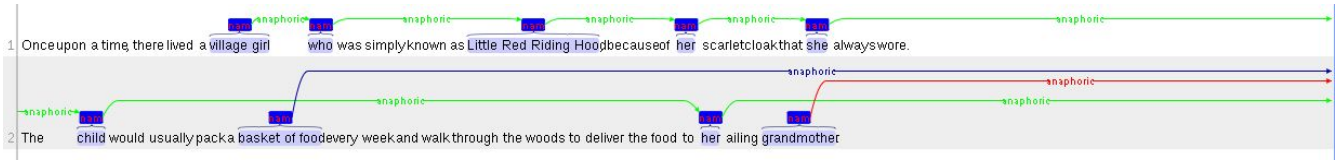


Figure 3. Example of chain annotations

To abort the creation of an annotation, hold **CTRL** when you release the mouse button.

Table 5. 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.

Primitive Features

Supported primitive features types are string, boolean, integer, and float. Boolean features are displayed as a checkbox that can either be marked or unmarked. Integer and float features are displayed using a number field. String features are displayed using a text field or - in case they have a tagset - using a combobox.

Link Features

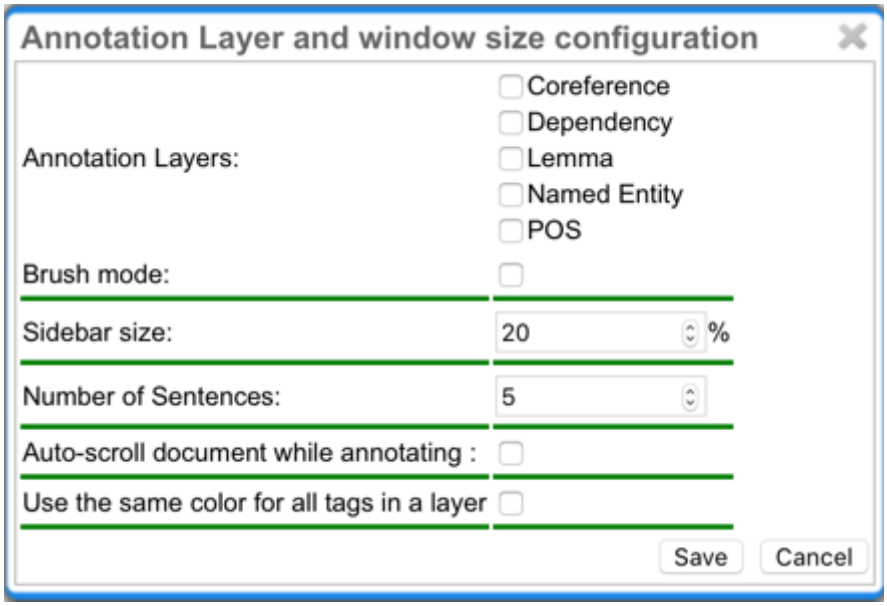
Link features can be used to link one annotation to others. Before a link can be made, a slot with a role must be added. Enter the role label in the text field and press the **add** button to create the slot. Next, click on field in the newly created slot to **arm** it. The field's color will change to indicate that it is armed. Now you can fill the slot by double-clicking on a span annotation. To remove a slot, arm it and then press the **del** button.

Changing role names

To change a previously selected role name, no prior deletion is needed. Just double-click on the instance you want to change, it will be highlighted in orange, and chose another role name.

Settings

Once the document is opened, a default of 5 sentences is loaded on the annotation page. The **Settings** button will allow you to specify the settings of the annotation layer.



Next to **Annotation layers**, you to select the annotation layer which is displayed during annotation. This is useful to reduce clutter if there are many annotation layers. Mind that hiding a layer which has relations attached to it will also hide the respective relations. E.g. if you disable POS, then no dependency relations will be visible anymore.

The **Remember layer** checkbox controls if the annotation layer selected in the **Actions** box. It will work as main layer during the annotation process. Only instances of this layer will be created, even if an annotation in another layer is selected. If necessary, it is possible to change active instances.

Still, if a new instance is selected, the main layer is automatically activated.

The **Sidebar size** controls the width of the sidebar containing the annotation detail edtior and actions box. In particular on small screens, increasing this can be useful. The sidebar can be configured to take between 10% and 50% of the screen.

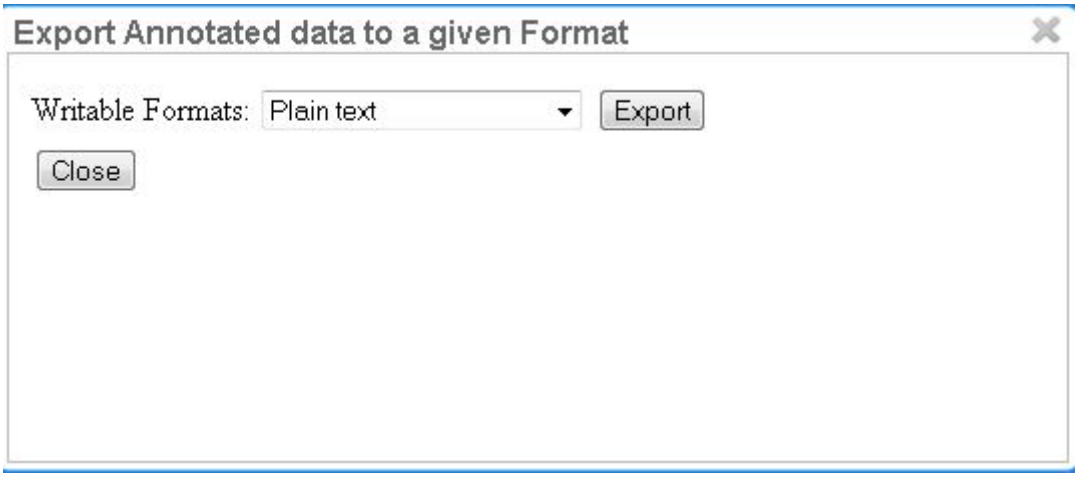
The **Number of sentences** controls how many sentences are visible in the annotation area. The more sentences are visible, the slower the user interface will react.

The **Auto-scroll** setting controls if the annotation view is centered on the sentence in which the last annotation was made. This can be useful to avoid manual navigation.

If **Use the same color for all tags in a layer** is chosen, annotations are colored per layer. If this option is off, then annotations are colored by their labels (all annotations with the same label also have the same color). Mind that there is a limited number of colors such that eventually colors will be reused.

Export

Annotations are always immediately persistent in the backend database. Thus, it is not necessary to save the annotations explicitly. Also, losing the connection through network issues or timeouts does not cause data loss. To obtain a local copy of the current document, click on **export** button. The following frame will appear:



Choose your preferred format. Please take note of the facts that the plain text format does not contain any annotations and that the files in the binary format need to be unpacked before further usage. For further information the supported formats, please consult the corresponding chapter Formats.

The document will be saved to your local disk, and can be re-imported via adding the document to a project by a project administrator. Please export your data periodically, at least when finishing a document or not continuing annotations for an extended period of time.

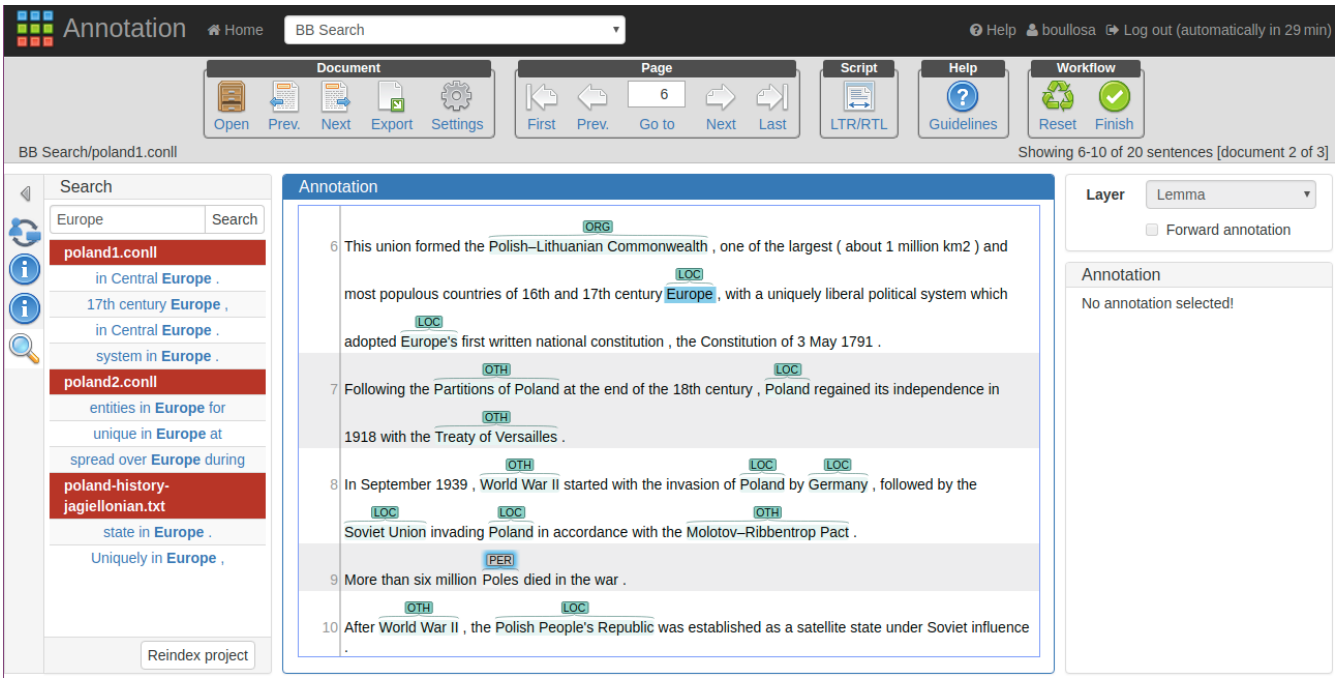
Search

The INCEpTION search module allows to search for words, passages and annotations made in the documents of a given project. For doing a search, the user must access the search sidebar located in

the left of the screen, write a query and press the **Search** button. The results will be shown below the query, grouped by the document where they come from. Every result is shown in KWIC (keyword in context) style, i.e., surrounded by a left and right context to facilitate its identification.

Clicking on a result will make the central main editor automatically jump to the position of that result inside the original document. Only documents contained in the current project will be retrieved by a given search.

INCEpTION allows the configuration of different search providers. Currently, the default search is provided by **Mtas** (Multi Tier Annotation Search), a Lucene/Solr based search and indexing mechanism developed by Meertens Institut (https://meertensinstituut.github.io/mtas).



Mtas search syntax

The INCEpTION Mtas search provider allows queries to be executed using CQL (Corpus Query Language), as shown in the following examples. More examples and information about CQL syntax can be found in https://meertensinstituut.github.io/mtas/search_cql.html.

When performing queries, the user must reference the annotation types using the layer names, as defined in the project schema. In the same way, the features must be referenced using their names as defined in the project schema. In both cases, empty spaces in the names must be replaced by an underscore.

Thus, **Lemma** refers to the **Lemma** layer, **Lemma.Lemma** refers to the the **Lemma** feature in the **Lemma** layer. In the same way, **Named_entity** refers to **Named entity** layer, and **Named_entity.value** refers to the **value** feature in the **Named entity** layer.

Annotations made over single tokens can be queried using the [...] syntax, while annotations made over multiple tokens must be queried using the <../> syntax.

In the first case, the user must always provide a feature and a value. The following syntax returns all single token annotations of the **LayerX** layer whose **FeatureX** feature have the given value.

```
[LayerX.FeatureX="value"]
```

In the second case, the user may or not provide a feature and a value. Thus, the following syntax will return all multi-token annotations of the **LayerX** layer, regardless of their features and values.

```
<LayerX/>
```

On the other hand, the following syntax will return the multi-token annotations whose **FeatureX** feature has the given value.

```
<LayerX.FeatureX="value"/>
```

Notice that the multi-token query syntax can also be used to retrieve single token annotations (e.g. POS or lemma annotations).

Basic Annotation queries

Single token: all occurrences of the token **Galicia**

```
Galicia
```

Single token: all occurrences of the token **Galicia** *(alternative)*

```
"Galicia"
```

Multiple tokens: all occurrences of the token sequence **The capital of Galicia**

```
The capital of Galicia
```

Multiple tokens: all occurrences of the token sequence **The capital of Galicia** *(alternative)*

```
"The" "capital" "of" "Galicia"
```

Lemma: all occurrences of the lemma **sign**

```
[Lemma.Lemma="sign"]
```

Named entities: all named entity annotations

```
<Named_entity/>
```

Named entities: all occurrencies of a particular kind of named entity (in this case, **location** *named entities)*

```
<Named_entity.value="LOC"/>
```

Sequence: all occurrences of the lemma **be** *immediately followed by the lemma* **signed**

```
[Lemma.Lemma="be"] [Lemma.Lemma="sign"]
```

Sequence: all occurrences of the token **house** *immediately followed by a verb*

```
"house" [POS.PosValue="VERB"]
```

Sequence: all occurrences of a verb immediately followed by a named entity

```
[POS.PosValue="VERB"]<Named_entity/>
```

Sequence: All occurrences of two named entities in a row

```
<Named_entity/>{2}
```

Sequence: All occurrences of two named entities in a row (alternative syntax)

```
<Named_entity/> <Named_entity/>
```

Sequence: All occurrences of a named entity followed by a token (whatever it is) and another named entity:

```
<Named_entity/> [] <Named_entity/>
```

Sequence: All occurrences of a named entity followed by an optional token and another named entity:

```
<Named_entity/> []? <Named_entity/>
```

Sequence: All occurrences of two named entities separated by exactly two tokens

```
<Named_entity/> []{2} <Named_entity/>
```

Sequence: All occurrences of two named entities separated by among one and three tokens

```
<Named_entity/> []{1,3} <Named_entity/>
```

OR: All named entities of type LOC or OTH

```
(<Named_entity.value="OTH"/> | <Named_entity.value="LOC"/>)
```

Within: All occurrences of the lemma **sign** *annotated as a verb*

```
[POS.PosValue="VERB"] within [Lemma.Lemma="sign"]
```

Within: All occurrences of a determinant inside a named entity

```
[POS.PosValue="DET"] within <Named_entity/>
```

Not within: All occurrences of a determinant not inside a named entity

```
[POS.PosValue="DET"] !within <Named_entity/>
```

Containing: All occurrences of named entities containing a determinant

```
<Named_entity/> containing [POS.PosValue="DET"]
```

Not containing: All occurrences of named entities not containing a determinant

```
<Named_entity/> !containing [POS.PosValue="DET"]
```

Intersecting: All named entities of type LOC intersecting with a semantic argument

```
<Named_entity.value="LOC"/> intersecting <SemArg/>
```

OR combined with Within: All named entities of type LOC or OTH contained in a semantic argument

```
(<Named_entity.value="OTH"/> | <Named_entity.value="LOC"/>) within <SemArg/>
```

OR combined with Intersecting query: Named entities of type LOC or OTH intersecting with a semantic argument

```
(<Named_entity.value="OTH"/> | <Named_entity.value="LOC"/>) intersecting <SemArg/>
```

Relation queries

When relations are index, they are indexed by their target span.

Search for dependency targets

```
<Dependency/>
```

Search for dependency based on a feature value

```
<Dependency.Relation="nsubj"/>
```

Search for dependency target by the source text

```
<Dependency-source="John"/>
```

Search for dependency target by the target text

```
<Dependency-target="Miller"/>
```

The following examples work for custom relation layers, but not for the built-in **Dependency** layer. We assume a span layer called `component` and a relation layer called `rel` attached to it. Both layers have a string feature called `value`.

Search for rel annotations by feature on the relation source

```
<rel-source.value="foo"/>
```

Search for rel annotations by feature on the relation target

```
<rel-target.value="foo"/>
```

Concept Annotation queries

Generic Search over annotated KB entities : all occurrences for KB entity **Bordeaux**

```
<KB-Entity="Bordeaux"/>
```

The following query returns all mentions of **ChateauMorgonBeaujolais** or any of its subclasses in the associated knowledge base.

Named Entity Identifier for KB instance: all mentions of **ChateauMorgonBeaujolais**

```
<Named_entity.identifier="ChateauMorgonBeaujolais"/>
```

Mind that the label of a knowledge base item may be ambiguous, so it may be necessary to search by IRI.

Named Entity Identifier for KB instance: all mentions of **ChateauMorgonBeaujolais** *by IRI*

```
<Named_entity.identifier="http://www.w3.org/TR/2003/PR-owl-guide-
20031209/wine#ChateauMorgonBeaujolais"/>
```

Named Entity Identifier : all exact mentions of **ChateauMorgonBeaujolais** .

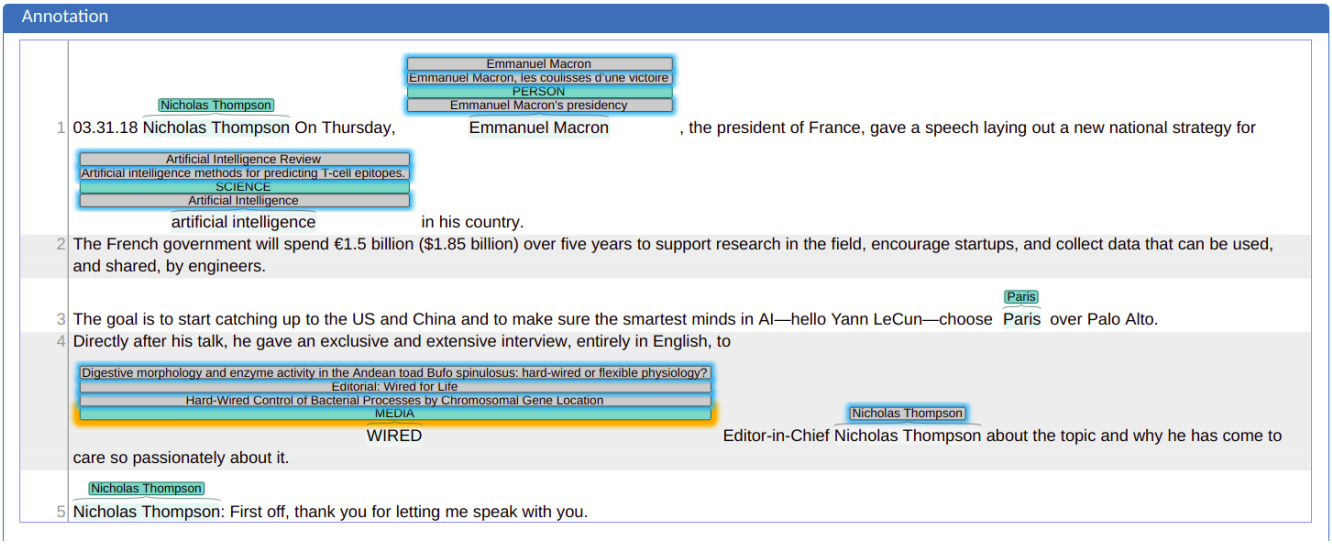
```
<Named_entity.identifier-exact="ChateauMorgonBeaujolais"/>
```

OR All exact mentions of either **ChateauMorgonBeaujolais** *or* **AmericanWine**

```
(<Named_entity.identifier-exact="ChateauMorgonBeaujolais"/> |
<Named_entity.identifier-exact="AmericanWine"/>)
```

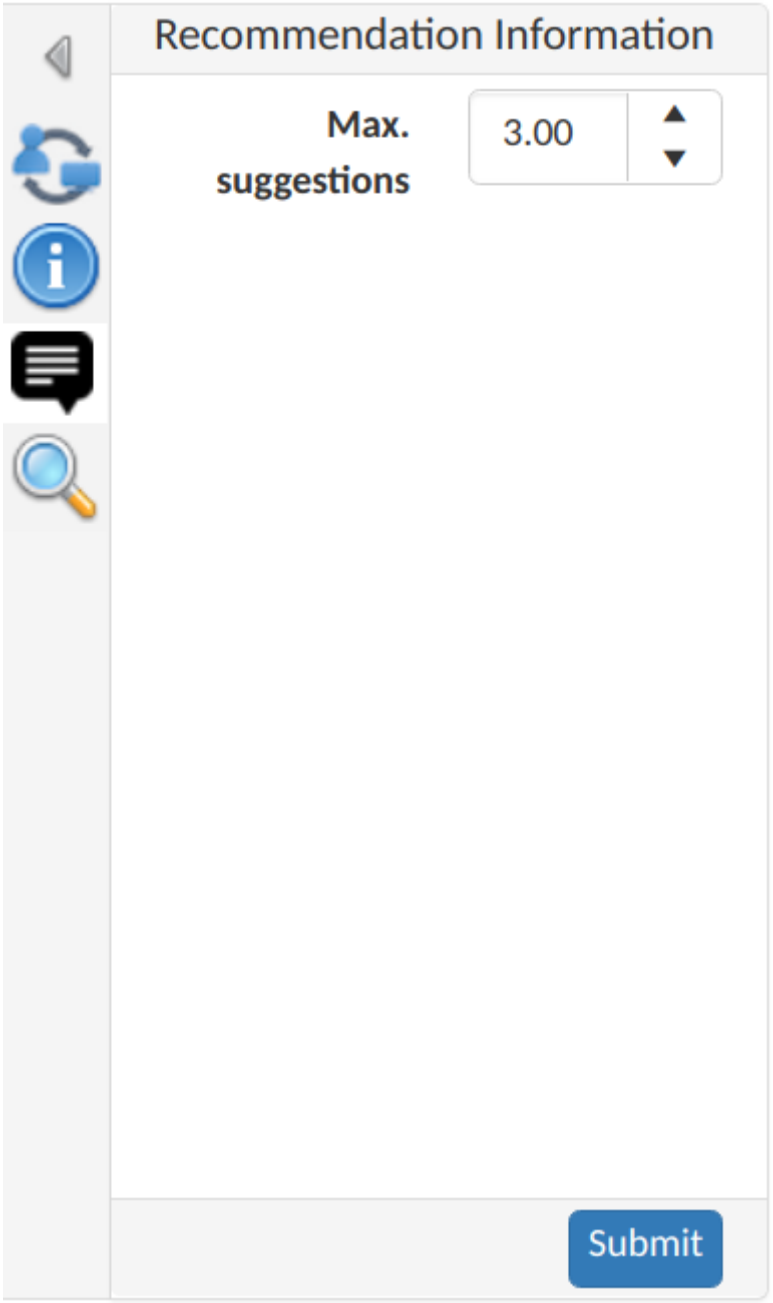
Recommenders

If you have configured one or more recommenders in the Project Settings, you will see recommendations like in the screenshot below. Clicking **Reset** in the *Workflow* area will remove all predictions, however it will also remove all hand-made annotations. Predictions made by a specific recommender can be deleted by removing the corresponding recommender in the Project Settings .



Recommendation Sidebar

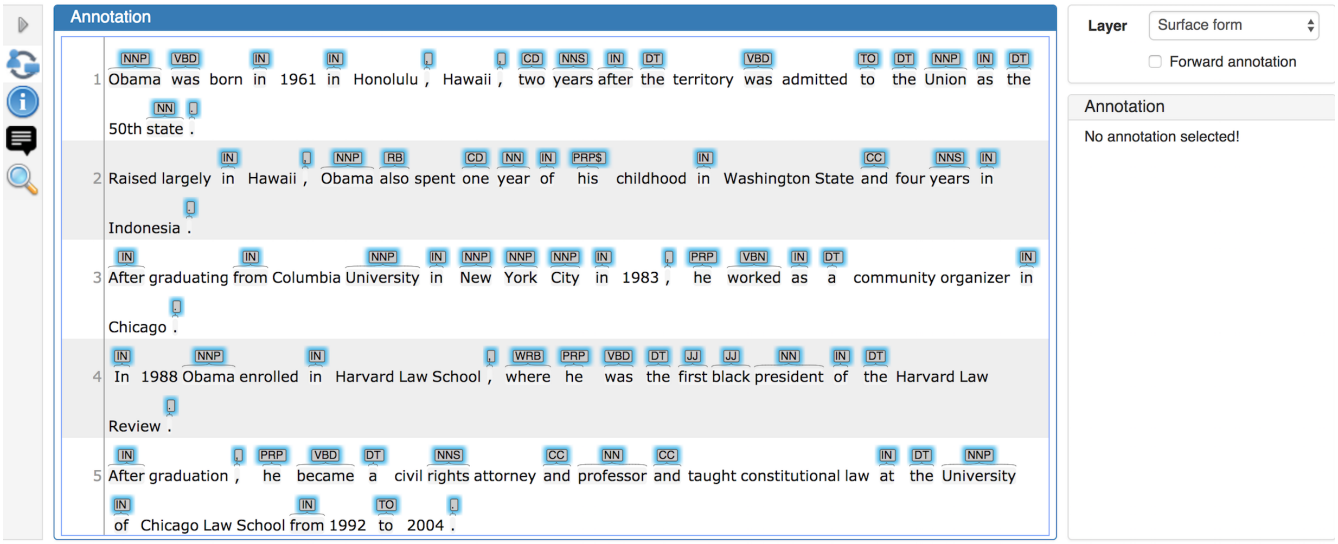
Clicking the speech bubble on the left opens the recommendation sidebar. There you can set the maximum number of recommendations for each token. Don't forget to click on **Submit**.



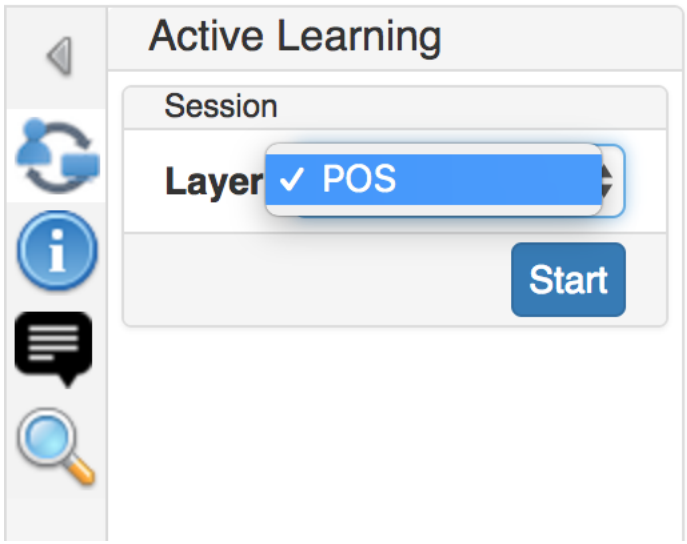
Active Learning

Active learning is a family of methods which seeks to optimize the learning rate of classification algorithms by intelligently soliciting labels from a human user for which the system only has low confidence. This means that recommenders can make better suggestions with less user interactions, allowing the user to perform quicker and more accurate annotations.

Once the recommenders are set in the Project Settings, and assuming the project contains documents for annotation and enough annotations for recommenders to generate recommendations, one can now switch to the annotation page. The recommendations should be shown above the tokens:

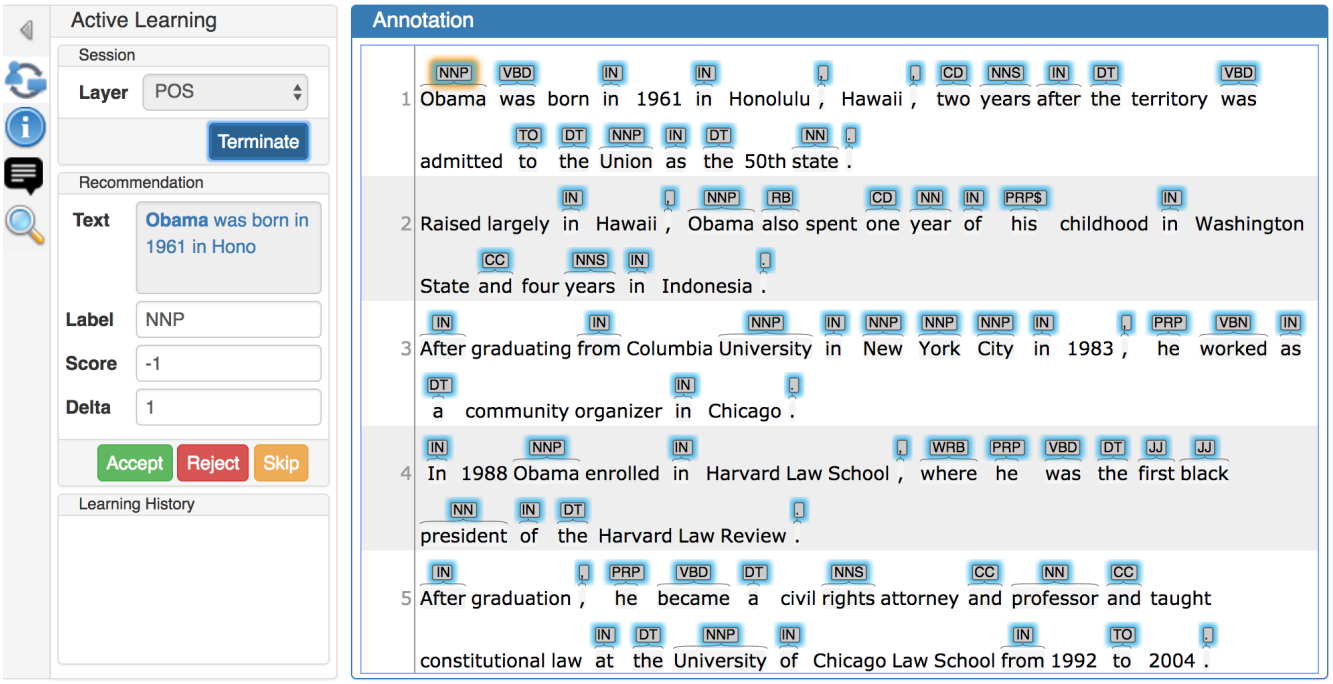


One can now click the active learning icon on the left side and the Active Learning sidebar shows up . One can now select the layer like POS layer for annotation and click *Start* for starting an active learning session:

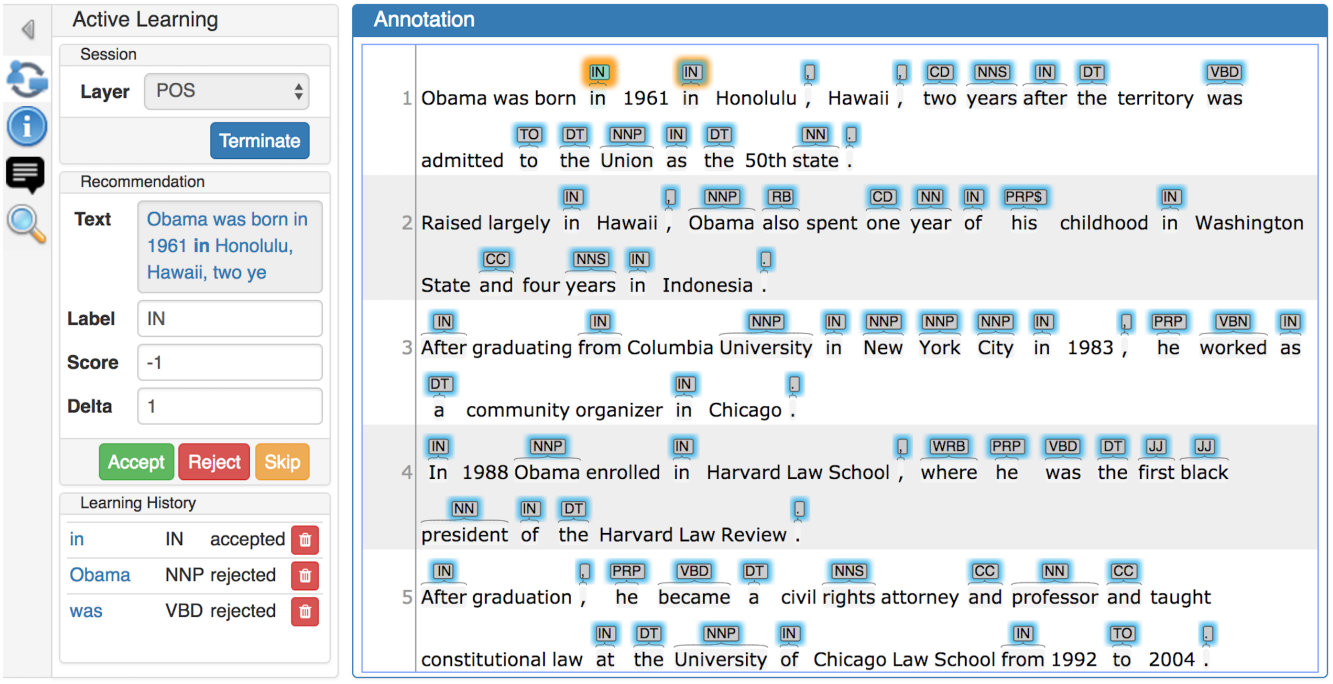


The Active Learning sidebar will then start showing recommendations, one by one, according to the *uncertainty sampling* learning strategy. For every recommendation, it shows the related text, the suggested annotation, the confidence score and a delta that represents the difference between the given score and the closest score calculated for another suggestion made by the same recommender to that text. The recommendation is also highlighted in the central annotation editor.

One can now *Accept*, *Reject* or *Skip* this recommendation in the Active Learning sidebar:



The acceptance, rejection or skipping will be recorded and displayed in the learning history of the Active Learning sidebar. After a suggestion is accepted, the text is annotated with that recommended annotation. If the user rejects a suggestion, the recommendation is deleted. Finally, if the user skips a suggestion, that recommendation will continue being shown in the central annotation editor. Eventually, it could be shown again at the end of the active learning session, when there are no more undealt suggestions.



After the user takes an action on the current suggestion, the next recommendation will then show up in the sidebar, and the central annotation editor will jump to its corresponding location (which could sometimes be in another document).

The learning history contains a log of all the actions that were taken by the user regarding the suggestions given by the recommenders (acceptances, rejections and skippings). Clicking on an item of the learning history will make the central editor jump to the corresponding place in the

document . Any entry in the learning history can be deleted by clicking the corresponding trash bin icon. If this learning history is a valid acceptance, after the learning record is deleted, a confirmation dialogue box pops up to confirm whether to delete the annotation too.

The user may finish the current active learning session whenever he wants. If there are pending suggestions, they might be shown in the next active learning session that he starts.

Concept Linking

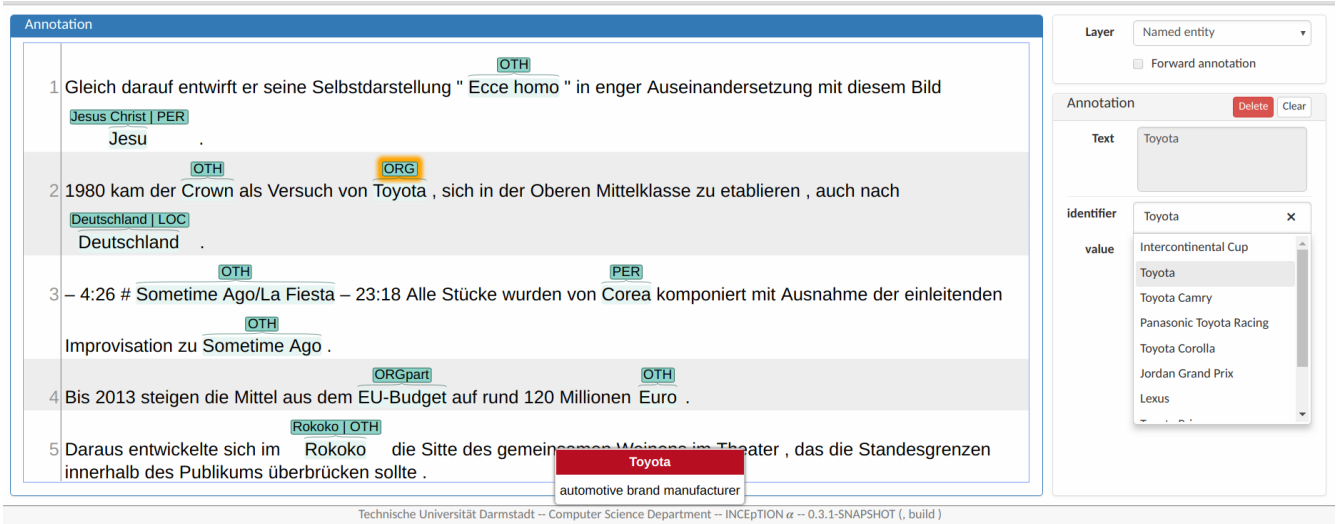
Concept Linking is the task of identifying concept mentions in the text and linking them to their corresponding concepts in a knowledge base. Use cases of Concept Linking are commonly found in the area of biomedical text mining, e.g. to facilitate understanding of unexplained terminology or abbreviations in scientific literature by linking biological entities.

Contextual Disambiguation

Like many words, concept names can be ambiguous. There can be potentially many different concepts having the same name (consider the large number of famous people called John Smith). Thus, it is helpful to rank the candidates before showing them to the user in the annotaton interface. If the ranking works well, the user can quickly choose on of the top-ranking candidates instead of having to scroll through a long list.

The approach we are using for ranking the candidates considers the context of the concept mention in the text as well as the context of the candidate concepts in the knowledge base. This allows a ranking to be performed even in projects where no concepts have been linked yet (i.e. there is no training data that could be used for a machine learning classifier).

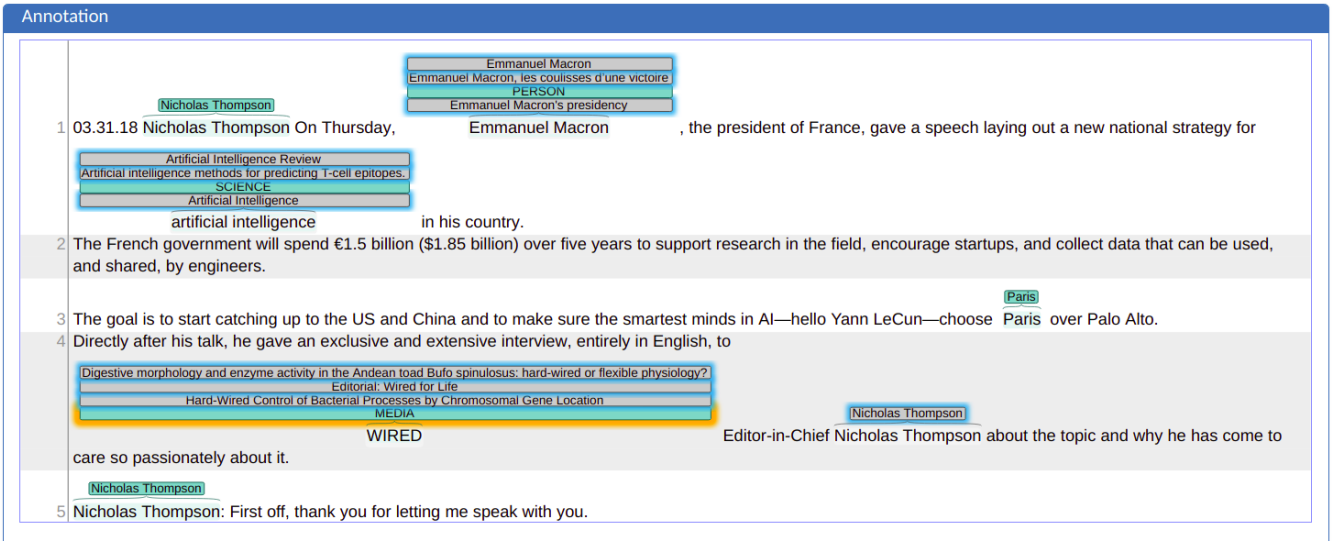
To link a concept mention to the knowledge base, first select the mention annotation, then select the concept feature in the right sidebar of the annotation editor and start typing the name of a concept. A ranked list of candidates is then displayed in the form of a drop-down menu. In order to make the disambiguation process easier, descriptions are shown for each candidate.



The suggestions are updated every time it receives new input.

Automated Concept Suggestions

The Named Entity Linker (NEL) displays three highest-ranked candidates as suggestions boxes over each mention annotated as Named Entity. The user can accept, reject or ignore these suggestions. If a suggestion is rejected, it is not showed again. It is possible to combine the NEL with the existing Named Entity Recommenders for the NE type, which makes the annotation process even faster. The recommender needs to be set up in the Project Settings.



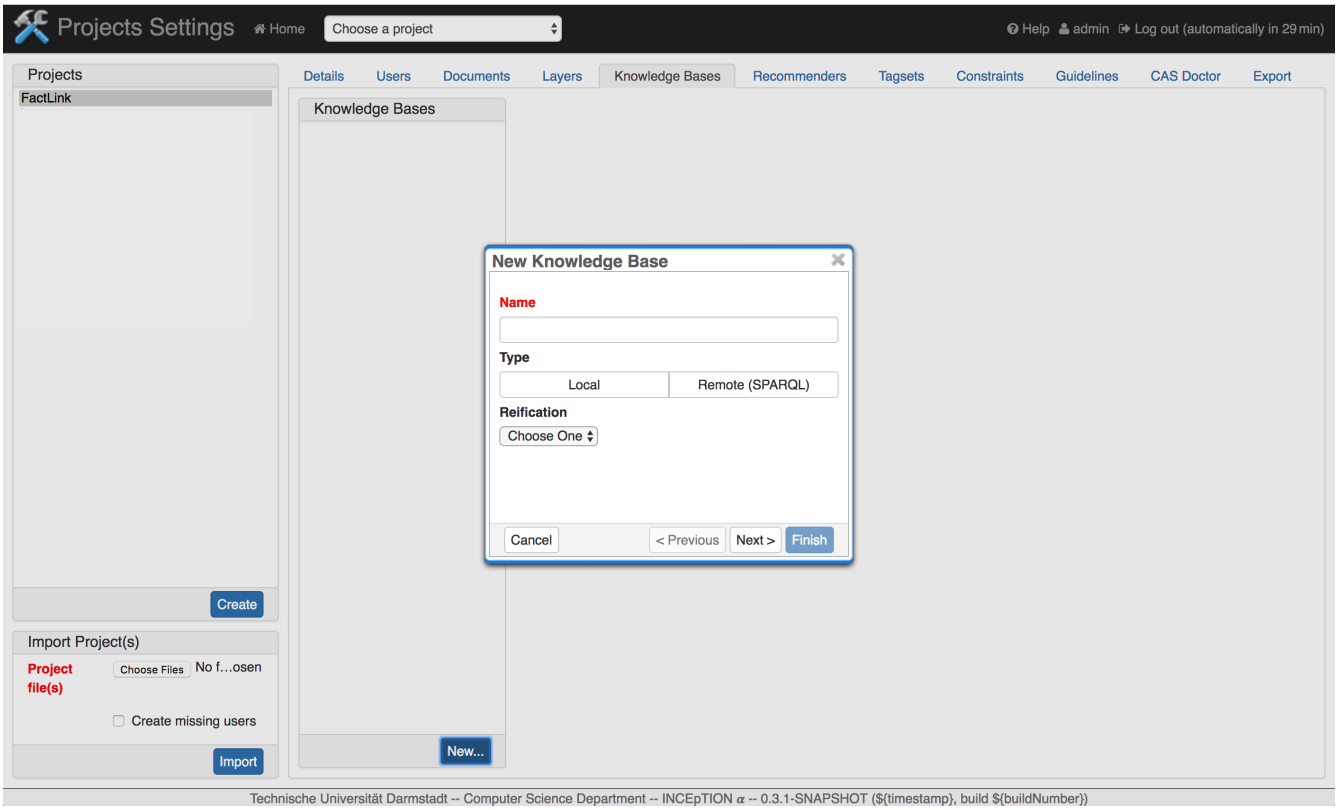
Fact Extraction

Fact extraction is the task of extracting facts from documents. It is defined that a fact consists of a subject, a predicate, an object and qualifiers. Fact extraction in INCEpTION includes annotating the mentions for each fact component, and linking these mentions to instances of concept classes or properties from the knowledge base.

This section briefly describes how to use the fact extraction functionality in INCEpTION alongside a running example. The example covers creating a local knowledge base supporting qualifiers in the **Projects Settings**, managing qualifiers in the **Knowledge Base**, and extracting a fact in the **Annotation** page.

A local knowledge base supporting qualifiers

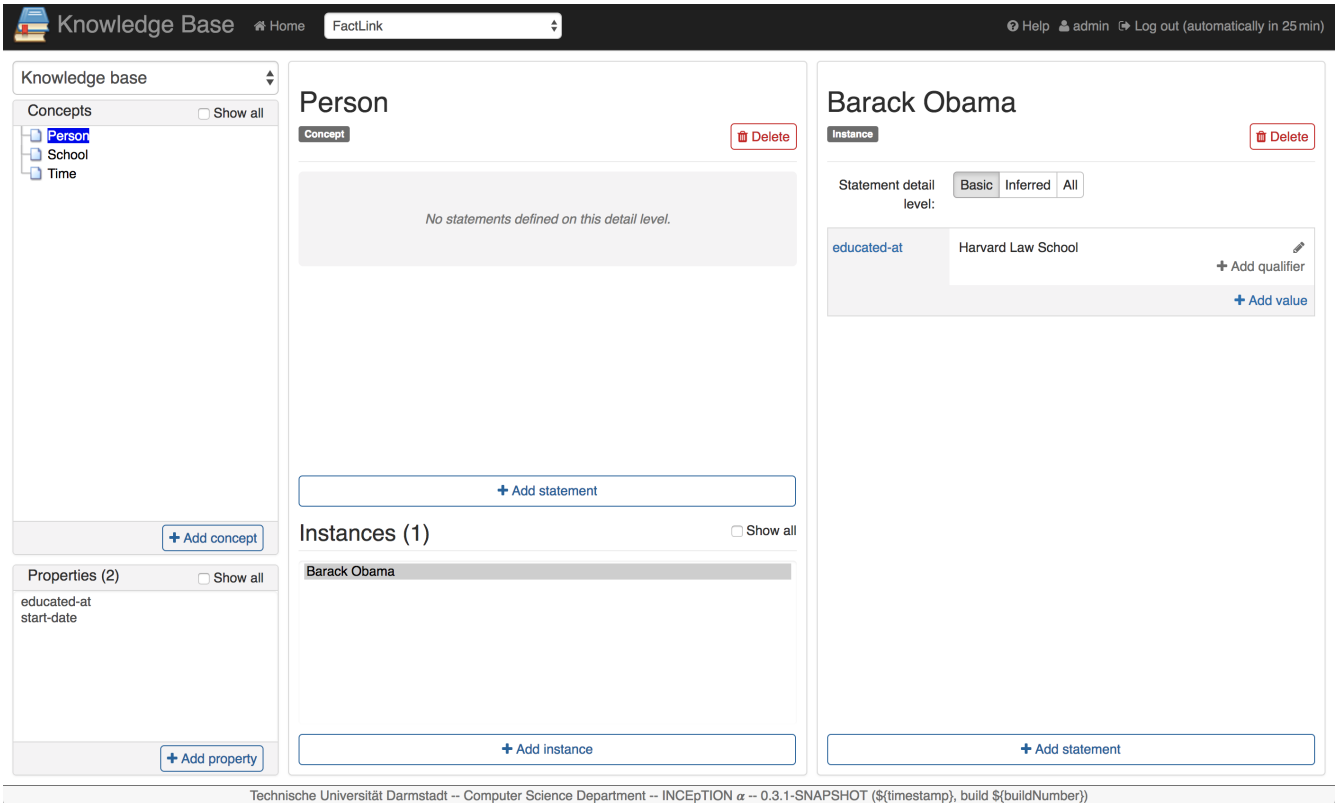
In the **Projects Settings**, switch to the **Knowledge Bases** tab, then click **New...** on the bottom and a dialogue box shows as in the figure below.



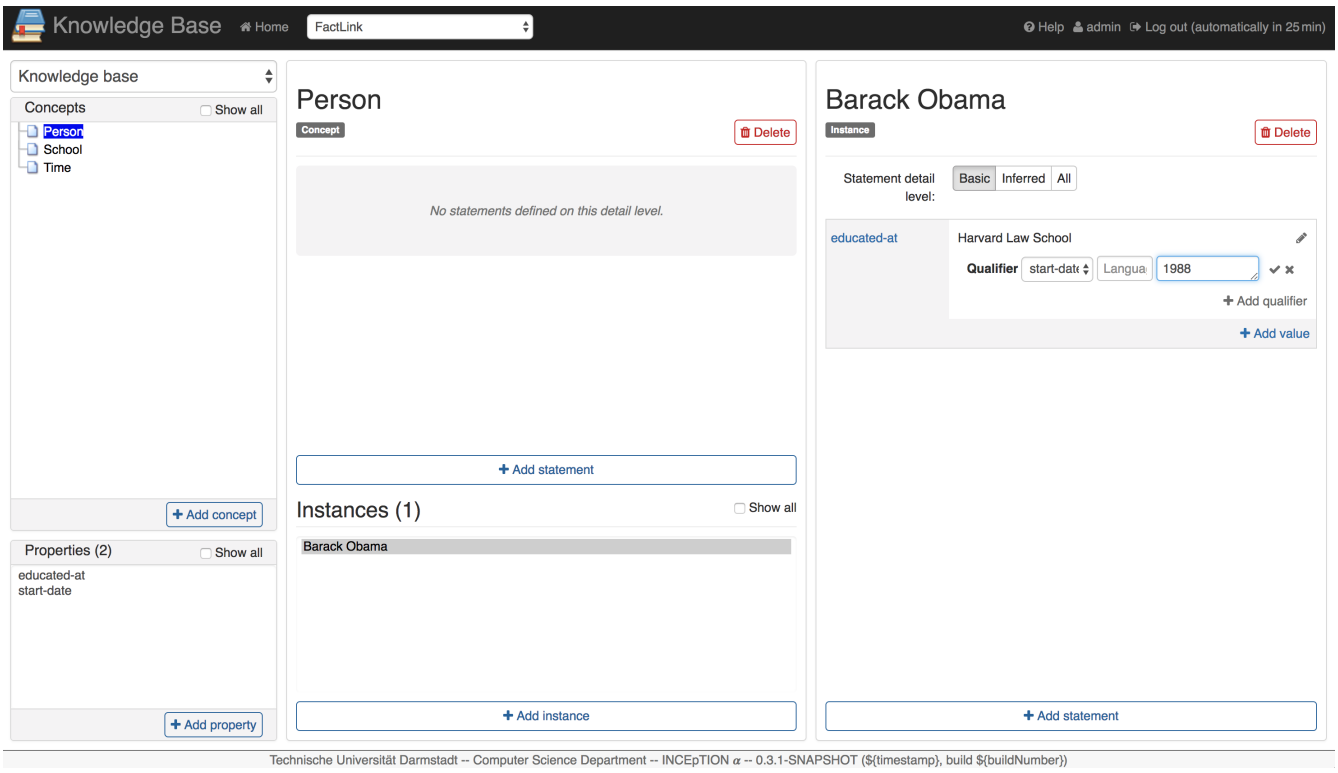
To create a local knowledge base, one needs to choose **Local** for the type. For the reification, **NONE** is the defaut case, but one cannot add or view qualifiers in the knowledge base with **NONE**. So, to support qualifiers, one needs to choose **WIKIDATA** for the Reification. One can then follow the wizard to finish the setting.

Managing qualifiers in the knowledge base

Assuming one has already created concepts, properties, instances and a statement about instance *Barack Obama* in this knowledge base:



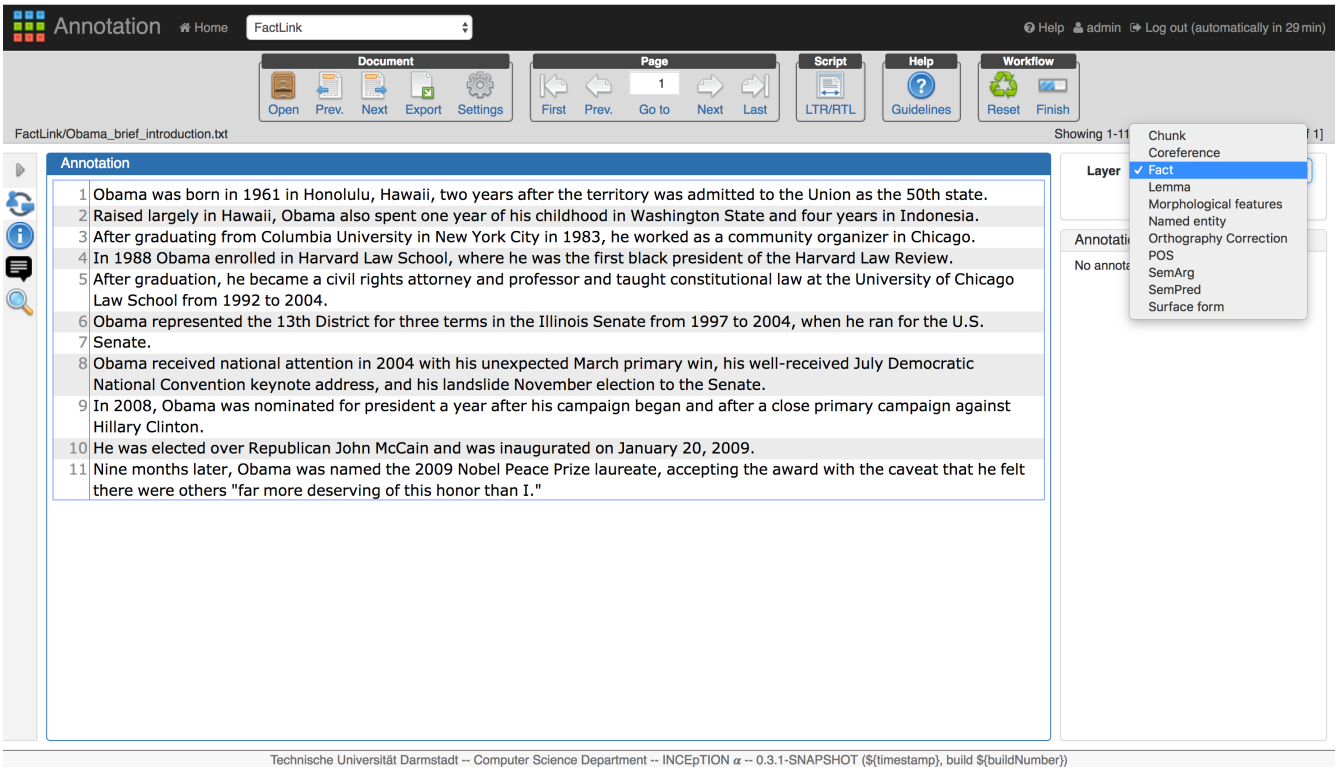
One can click the button + **Add qualifier**, choose a property *start-date* for this qualifier name, enter *1988* for this qualifier value, and click √ to save this qualifier.



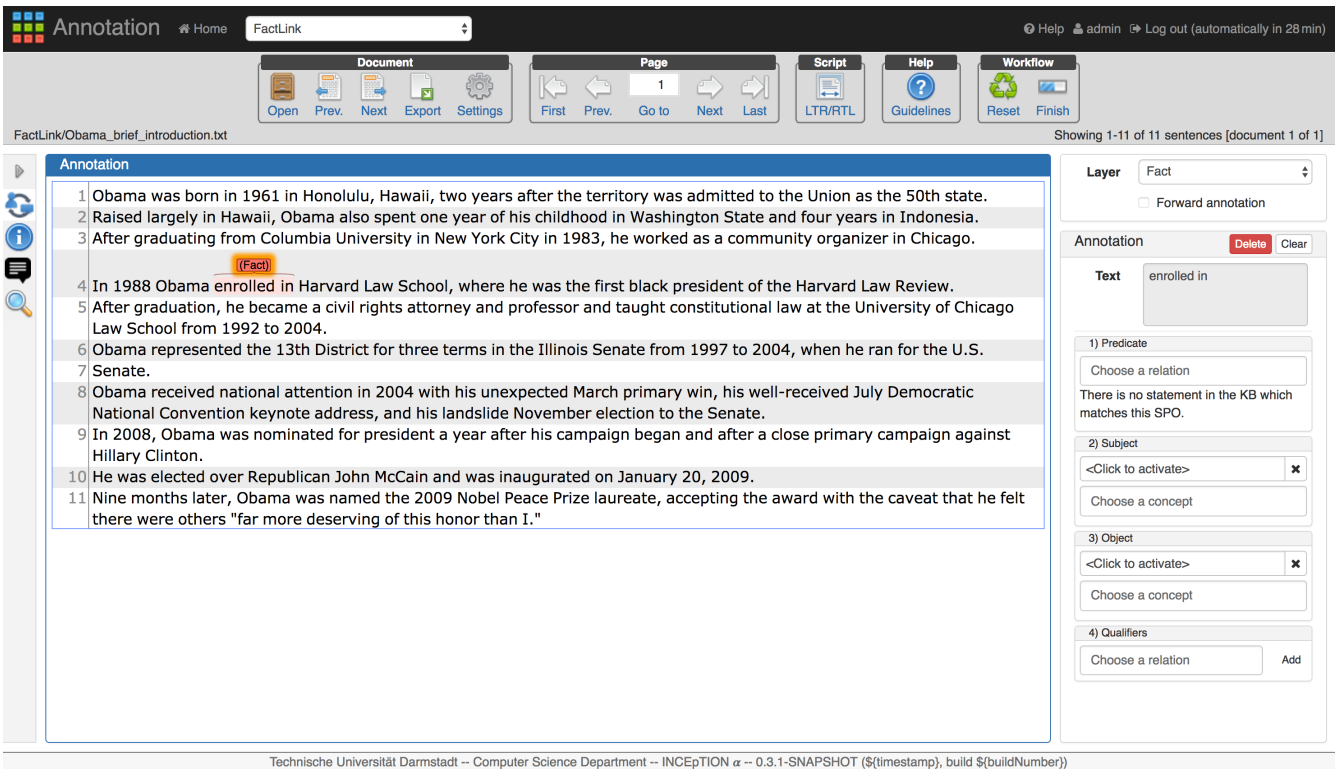
One can click the pencil icon on the same line as this qualifer to edit and delete this qualifier.

Linking a fact in the annotation page

Assuming the project contains documents for annotation, one can now switch to the annotation page and choose the fact layer:

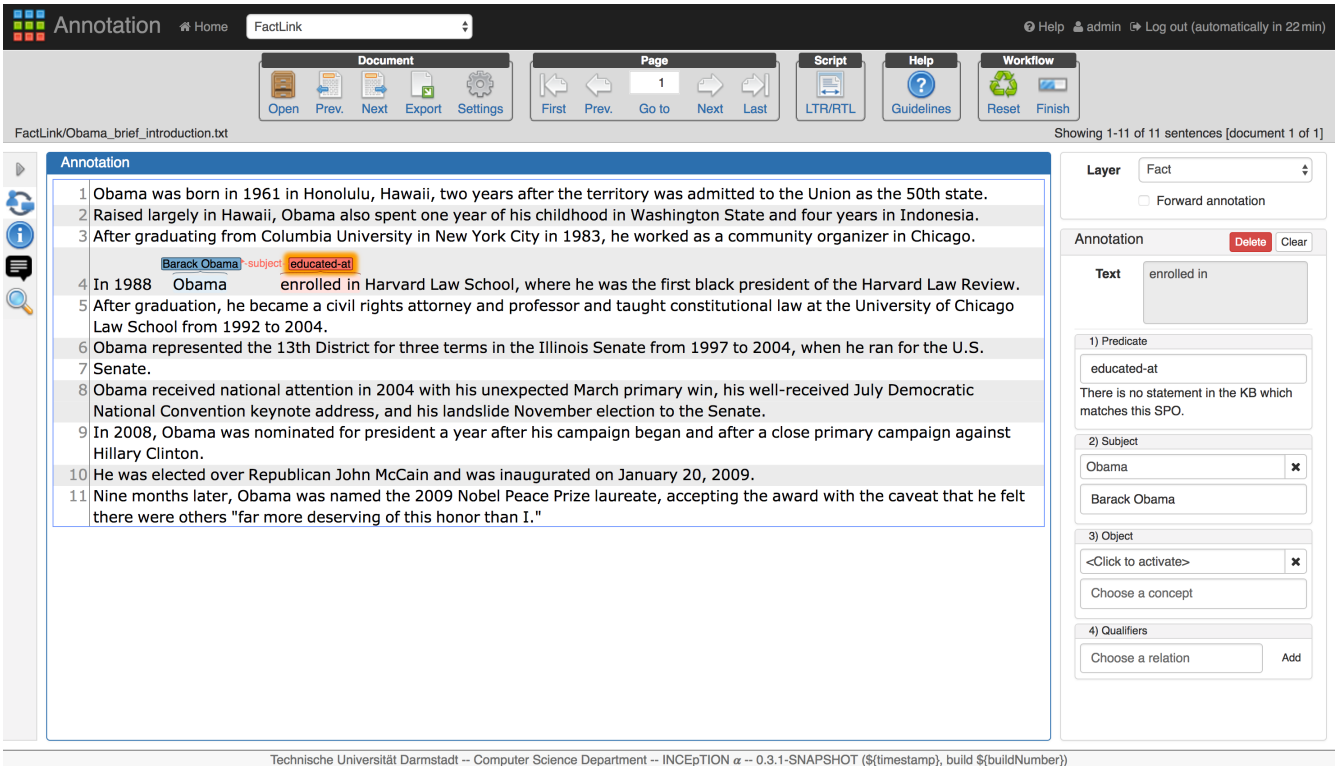


One can now mark the predicate mention of a fact. In the screenshot below, *enrolled in* is selected. The right sidebar shows feature editors for the predicate, the subject, the object and the qualifiers. One can then choose *educated-at* in the dropdown field of the predicate feature editor to link the mention of this predicate to the property in the knowledge base. The candidate list of this dropdown field is a list of properties in the knowledge base.



To annotate the subject mention of a fact, one needs to click <**Click to activate**> in the subject feature editor. After this input field turns orange, one can now mark the subject mention *Obama_of a fact. One can then choose _Barack Obama* in the dropdown field of the subject feature editor to link the mention of this subject to the instance in the knowledge base. The candidate list of this

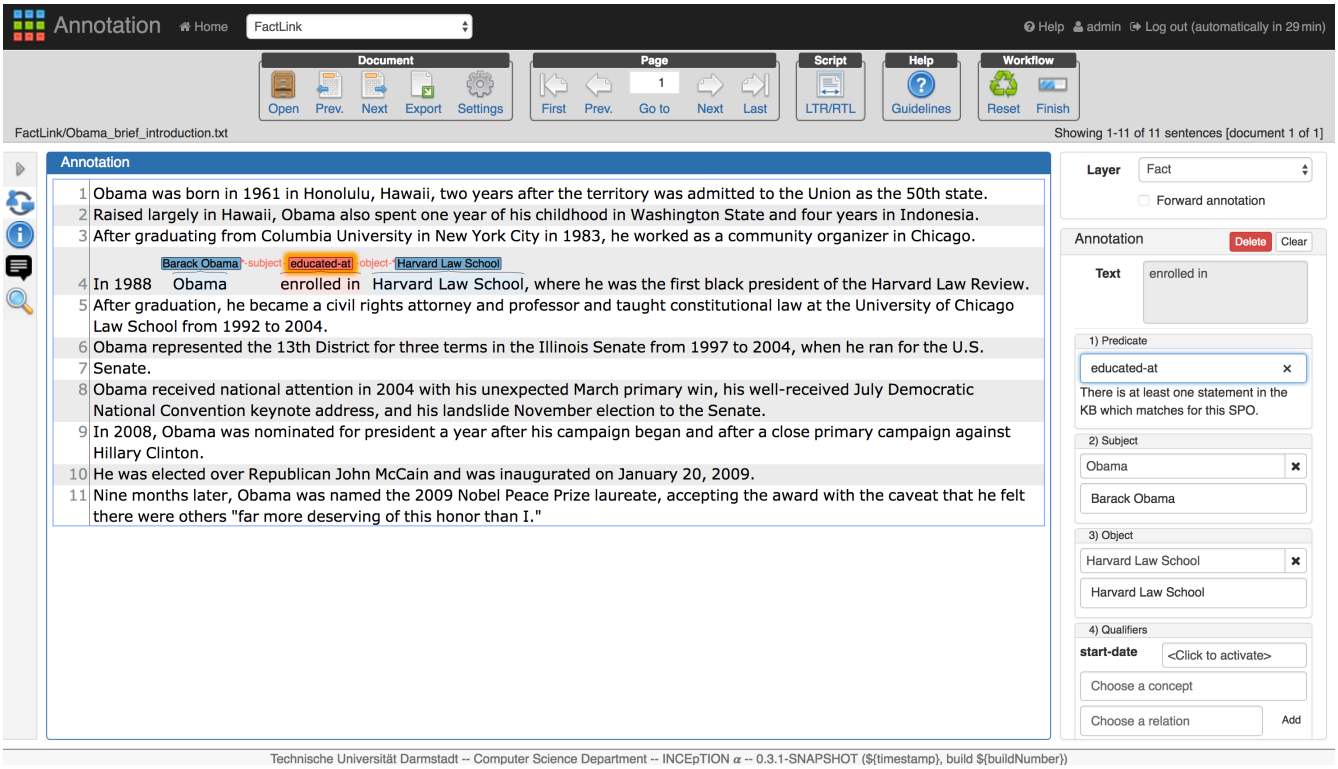
dropdown field is a list of instances in the knowledge base. The default setting returns all the instances from all the knowledge base in this project. See Fact linking with multiple knowledge bases to select a specific knowledge base. After linking the subject to the knowledge base instance, in the main editor, there is an arrow from the label above the predicate to the label above the subject, with a name *subject*.



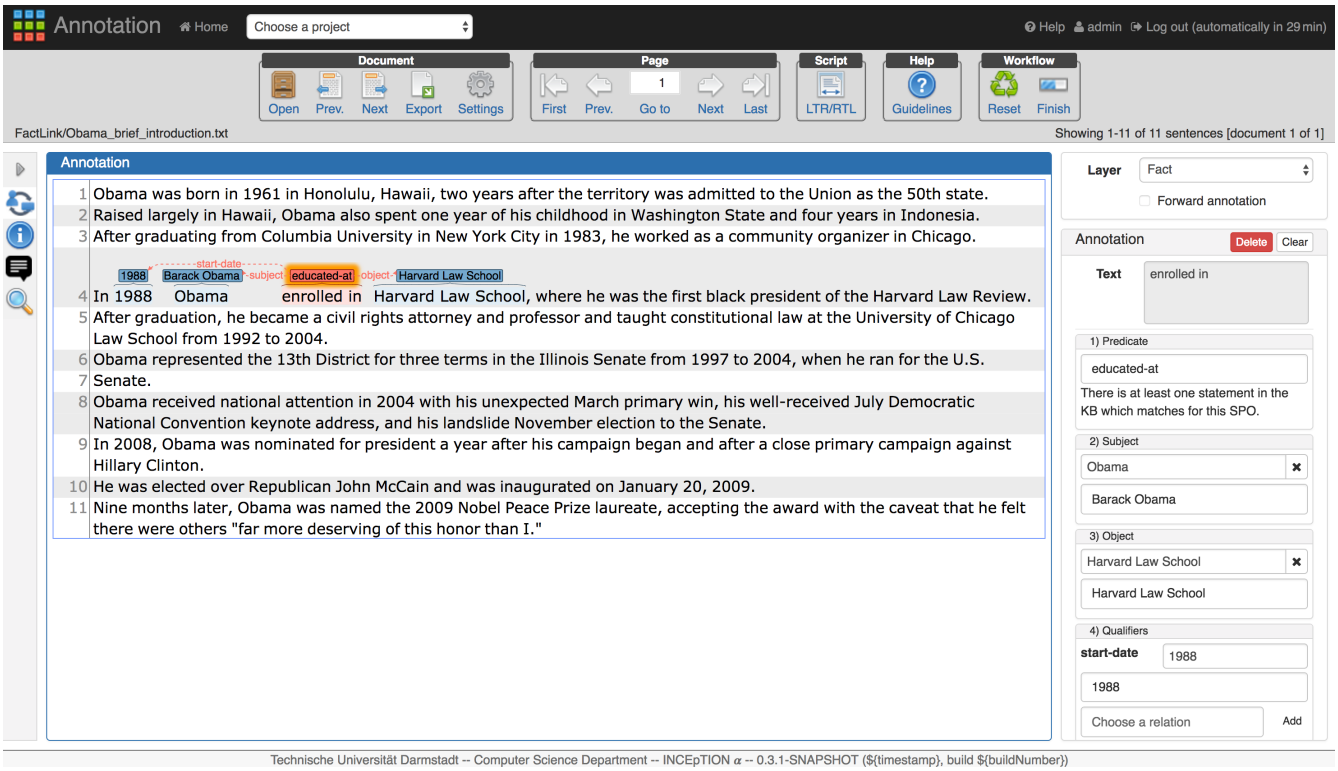
Extracting the object mention is same as extracting the subject.

If this fact is already saved in the knowledge base, the label in the predicate feature editor shows **There is at least one statement in the KB which matches for this SPO**.

To extract a qualifier, one needs to choose the qualifier name from the drop-down field in front of the button **add** in the qualifier feature editor. The list of candidates for the qualifier name is a list of properties from the knowledge base. After clicking **add**, a mention input field and a dropdown field appear to collect this qualifier value information. In the screenshot below, the qualifier name is *start-date*.



One can then click to activate the mention input, annotate the mention, and choose an instance from the dropdown field to link the value of this qualifier to the knowledge base. The list of candidates for the qualifier value is a list of instances from the knowledge base.

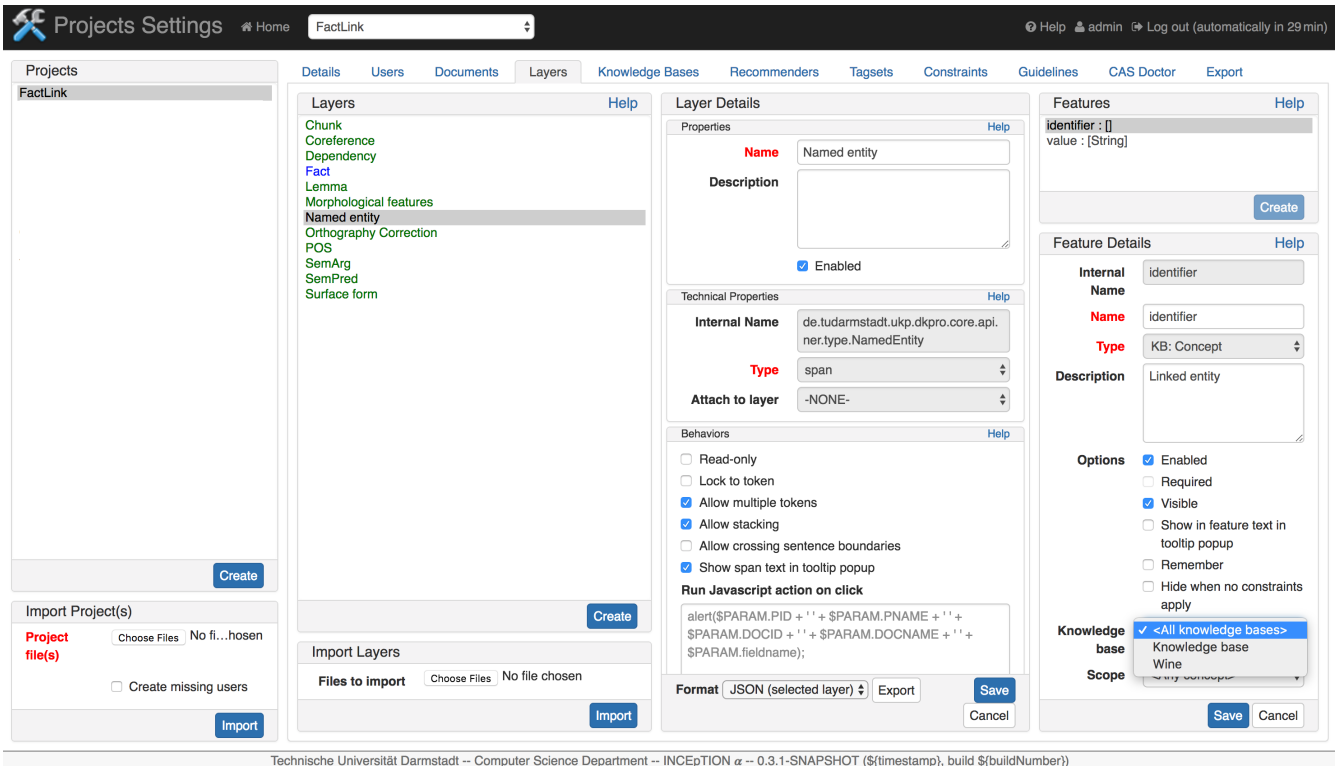


After annotating this qualifer, in the main editor, there is an arrow from the label above the predicate to the label above the qualifier value, with the qualifier name *start-date*.

So, a fact (Barack Obama, educated-at, Harvard Law School, Start-date: 1988) with its mentions is linked in this example.

Fact linking with multiple knowledge bases

If a project has multiple knowledge bases, a user can choose to link the mention to a certain knowledge base or to all knowledge bases. This configuration is done in the **Projects Settings**. One needs to switch to the **Layers** tab first, then to choose the **Named entity** layer and the **identifier** feature. After that one can configure the linked knowledge base information in **Feature Details**, choose the desired knowledge base from the dropdown list of the field **Knowledge base** as shown in the figure below.



Curation

This functionality is only available to **project managers** (managers of existing projects), **curators**, and **administrators**. Curators and project managers only see projects in which they hold the respective roles.

When navigating to the **Curation Page**, the procedure for opening projects and documents is the same as in Annotation. The navigation within the document is also equivalent to Annotation.

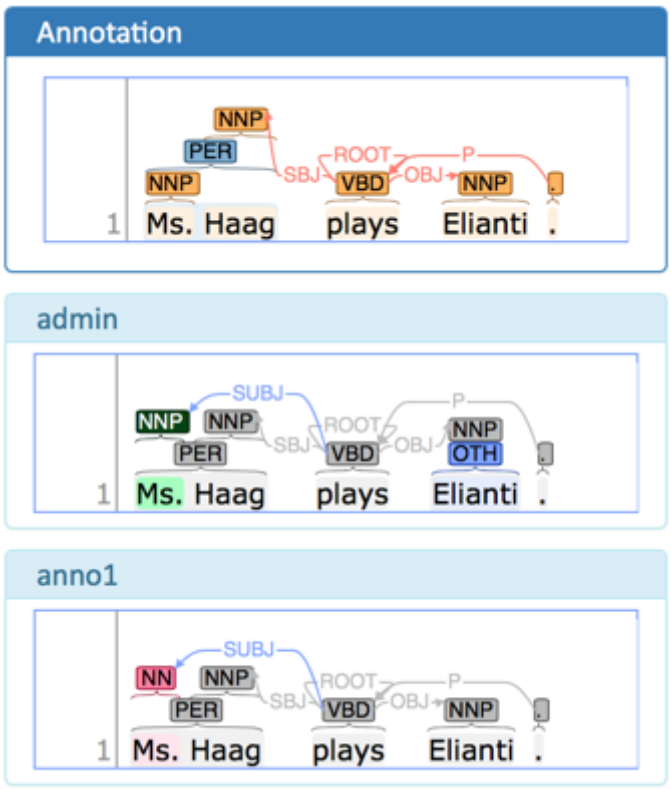
Table 6. Explanation of the project colors in the curation open document dialog

No curatable documents	Red
Curatable documents	Green

Table 7. Explanation of the document colors in the curation open document dialog

New	Black
Annotation in progress	Black
Curation in progress	Blue
Curation finished	Red

In the left frame of the window, named **Sentences**, an overview of the chosen document is displayed. Sentences are represented by their number inside the document. Sentences containing a disagreement between annotators are colored in red. Click on a sentence in order to select it and to to edit it in the central part of the page.



The center part of the annotation page is divided into the **Annotation** pane which is a full-scale annotation editor and contains the final data from the curation step. Below it are multiple read-

only panes containing the annotations from individual annotators. Clicking on an annotation in any of the annotator's panes transfers the respective annotation to the **Annotation** pane.

When a document is opened for the first time in the curation page, the application analyzes agreements and disagreemens between annotators. All annotations on which all annotators agree are automatically copied to the **Annotation** pane. Any annotations on which the annotators disagree are skipped.

The annotator's panes are color-coded according to their relation with the contents of the **Annotation** pane and according to the agreement status. If the annotations were the same, they are marked **grey** in the lower panes. If the annotations are disparate, the markings are dark blue in the lower frames. By default, they are not taken into the merged file. If you choose one annotation to be right by clicking on it, the chosen annotation will turn green in the frame of the corresponding annotator. Also, the annotation will say **USE** next to the classification.

Note that the **Annotation** pane is not color-coded. It uses whatever coloring strategy is configured in the **Settings** dialog.

The annotations which were not chosen to be in the merged file are marked dark blue. The annotations which were wrongly classified are marked in red.

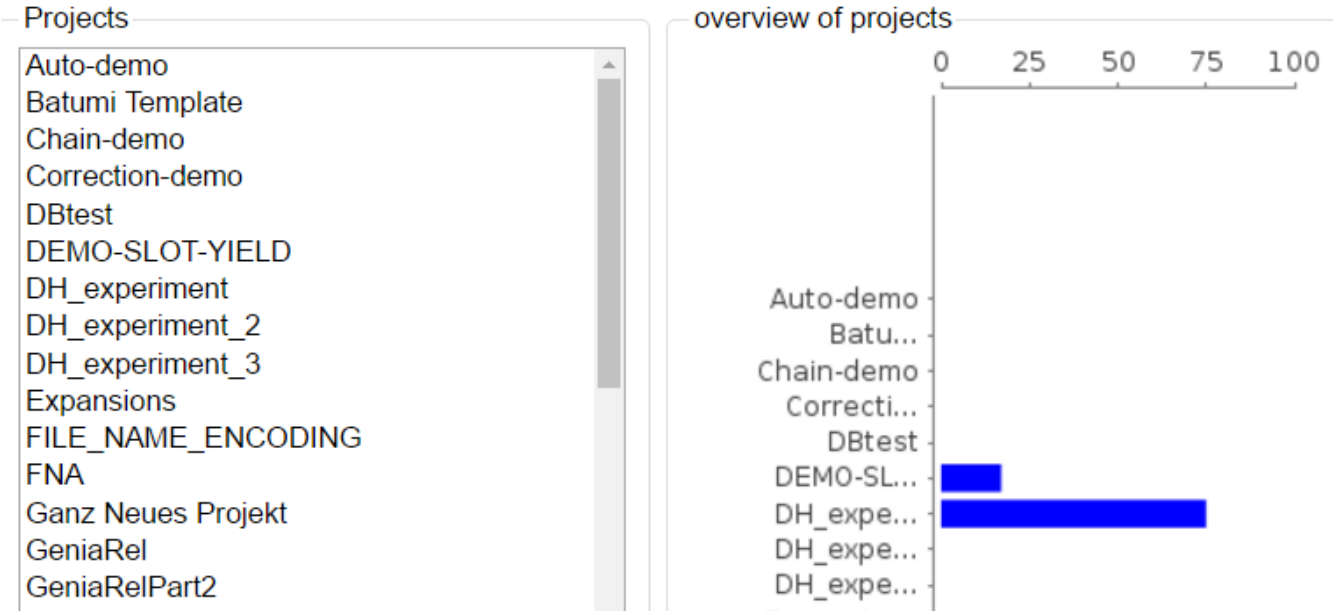
Table 8. Explanation of the annotation colors in the annotator's panes (lower panes)

Grey	all annotators agree
Blue	disagreement requiring curation; annotators disagree and there is no corresponding annotation in the upper **Annotation** pane yet
Green	accepted; matches the corresponding annotation in the upper **Annotation** pane
Red	rejected; different to the corresponding annotation in the upper **Annotation** pane

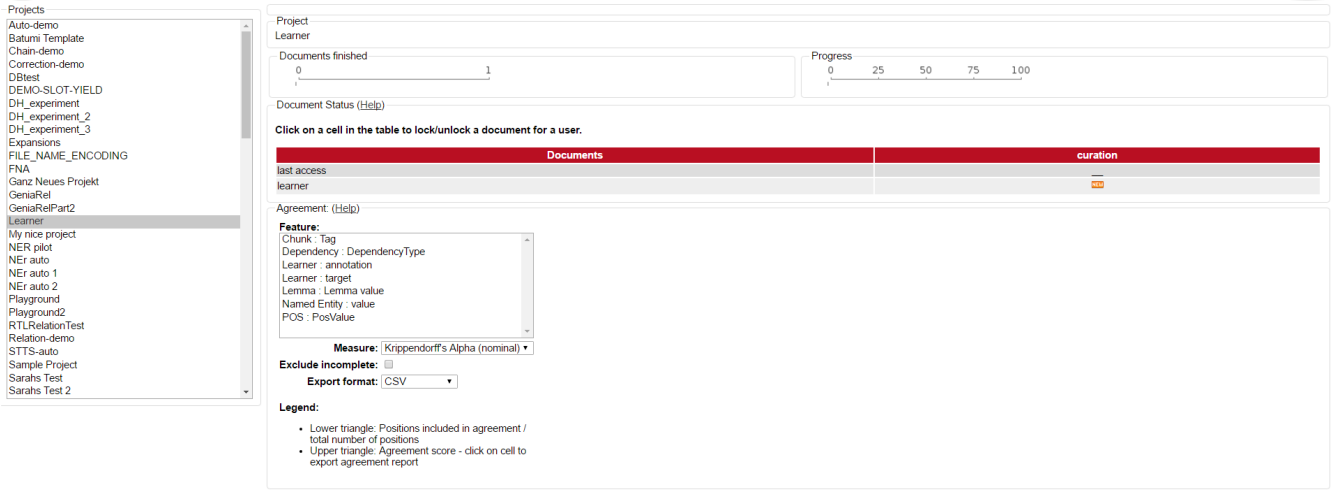
Monitoring

This functionality is only available to **project managers** (managers of existing projects), **curators**, and **administrators**. Curators and project managers only see projects in which they hold the respective roles.

As an administrator, you are able to observe the progress and document status of projects you are responsible for. Moreover, you are able to see the time of the last login of every user and observe the agreement between the annotators. After clicking on **Monitoring** in the main menu, the following page is displayed:



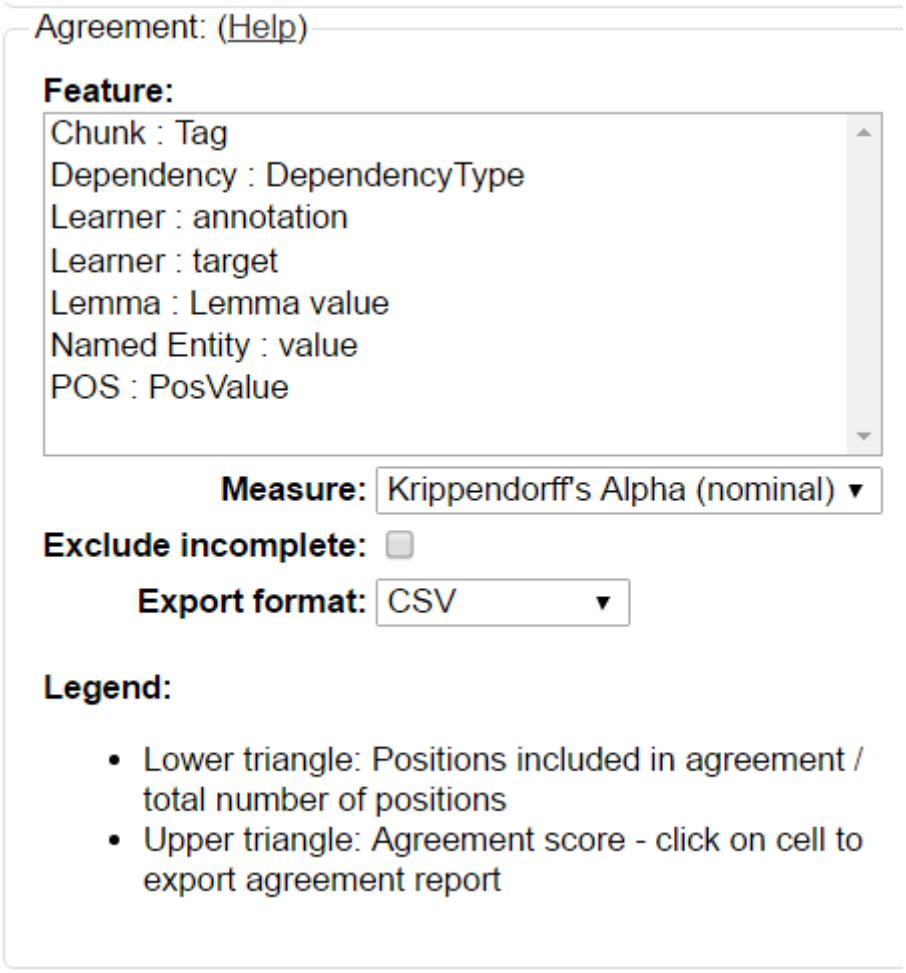
In the right frame, the overall progress of all projects is displayed. In the left frame one sees all projects, that one has an administrator role in. By clicking on one of the projects on the left, it may be selected and the following view is opened:



The percentual progress out of the workload for individual annotators may be viewed as well as the number of finished documents.

Below the document overview, a measuring for the inter-annotator-agreement can be selected by opening the **Measure** dropdown menu. Three different units of measurement are possible: Cohen's

kappa as implemented in DKPro Statistics, Fleiss' kappa and Krippendorff's alpha. Below the **Measure** dropdown menu, an export format can be chosen. Currently, only CSV format is possible.



Above the **Measure** dropdown menu, the **Feature** box allows the selection of layers for which an agreement shall be computed. Double-clicking on a layer starts the computation of the agreement and an outline is shown to the left side of the box:

	darina	rebekka
darina	-	no positions
rebekka	0/0	-

The following table explains the different symbols which explain the status of a document for a user and the described task.

Table 9. Document Status

Symbol	Meaning
	Annotation has not started yet
	Document not available to user
	Annotation is in progress
	Annotation is complete
	Curation is in progress

You can also alter the document status of annotators. By clicking on the symbols you can change

between **Done** and **In Progress**. You can also alter between **New** and **Locked** status. The second column of the document status frame displays the status of the curation.

As there is only one curator for one document, curation is not divided into individual curators.

Scrolling down, two further frames become visible. The left one, named **Layer**, allows you to chose a layer in which pairwise kappa agreement between annotators will be calculated.

Agreement

Agreement can be inspected on a per-feature basis and is calculated pair-wise between all annotators across all documents.

The first time a feature is selected for agreement inspection, it takes a moment to calculate the differences between the annotated documents. Switching between different features subsequently is much faster.

Agreement is calculated in two steps:

1. **Generation of positions and configuration sets** - all documents are scanned for annotations and annotations located at the same positions are collected in configuration sets. To determine if two annotations are at the same position, different approaches are used depending on the layer type. For a span layer, the begin and end offsets are used. For a relation layer, the begin and end offsets of the source and target annotation are used. Chains are currently not supported.
2. **Calculation of pairwise agreement** - based on the generated configuration sets, agreement is calculated. There are two cases where a configuration set may be omitted from the pairwise agreement calculation:
    a. one of the users did not make an annotation at the position;
    b. one or both of the users did not assign a value to the feature on which agreement is calculated at the position.

The lower part of the agreement matrix displays how many configuration sets were used to calculate agreement and how many were found in total. The upper part of the agreement matrix displays the pairwise Cohen's kappa scores.

The agreement calculations considers an unset feature (with a `null` value) to be equivalent to a feature with the value of an empty string. Empty strings are considered valid labels and are not excluded from agreement calculation.

Annotations for a given position are considered complete when both annotators have made an annotation. Unless the agreement measure supports `null` values (i.e. missing annotations), incomplete annotations are implicitly excluded from the agreement calculation. If the agreement measure does support incomplete annotations, then excluding them or not is the users' choice.

Table 10. Possible combinations for agreement

Feature value annotator 1	Feature value annotator 2	Agreement	Complete
X	X	yes	yes
X	Y	no	yes
no annotation	Y	no	no
empty	Y	no	yes
empty	**empty**	yes	yes

Feature value annotator 1	Feature value annotator 2	Agreement	Complete
null	**empty**	yes	yes
empty	**no annotation**	no	no

Multiple interpretations in the form of stacked annotations are not supported in the agreement calculation! This also includes relations for which source or targets spans are stacked.

Knowledge Base

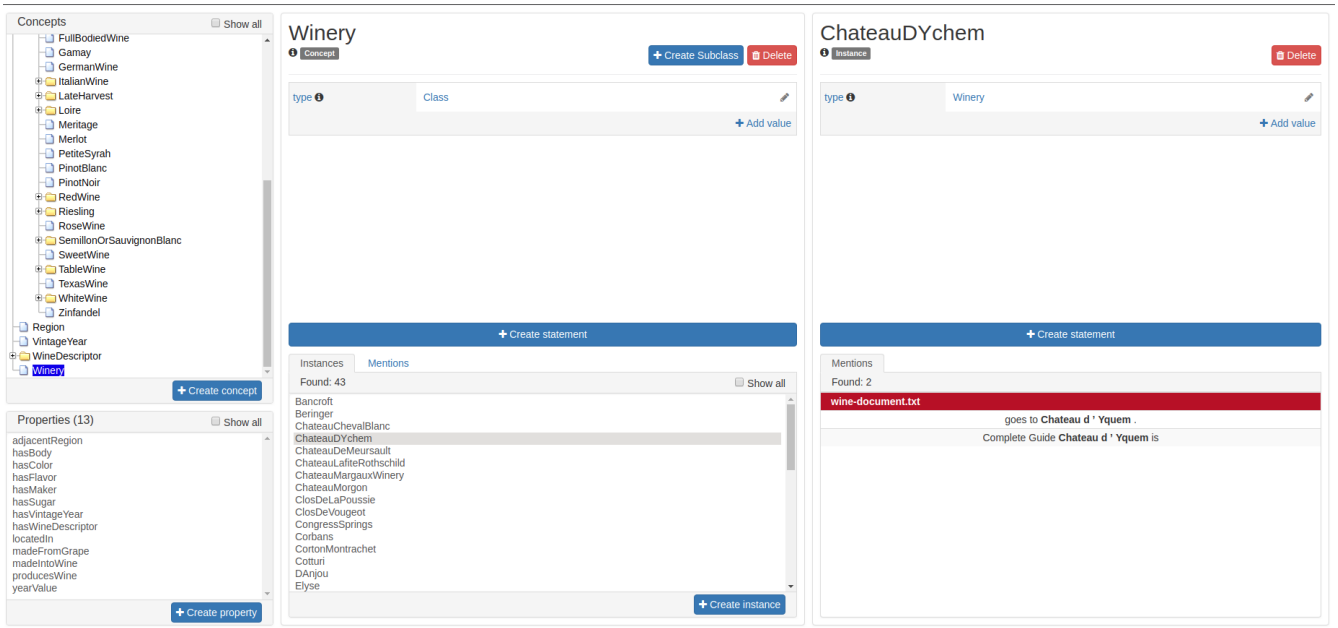
The knowledge base (KB) module of INCEpTION enables the user to create a KB from scratch or to import it from an RDF file. Alternatively, the user can connect to a remote KB using SPARQL. However, editing the content of remote KBs is currently not supported. This knowledge base can then be for instance used for entity or fact linking and knowledge base population.

This section briefly describes how to set up a KB in the KB management page on **Projects Settings**, explains the functionalities provided by the **Knowledge Base** page and covers the **concept** and **property** feature types.

Knowledge Base Page

The knowledge base page provides a concept tree hierarchy with a list of instances and statements, together with the list of properties as shown in the figure below. For local knowledge bases, the user can edit the KB contents here, which includes adding, editing and deleting concepts, properties, statements and instances.

The knowledge base page provides the specific mentions of concepts and instances annotated in the text in the **Mentions** panel which integrates the knowledge base page with the annotated text.



The concept tree in this page is designed using the **subClass** relationship for the configured mapping. Each concept associates itself with a list of instances (in case it has one) on the **Instance** panel which appear when we click on a specific concept along with the **Mentions** of the concept in the annotated text. The click on a specific instance shows the panel for the list of statements associated with the instance along with **Mentions** of the instance in the annotated text. In the left bottom side of the page, it lists the set of properties from the knowledge base. Clicking on the property showcases the statements associated with the property such as labels, domains, ranges, etc.

In case the user has the privilege to edit the knowledge base, the user may add statements for concepts, instances and properties.

Statement editors

INCEpTION allows the user to edit local knowledge bases. This includes adding statements or subclassing concepts and their instances.

In order to create a statement for a particular knowledge base entity, the **Create Statement** can be used.

When creating a new statement about an instance, a list of available properties is shown. After selecting the property of choice, the object of the statement has to be specified. The possible properties for a given subject are restricted by domain the domain of property, i.e. the property **born_in** would need an instance of **human** as the subject.

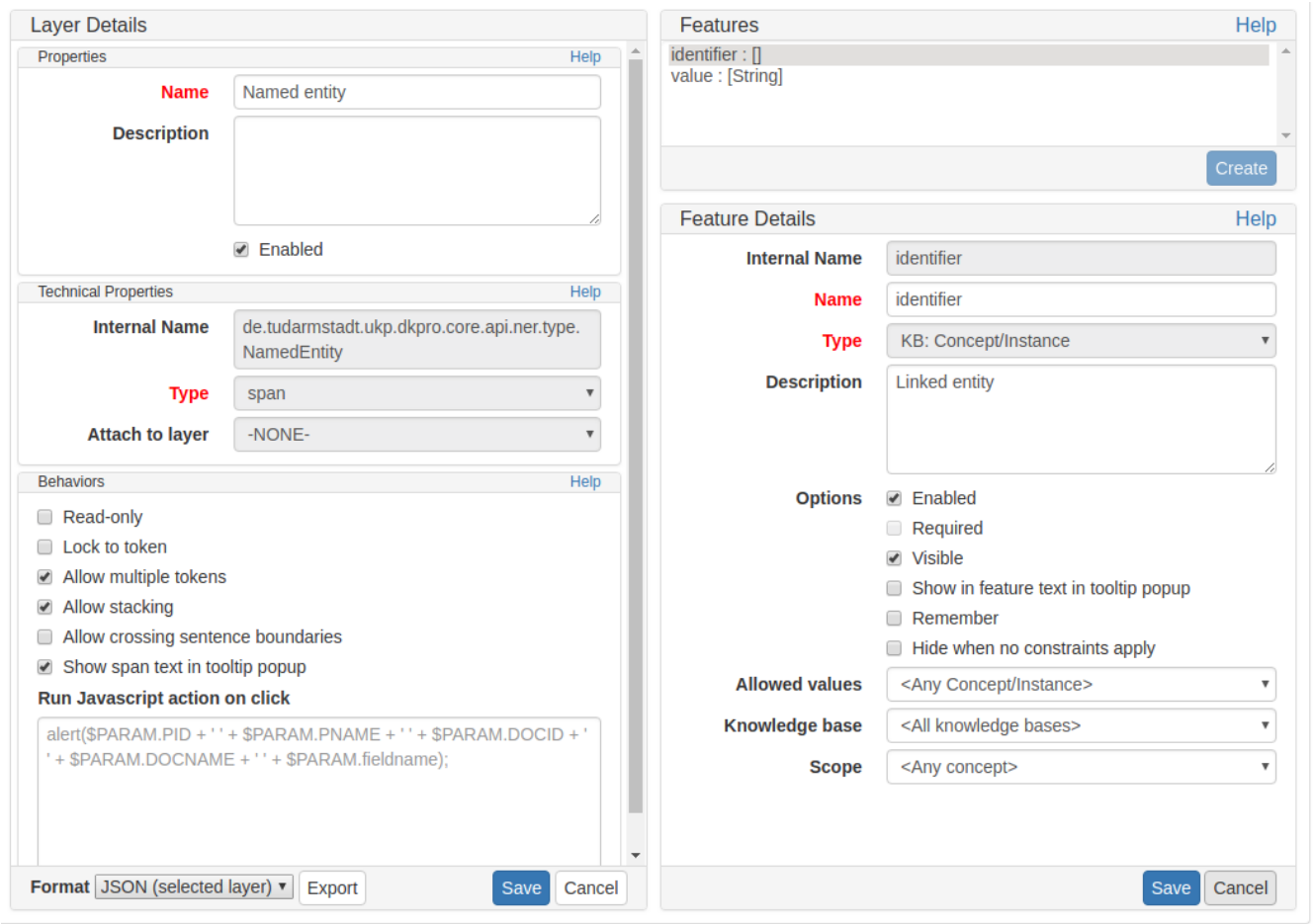
The same is true for the object of a statement: After choosing the property for a concept, the object has to be specified. The possible objects are limited by the range of the property if given. Right now, four different editors are available to specify features for:

1. **Boolean**: Allows either **true** or **false**
2. **Numeric**: Accepts integers or decimals
3. **String**: String with a language tag or an URI identifying a resource that is not in the knowledge base
4. **KB Resource**: This is provided as an option when the property has a range as a particular concept from the knowledge base. In this option, the user is provided with an auto-complete field with a list of knowledge base entities. This includes the subclass and instances of the range specified for the property.

Concept features

Concept features are features that allow referencing concepts in the knowledge base during annotation.

To create a new concept feature, a new feature has to be created under **Projects Settings → Layers**. The type of the new feature should be **KB: Concept/Instance**. Features of this type also can be configured to either take only concepts (select **Only Concept**), only instances (select **Only Instances**) or both (select **Any Concept/Instance**).



When creating a new annotation with this feature, then the user is offered a dropdown with possible entities from the knowledge base. This dropdown is then limited to only concepts or features or both when selecting the respective filter in the feature configuration.

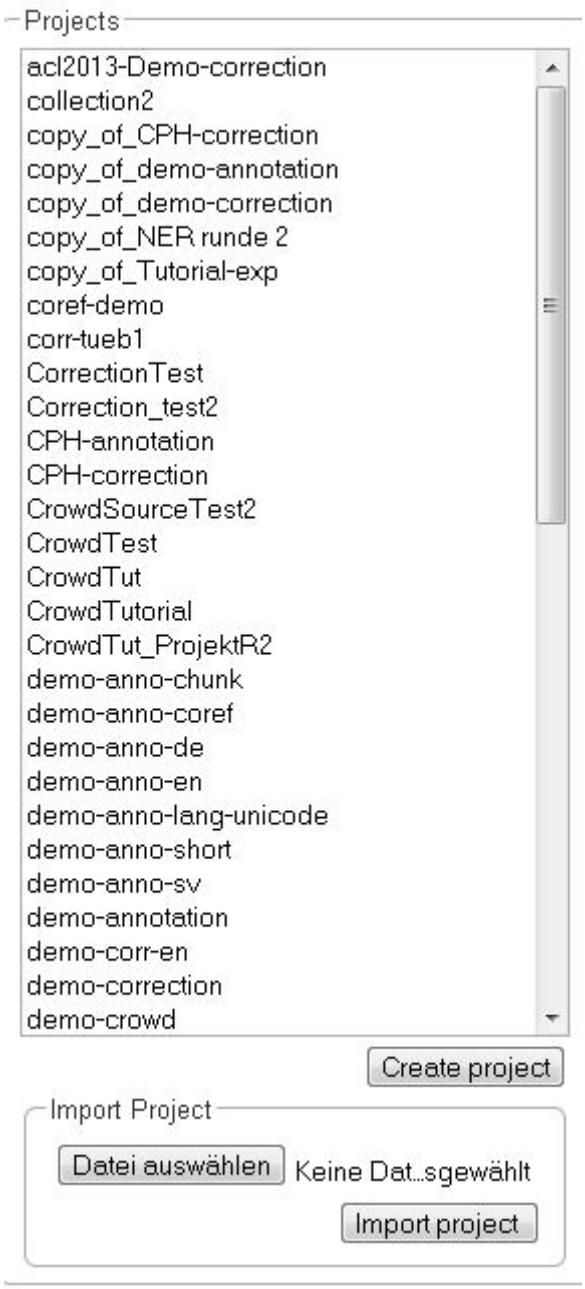
Projects

This functionality is only available to **project managers** (managers of existing projects), **project creators** (users with the ability to create new projects), and **administrators**. Project managers only see projects in which they hold the respective roles. Project creators only see projects in which they hold the project manager role.

This is the place to specify/edit annotation projects. You can either select one of the existing projects for editing, or click **Create Project** to add a project.

Although correction and automation projects function similarly, the management differs after the creation of the document. For further description, look at the corresponding chapters [sect_automation] and [sect_correction].

Only admins are allowed to create projects. Click on **Create Project** to create a new project.

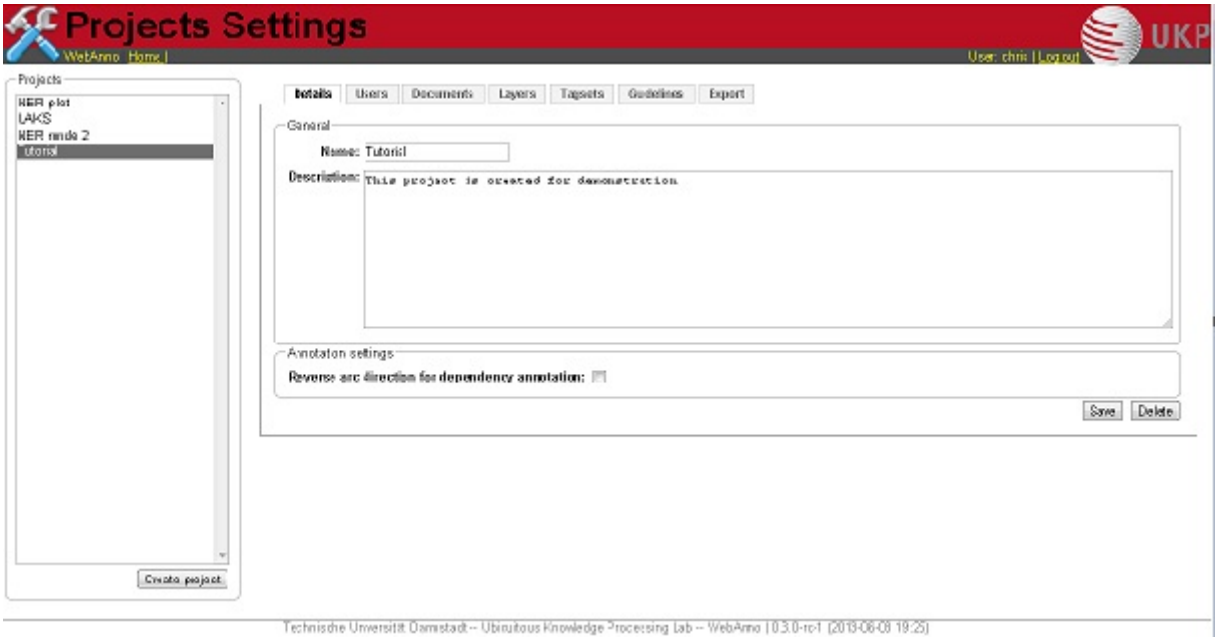


After doing so, a new pane is displayed, where you can name and describe your new project. It is also important to chose the kind of project you want to create. You have the choice between annotation, automation, and correction. Please do not forget to save.

After saving the details of the new project, it can be treated like any other already existing one. Also, a new pane with many options to organize the project is displayed.

To delete a project, click on it in the frame **Details**. The project details are displayed. Now, click on **Delete**.

The pane with the options to organize and edit a project, as described above, can also be reached by clicking on the desired project in the left frame.



By clicking on the tabs, you can now set up the chosen project.

Import

This functionality is only available to **administrators**.

Projects are associated with the accounts of users that act as project managers, annotators, or curators. When importing a previously exported project, you can choose to automatically **generate missing users** (enabled by default). If this option is disabled, projects still maintain their association to users by name. If the respective user accounts are created manually after the import, the users will start showing up in the projects.

Generated users are disabled and have no password. They must be explicitly enabled and a password must be set before the users can log in again.

Users

After clicking on **Users**, you are displayed a new pane in which you can add new users by clicking on the button **Add User**. After doing so, you get a list of users in the system which can be added to the project. By making a tick in front of the login, you can chose a new user.

Please do not forget to save after choosing all members of the project. Close the pane by clicking on **Cancel**. The rights of users created like this are that of an annotator. If you want to expand the user's status, you can do so by clicking on the user and then on **Change Permission**. The following frame will pop up.

After ticking the wished permissions, click on **Update**. To remove a user, click on the login and then **Remove User**.

Documents

To add or delete documents, you have to click on the tab **Documents** in the project pane. Two frames will be displayed. In the first frame you can import new documents.

Choose a document by clicking on **Choose Files**. Please mind the format, which you have to choose above. Then click on **Import Document**. The imported documents can be seen in the frame below. To delete a document from the project, you have to click on it and then click on **Delete** in the right lower corner.

Layers

All annotations belong to an annotation **layer**. Each layer has a structural **type** that defines if it is a **span**, a **relation**, or a **chain**. It also defines how the annotations behave and what kind of features it carries.

Creating a custom layer

This section provides a short walkthrough on the creation of a custom layer. The following sections act as reference documentation providing additional details on each step. In the following example, we will create a custom layer called **Sentiment** with a feature called **Polarity** that can be **negative**, **neutral**, or **positive**.

1. Create the layer *Sentiment*
    - Go to the **Layers** tab in your project's settings and press the **Create layer** button
    - Enter the name of the layer in **Layer name**: *Sentiment*
    - Choose the **type** of the layer: *Span*
    - Enable **Allow multiple tokens** because we want to mark sentiments on spans longer than a single token.
    - Press the **Save layer** button
2. Create the feature *Polarity*
    - Press the **New feature** button
    - Choose the **type** of the feature: *uima.cas.String*
    - Enter the **name** of the feature: *Polarity*
    - Press **Save feature**
3. Create the tagset *Polarity values*
    - Go to the **Tagsets** tab and press **Create tagset**
    - Enter the **name** of the tagset: *Polarity values*
    - Press **Save tagset**
    - Press **Create tag**, enter the **name** of the tag: *negative*, press **Save tag**
    - Repeat for *neutra* and *positive*
4. Assign the tagset *Polarity values* to the feature *Polarity*
    - Back in the **Layers** tab, select the layer: *Sentiment* and select the feature: *Polarity*
    - Set the **tagset** to *Polarity values*
    - Press **Save feature**

Now you have created your first custom layer.

Built-in layers

INCEpTION comes with a set of built-in layers that allow you to start annotating immediately. Also, many import/export formats only work with these layers as their semantics are known. For this reason, the ability to customize the behaviors of built-in layers is limited and it is not possible to extend them with custom features.

Table 11. Built-in layers

Layer	Type	Enforced behaviors
Chunk	Span	Lock to multiple tokens, no stacking, no sentence boundary crossing
Coreference	Chain	*(no enforced behaviors)*
Dependency	Relation over POS,	No stacking, no sentence boundary crossing
Lemma	Span	Locked to token offsets, no stacking, no sentence boundary crossing
Named Entity	Span	*(no enforced behaviors)*
Part of Speech (POS)	Span	Locked to token offsets, no stacking, no sentence boundary crossing

The coloring of the layers signal the following:

Table 12. Color legend

Color	Description
green	built-in annotation layer, enabled
blue	custom annotation layer, enabled
red	disabled annotation layer

To create a custom layer, select **Create Layer** in the **Layers** frame. Then, the following frame will be displayed.

Properties

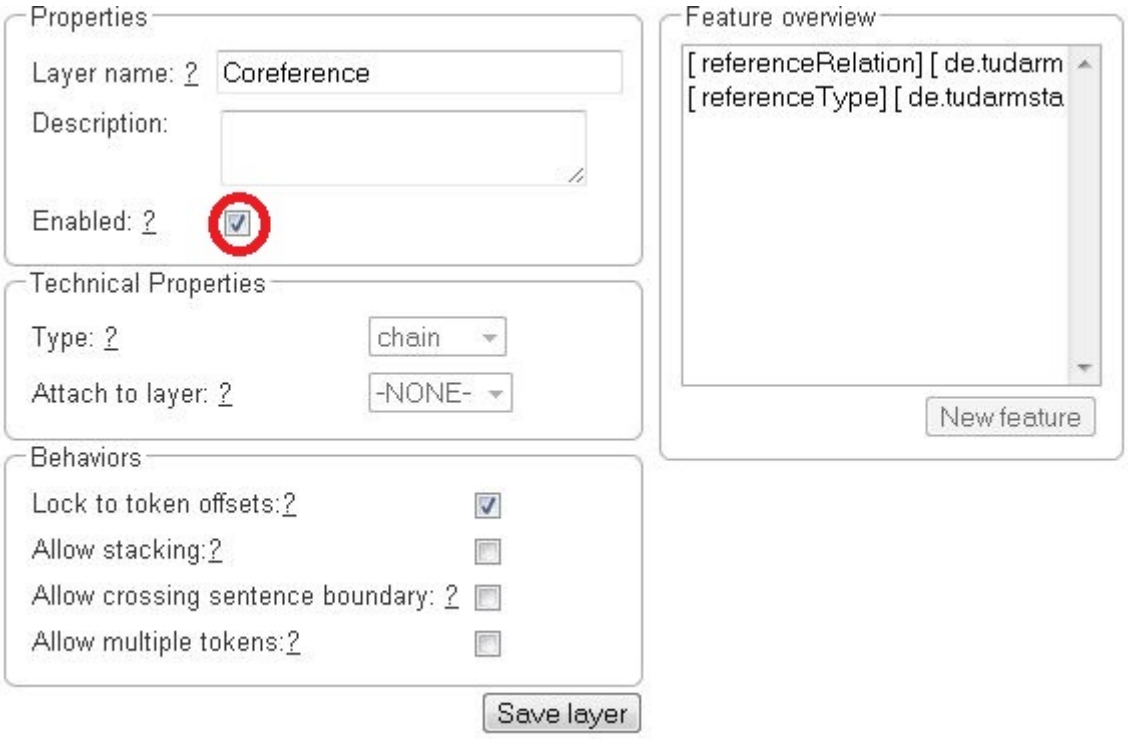
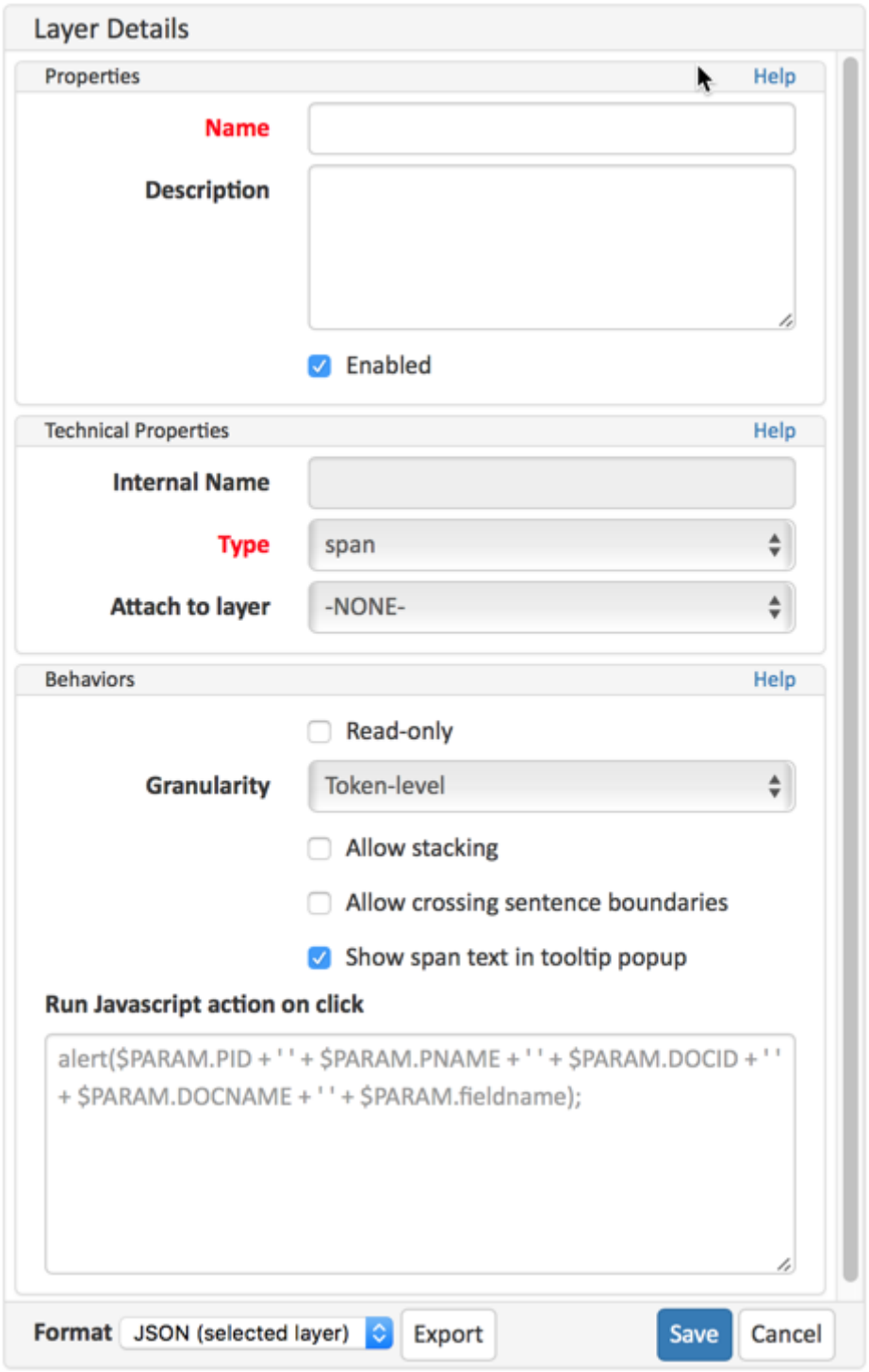


Table 13. Properites

Property	Description
Layer name	The name of the layer (obligatory)
Description	A description of the layer. This information will be shown in a tooltip when the mouse hovers over the layer name in the annotation detail editor panel.
Enabled	Whether the layer is enabled or not. Layers can currently not be deleted, but they can be disabled.

When a layer is first created, only ASCII characters are allowed for the layer name because the internal UIMA type name is derived from the initial layer name. After the layer has been created, the name can be changed arbitrarily. The internal UIMA type name will not be updated. The internal UIMA name is e.g. used when exporting data or in constraint rules.



Technical Properties

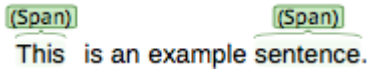
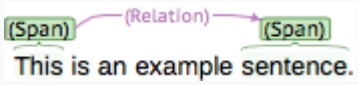
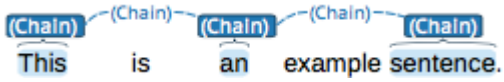
In the frame **Technical Properties**, the user may select the type of annation that will be made with this layer: span, relation, or chain.

Table 14. Technical Properites

Property	Description
Internal name	Internal UIMA type name
Type	The type of the layer (obligatory, see below)
Attach to layer *(Relations)*	Determines which span layer a relation attaches to. Relations can only be created between annotations of this span layer.

The layer type defines the structure of the layer. Three different types are supported: spans, relations, and chains.

Table 15. Layer types

Type	Description	Example
Span	Continous segment of text delimited by a start and end character offset. The example shows two spans.	
Relation	Binary relation between two spans visualized as an arc between spans. The example shows a relation between two spans.	
Chain	Directed sequence of connected spans in which each span connects to the following one. The example shows a single chain consisting of three connected spans.	

For relation annotations the type of the spans which are to be connected can be chosen in the field **Attach to layer**. Here, only non-default layers are displayed. To create a relation, first the span annotation needs to be created.

Currently for each span layer there can be at most one relation layer attaching to it.

It is currently not possible to create relations between spans in different layers. For example if you define span layers called **Men** and **Women**, it is impossible to define a relation layer **Married to** between the two. To work around this limitation, create a single span layer **Person** with a feature **Gender** instead. You can now set the feature **Gender** to **Man** or **Woman** and eventually define a relation layer **Married to** attaching to the **Person** layer.

Behaviours

Table 16. Behaviors

Behavior	Description
Read-only	The layer may be viewed but not edited.

Behavior	Description
Granularity *(span, chain)*	The granularity controls at which level annotations can be created. When set to **Character-level**, annotations can be created anywhere. Zero-width annotations are permitted. When set to **Token-level** or **Sentence-level** annotation boundaries are forced to coincide with token/sentence boundaries. If the selection is smaller, the annotation is expanded to the next larger token/sentence covering the selection. Again, zero-width annotations are permitted. When set to **Single tokens only** may be applied only to a single token. If the selection covers multiple tokens, the annotation is reduced to the first covered token at a time. Zero-width annotations are not permitted in this mode. Note that in order for the **Sentence-level** mode to allow annotating multiple sentences, the **Allow crossing sentence boundary** setting must be enabled, otherwise only individual sentences can be annotated.
Allow stacking	Allow multiple annotations in this layer to be made at exactly the same position. If this option is disabled, a new annotation made at the same location as an existing annotation will replace the existing annotation.
Allow crossing sentence boundary *(chain)*	Allow annotations to cross sentence boundaries.
Behave like a linked list	Controls what happens when two chains are connected with each other. If this option is **disabled**, then the two entire chains will be merged into one large chain. Links between spans will be changed so that each span connects to the closest following span - no arc labels are displayed. If this option is **enabled**, then the chains will be split if necessary at the source and target points, reconnecting the spans such that exactly the newly created connection is made - arc labels are available.

Features

In this section, features and their properties can be configured.

When a feature is first created, only ASCII characters are allowed for the feature name because the internal UIMA name is derived from the initial layer name. After the feature has been created, the name can be changed arbitrarily. The internal UIMA feature name will not be updated. The internal UIMA name is e.g. used when exporting data or in constraint rules.

Table 17. Feature properties

Property	Description
Internal name	Internal UIMA feature name
Type	The type of the feature (obligatory, see below)
Name	The name of the feature (obligatory)
Description	A description that is shown when the mouse hovers over the feature name in the annotation detail editor panel.
Enabled	Features cannot be deleted, but they can be disabled
Show	Whether the feature value is shown in the annotation label. If this is disabled, the feature is only visible in the annotation detail editor panel.
Remember	Whether the annotation detail editor should carry values of this feature over when creating a new annotation of the same type. This can be useful when creating many annotations of the same type in a row.
Tagset *(String)*	The tagset controlling the possible values for a string feature.

The following feature types are supported.

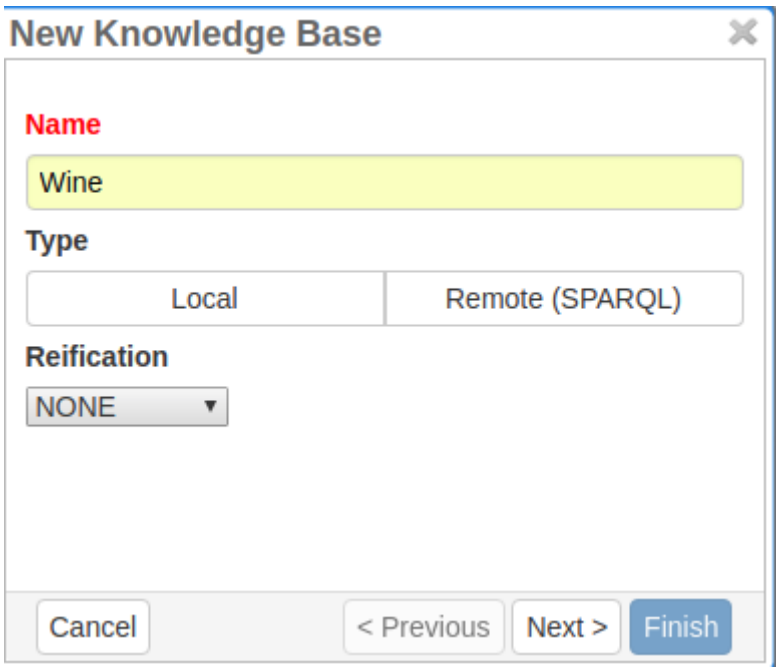
Table 18. Feature types

Type	Description
uima.cas.String	Textual feature that can optionally be controlled by a tagset. It is rendered as a text field or as a combobox if a tagset is defined.
uima.cas.Boolean	Boolean feature that can be true or false and is rendered as a checkbox.
uima.cas.Integer	Numeric feature for integer numbers.
uima.cas.Float	Numeric feature for decimal numbers.
uima.tcas.Annotation *(Span layers)*	Link feature that can point to any arbitrary span annotation
other span layers *(Span layers)*	Link feature that can point only to the selected span layer.

Please take care that when working with non-custom layers, they have to be ex- and imported, if you want to use the resulting files in e.g. correction projects.

Knowledge Bases

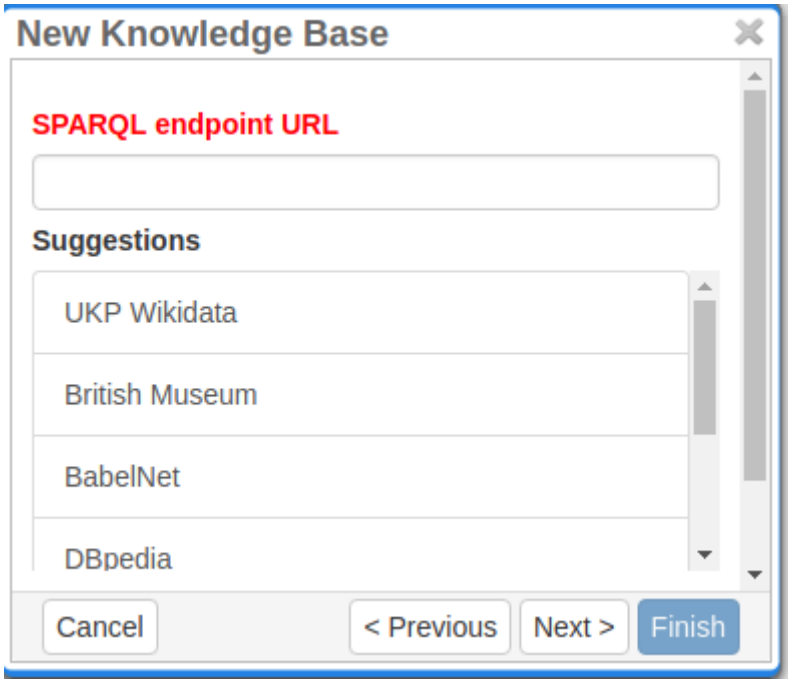
In the **Projects Settings**, switch to the **Knowledge Bases** tab, then click **New...** on the bottom and a dialogue box shows as in the figure below.



To create a **local** or **remote** knowledge base, one needs to choose **Local** or **Remote** for the type. For the reification, **NONE** is the default case, but to support qualifiers, one needs to choose **WIKIDATA**.

For the local KB, the user can optionally choose a RDF file from which to import the initial data. Alternatively, the user can skip the step to create an empty KB to create a knowledge base from scratch. It is also always possible to import data from an RDF file after the creation of a KB. It is also possible to multiple RDF files into the same KB, one after another.

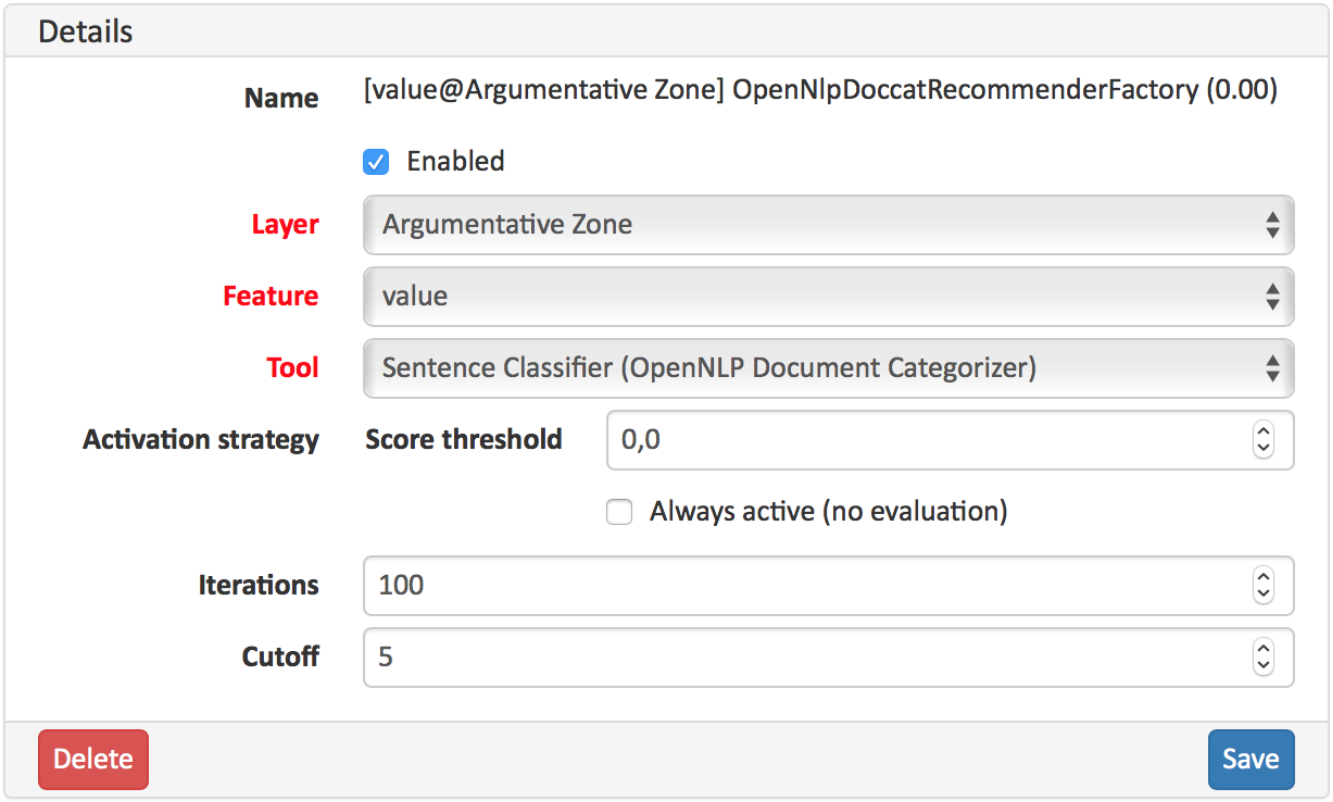
For remote KBs, INCEpTION provides the user with some pre-configured knowledge base such as WikiData, British Museum, BabelNet, DBPediaa or Yago. The user can also set up a custom remote KB, in which case the user needs to provide the SPARQL endpoint URL for the knowledge base as in the figure below.



Schema mapping

Different types of knowledge base schemata are supported via a configurable mapping mechanism. The user can choose one of the pre-configured mapping or provide her own custom defined mapping as shown in the screenshot below.

In the advanced settings, the user can leverage this feature of KB settings when one doesn't want the entire knowledge base to be used and rather choose to identify some specific root concepts. This feature specially helps in case of large knowledge bases such as Wikidata.



Full text search

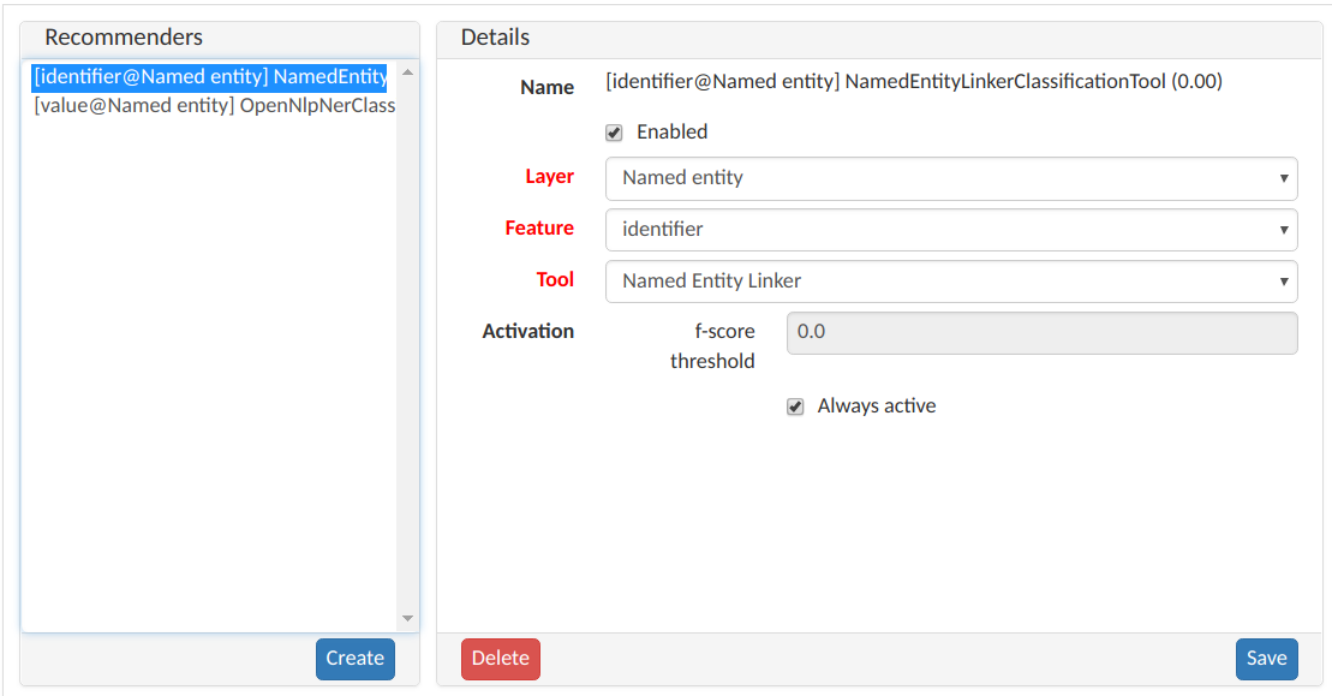
Full text search in knowledge bases enables searching for entities by their textual context, e.g. their label. This is a prerequisite for some advanced features such as re-ranking linking candidates during entity linking. Two full text search modes are supported:

- the `bif:contains` mode can be used with some remote triple stores such as Virtuoso.
- the `lucenesail#matches` mode can be used with local knowledge bases or possible with remote knowledge bases using the RDF4J Lucene SAIL.

Recommenders

Recommenders provide annotation support by predicting potential labels. These can be either accepted or rejected by the user. The recommenders learn from this interaction to further improve the quality of predictions.

After clicking on **Recommenders**, you are displayed a new pane in which you can add new recommenders by clicking on the button **Create**. You have to select the layer, feature and the classification tool. The recommenders are trained every time you create, update or delete an annotation; and evaluated every second time. During recommender evaluation the f-score of each recommender is calcultated, and recommenders with a lower f-score than their threshold will not be selected for the next training step. By making a tick on the checkbox next to **Always active** or leaving the threshold at 0, you can choose to skip f-score evaluation to ensure that the recommender runs at all times.



Please do not forget to save after configuring a recommender. Close the pane by clicking on **Cancel**. If you want to edit a recommender, you can do so by clicking on the recommender and save after editing.

To remove a recommender, click on the recommender and then on **Delete**. This will also remove all predictions by this recommender.

String Matcher

This recommender projects the annotations already made to tokens with the same text.

Sentence Classifier (OpenNLP Document Categorizer)

This recommender is available for sentence-level annotation layers where cross-sentence annotations are disabled. It learns labels using a sentence-level bag-of-word model using the OpenNLP Document Categorizer.

Token Sequence Classifier (OpenNLP POS)

This recommender uses the OpenNLP Part-of-Speech Tagger to learn a token-level sequence tagging model for layers that anchor to single tokens. The model will attempt to assign a label to every single token. The model considers all sentences for training in which at least a one annotation with a feature value exists.

Multi-Token Sequence Classifier (OpenNLP NER)

This recommender uses the OpenNLP Name Finder to learn a sequence tagging model for multi-token annotations. The model generates a BIO-encoded representation of the annotations in the sentence.

If a layer contains overlapping annotations, the recommender does not fail, but it is also not able to learn properly.

Named Entity Linker

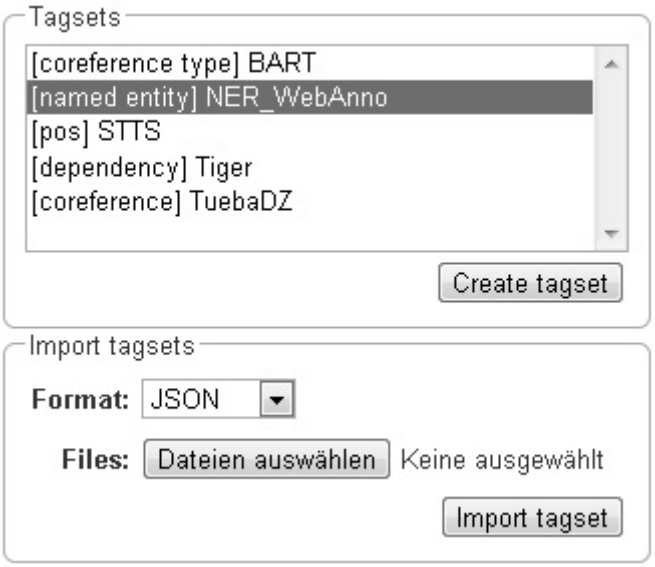
This recommender can be used with concept features on span layers. It does not learn from training data, but instead attempts to match the context of the entity mention in the text with the context of candidate entities in the knowledge base and suggests the highest ranked candidate entities. In order for this recommender to function, it is necessary that the knowledge base configured for the respective concept feature supports full text search.

External Recommender

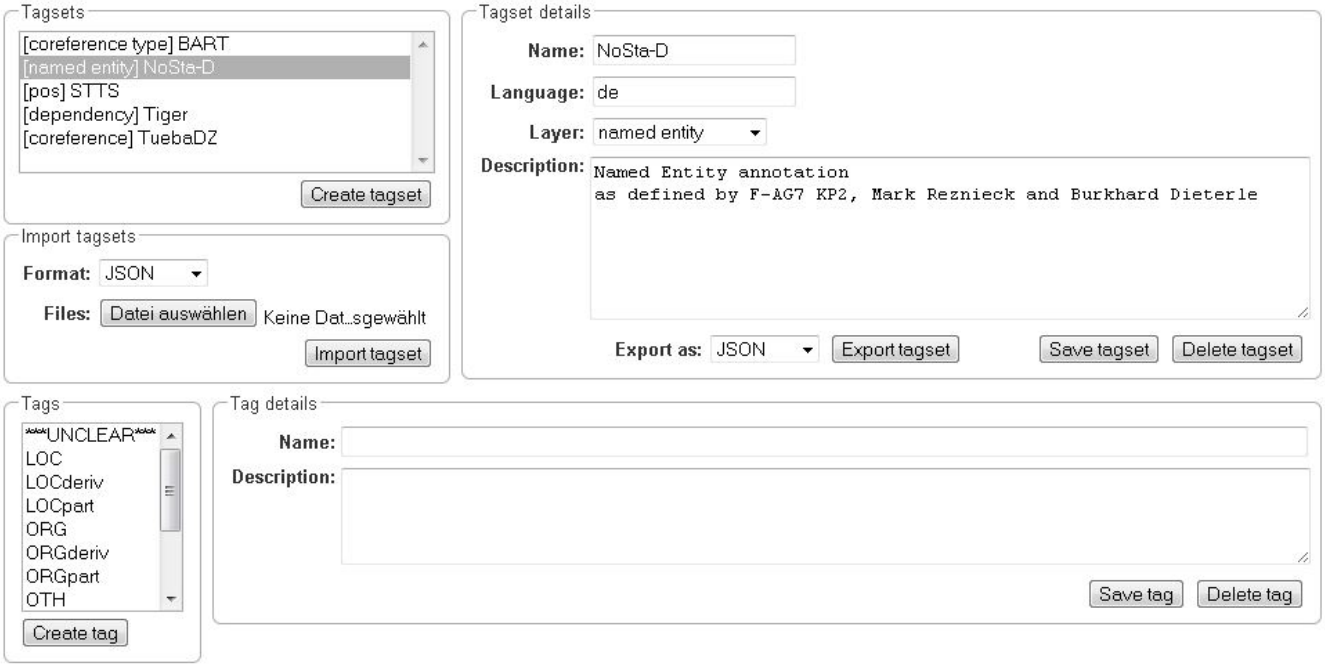
This recommender allows to use external web-services to generate predictions. For details on the protocol used in the communication with the external services, please refer to the developer documentation.

Tagsets

To administer the tagsets, click on the tab **Tagsets** in the project pane.

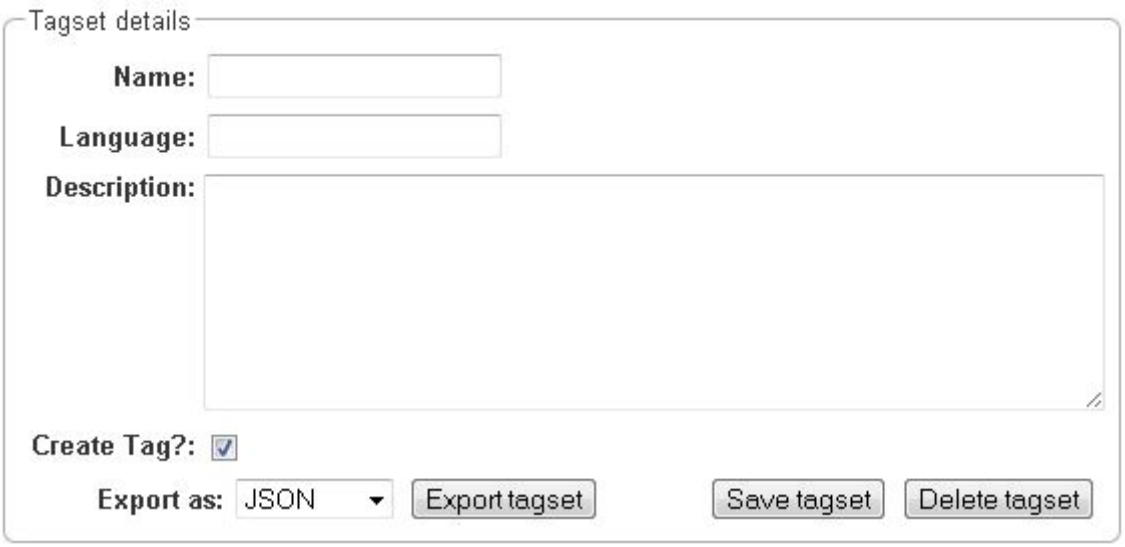


To administer one of the existing tagsets, select it by a click. Then, the tagset characteristics are displayed.



In the Frame **Tagset details**, you can change them, export a tagset, save the changes you made on it or delete it by clicking on **Delete tagset**. To change an individual tag, you select one in the list displayed in the frame **Tags**. You can then change its description or name or delete it by clicking **Delete tag** in **Tag details**. Please do not forget to save your changes by clicking on **Save tag**. To add a new tag, you have to click on **Create tag** in **Tag details**. Then you add the name and the description, which is optional. Again, do not forget to click **Save tag** or the new tag will not be created.

To create an own tagset, click on **Create tagset** and fill in the fields that will be displayed in the new frame. Only the first field is obligatory. Adding new tags works the same way as described for already existing tagsets. If you want to have a free annotation, as it could be used for lemma or meta information annotation, do not add any tags.



To export a tagset, choose the format of the export at the bottom of the frame and click **Export tagset**.

Guidelines

To add or delete guidelines, which will be accessible by users in the project, you have to select the tab **Guidelines**. Two new frames will be displayed. To upload guidelines, click on **Choose files** in the first frame – **Add guideline document**, select a file from your local disc and then click **Import guidelines**.

Uploaded guidelines are displayed in the second frame – **Guideline documents**. To delete a guideline document, click on it and then on **Delete** in the right lower corner of the frame.

Constraints

To import a constraints file, go to **Project** and click on the particular project name. On the left side of the screen, a tab bar opens. Choose **Constraints**. You can now choose a constraint file by clicking on **Choose Files**. Then, click on **Import**. Upon import, the application checks if the constraints file is well formed. If they conform to the rules of writing constraints, the constraints are applied.

Export

Two modes of exporting projects are supported:

- **Export the whole project** for the purpose of creating a backup, of migrating it to a new INCEpTION version, of migrating to a different INCEpTION instance, or simply in order to re-import it as a duplicate copy.
- **Export curated documents** for the purpose of getting an easy access to the final annotation results. If you do not have any curated documents in your project, this export option is not offered.

The format of the exported annotations is selected using the **Format** drop-down field. When **AUTO** is selected, the file format corresponds to the format of the source document. If there is no write support for the source format, the file is exported in the WebAnno TSV3 format instead.

Some browsers automatically extract ZIP files into a folder after the download. Zipping this folder and trying to re-import it into the application will generally not work because the process introduces an additional folder level within the archive. The best option is to disable the automatic extraction in your browser. E.g. in Safari, go to **Preferences** → **General** and disable the setting **Open "safe" files after downloading**.

When exporting a whole project, the structure of the exported ZIP file is as follows:

- **<project ID>.json** - project metadata file
- **annotation**
  - **<source document name>**
    - **<user ID>.XXX** - file representing the annotations for this user in the selected format.
- **annotation_ser**
  - **<source document name>**
    - **<user ID>.ser** - serialized CAS file representing the annotations for this user
- **curation**
  - **<source document name>**
    - **CURATION_USER.XXX** - file representing the state of curation in the selected format.
    - **CORRECTION_USER.XXX** - *correction* project: original document state, *automation* project automatically generated suggestions
- **curation_ser**
  - **<source document name>**
    - **CURATION_USER.ser** - serialized UIMA CAS representing the state of curation
    - **CORRECTION_USER.ser** - *correction* project: original document state, *automation* project automatically generated suggestions
- **log**
  - **<project ID>.log** - project log file
- **source** - folder containing the original source files

The files under `annotation` and `curation` are provided for convenience only. They are ignored upon import.

Currently, it is not possible to choose a specific format for bulk-exporting annotations. However, this mailing list post describes how DKPro Core can be used to transform the UIMA CAS formats into alternative formats.

Constraints

Constraints reorder the choice of tags based on the context of an annotation. For instance, for a given lemma, not all possible part-of-speech tags are sensible. Constraint rules can be set up to reorder the choice of part-of-speech tags such that the relevant tags are listed first. This speeds up the annotation process as the annotator can choose from the relevant tags more conveniently.

The choice of tags is not limited, only the order in which they are presented to the annotator. Thus, if the project manager has forgotten to set up a constraint or did possible not consider an oddball case, the annotator can still make a decision.

A **constraint set** consists of two components:

- import statement
- scopes
- Import statements* are composed in the following way:

```
import <fully_qualified_name_of_layer> as <shortName>;
```

It is necessary to declare short names for all fully qualified names because only short names can be used when writing a constraint rule. Short names cannot contain any dots or special characters, only letters, numbers, and the underscore.

If you are not sure what the fully qualified name of a layer is, you can look it up going to **Layers** in **Project settings**. Click on a particular layer and you can view the fully qualified name under **Technical Properties**.

Scopes consist of a **scope name** and one or more **rules** that refer to a particular annotation layer and define restrictions for particular conditions. For example, it is possible to reorder the applicable tags for a POS layer, based on what kind of word the annotator is focusing on.

While scope names can be freely chosen, scope rules have a fixed structure. They consist of **conditions** and **restrictions**, separated by an arrow symbol (→). Conditions consist of a **path** and a **value**, separated by an equal sign (=). Values always have to be embraced by double-quotes. Multiple conditions in the same rule are connected via the &-operator, multiple restrictions in the same rule are connected via the |-operator.

Typically a rule's syntax is

Single constraint rule

```
<scopeName> {
  <condition_set> -> <restriction_set>;
}
```

This leads to the following structure:

Multiple constraint rules

```
<scopeName> {
  <rule_1>;
  ...
  <rule_n>;
}
```

Both conditions and restrictions are composed of a **path** and a **value**. The latter is always enclosed in double quotes.

Structure of conditions and restrictions

```
<path>="<value>"
```

A **condition** is a way of defining whether a particular situation in INCEpTION is based on annotation layers and features in it.

A **condition set** consists of one or more conditions. They are connected with logical AND as follows.

```
<condition> & <condition>
```

A **restriction set** defines a set of restrictions which can be applied if a particular condition set is evaluated to true. As multiple restrictions inside one rule are interpreted as conjunctions, they are separated by the `|`-operator".

```
<restriction> | <restriction>
```

A **path** is composed of one or more steps, separated by a dot. A **step** consists of a **feature selector** and a **type selector**. **Type selectors** are only applicable while writing the condition part of a rule. They comprise a **layer operator** `@` followed by the type (Lemma, POS, etc). **Feature selectors** consist of a feature name, e.g.

```
pos.PosValue
```

Navigation across layers is possible via

```
@<shortLayerName>
```

Hereby all annotations of type `<shortLayerName>` at the same position as the current context are found.

Comments

The constraint language supports block comments which start with `/` and end with `/`. These comments may span across multiple lines.

```
/* This is a single line comment */

/*
   This is a multi-
   line comment
*/
```

Conditional features

Constraints can be used to set up conditional features, that is features that only become available in the UI if another feature has a specific value. Let's say that for example you want to annotate events and only **causing** events should additionally offer a **polarity** feature, while for **caused** events, there should be no way to select a polarity.

Sticking with the example of annotating events, conditional features can be set up as following:

- Go to the **Layer** tab of the project settings
- Create a new tagset called **Event category** and add the tags **causing** and **caused**
- Create a new tagset called **Event polarity** and add the tags **positive** and **negative**
- Create a new span layer called **Event**
- Add a string feature called **category** and assign the tagset **Event category**
- Save the changes to the **category** feature
- Add a string feature called **polarity** and assign the tagset **Event polarity**
- Enabled the checkbox **Hide Un-constraint feature** on the **polarity** feature
- Save the changes to the **polarity** feature
- Create a new text file called `constraints.txt` with the following contents .

```
import webanno.custom.Event as Event;

Event {
  category="causing" -> polarity="positive" | polarity="negative";
}
```

- Import `constraints.txt` in the tab **Constraints** in the project settings.

When you now annotate an **Event** in this project, then the **polarity** feature is only visible and editable if the **category** of the annotation is set to **causing**.

It is important that both of the features have tagsets assigned - otherwise the conditional effect will not take place.

Constraints for slot features

Constraints can be applied to the roles of slot features. This is useful, e.g. when annotating predicate/argument structures where specific predicates can only have certain arguments.

Consider having a span layer `SemPred` resembling a semantic predicate and bearing a slot feature `arguments` and a string feature `senseId`. We want to restrict the possible argument roles based on the lemma associated with the predicate. The first rule in the following example restricts the `senseId` depending on the value of a `Lemma` annotation at the same position as the `SemPred` annotation. The second rule then restricts the choice of roles for the arguments based on the `senseId`. Note that to apply a restriction to the role of a slot feature, it is necessary to append `.role` to the feature name (that is because `role` is technically a nested feature). Thus, while we can write e.g. `senseId = "Request"` for a simple string feature, it is necessary to write `arguments.role = "Addressee"`.

Note that some role labels are marked with the flag `(!)`. This is a special flag for slot features and indicates that slots with these role labels should be automatically displayed in the UI ready to be filled. This should be used for mandatory or common slots and saves time as the annotator does not have to manually create the slots before filling them.

```
SemPred {
  /* Rule 1 */
  @Lemma.value = "ask" -> senseId = "Questioning" | senseId = "Request" | senseId =
"XXX";
  /* .. other lemmata */
  /* Rule 2 */
  senseId = "Questioning" ->
    /* core roles */
    arguments.role = "Addressee" (!) | arguments.role = "Message" (!) | arguments.role
= "Speaker" (!) |
    /* non-core roles */
    arguments.role = "Time" | arguments.role = "Iterations";
  /* .. other senses */
}
```

Constraints language grammar

Constraints language grammar

```
// Basic structure --------------------------------------
<file>            ::= <import>* | <scope>*
<scope>           ::= <shortLayerName> "{" <ruleset> "}"
<ruleset>         ::= <rule>*
<import>          ::= "import" <qualifiedLayerName>
                      "as" <shortLayerName>
<rule>            ::= <conds> "->" <restrictions> ";"

// Conditions -------------------------------------------
<conds>           ::= <cond> | <cond> "&" <conds>
<cond>            ::= <path> "=" <value>
<path>            ::= <featureName> | <step> "." <path>
<step>            ::= <featureName> | <layerSelector>
<layerSelector>   ::= <layerOperator>? <shortLayerName>
<layerOperator>   ::= "@" // select annotation in layer X

// Restrictions -----------------------------------------
<restrictions>    ::= <restriction> |
                      <restriction> "|" <restrictions>
<restriction>     ::= <restrictionPath> "=" <value>
                      ( "(" <flags> ")" )
<restrictionPath> ::= <featureName> |
                      <restrictionPath> "." <featureName>
<flags>           ::= "!" // core role
```

User Management

This functionality is only available to **administrators**.

After selecting this functionality, a frame which shows all users is displayed. By selecting a user, a frame is displayed on the right.

Now you may change his role or password, specify an e-mail address and dis- or enable his account by placing the tick.

Disabling an account prevents the user from logging in. The user remains associated with any projects and remains visible in on the Monitoring page.

To create a new user, click on **Create** in the left frame. This will display a similar frame as the one described in the last paragraph. Here you have to give a login-name to the new user.

In both cases, do not forget to save your changes by pressing the **Save** button.

1. User roles

Role	Description
ROLE_USER	**User.** Required to log in to the application. Removal of this role from an account will prevent login even for users that additionally hold the ROLE_ADMIN!
ROLE_ADMIN	**Administrator.** Can manage users and has access to all other functionalities.
ROLE_PROJECT_CREATOR	**Project creator.** Can create new projects.
ROLE_REMOTE	**Remote API access.** Currently experimental and undocumented. Do not use.

Formats

Table 19. 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

WebAnno TSV 3.2 File format

In this section, we will discuss the WebAnno TSV (Tab Separated Value) file format version 3.2. The format is similar to the CoNNL file formats with specialized additions to the header and column representations. The file format inhabits a header and a body section. The **header** section present information about the different types of annotation layers and features used in the file. While importing the WebAnno TSV file, the specified headers should be first created in to the running WebAnno project. Otherwise, the importing of the file will not be possible.

The **body** section of the TSV file presents the document and all the associated annotations including sentence and token annotations.

Encoding and Offsets

TSV files are always encoded in UTF-8. However, the offsets used in the TSV file are based on UTF-16. This is important when using TSV files with texts containing e.g. Emojis or some modern non-latin Asian, Middle-eastern and African scripts.

WebAnno is implemented in Java. The Java platform internally uses a UTF-16 representation for text. For this reason, the offsets used in the TSV format currently represent offsets of the 16bit units in UTF-16 strings. This is important if your text contains Unicode characters that cannot be represented in 16bit and which thus require two 16bit units. For example a token represented by the Unicode character 🙌 (U+1F60A) requires two 16bit units. Hence, the offset count increased by 2 for this character. So Unicode characters starting at U+10000 increase the offset count by 2.

Example: TSV sentence containing a Unicode character from the Supplementary Planes

```
#Text=I like it 🙌 .
1-1 0-1 I _
1-2 2-6 like _
1-3 7-9 it _
1-4 10-12 🙌 *
1-5 13-14 . _
```

Since the character offsets are based on UTF-16 and the TSV file itself is encoded in UTF-8, first the text contained in the file needs to be transcoded from UTF-8 into UTF-16 before the offsets can be applied. The offsets cannot be used for random access to characters directly in the TSV file.

File Header

WebAnno TSV 3.2 file starts with the following header marker

Example: format in file header

```
#FORMAT=WebAnno TSV 3.2
```

Layers are marked by the `#` character followed by `T_SP=` for **span types** (including **slot features**), `T_CH=` for **chain layers**, and `T_RL=` for **relation layers**. Every layer is written in new line, followed by the features in the layer. If all layer type exists, first, all the span layers will be written, then the chain layer, and finally the relation layers. Features are separated by the `|` character and only the short name of the feature is provided.

Example: Span layer with simple features in file header

```
#T_SP=webanno.custom.Pred|bestSense|lemmaMapped|senseId|senseMapped
```

Here the layer name is **webanno.custom.Pred** and the features are named **bestSense**, **lemmaMapped**, **senseId**, **senseMapped**. Slot features start with a prefix `ROLE_` followed by the name of the role and the link. The role feature name and the link feature name are separated by the `_` character.

The target of the slot feature always follows the role/link name

Example: Span layer with slot features in file header

```
#T_SP=webanno.custom.SemPred|ROLE_webanno.custom.SemPred:RoleSet_webanno.custom.SemPre
dRoleSetLink|uima.tcas.Annotation|aFrame
```

Here the name of the role is **webanno.custom.SemPred:RoleSet** and the name of the role link is **webanno.custom.SemPredRoleSetLink** and the target type is **uima.tcas.Annotation**.

Chain layers will have always two features, **referenceType** and **referenceRelation**.

Example: Chain layers in file header

```
#T_CH=de.tudarmstadt.ukp.dkpro.core.api.coref.type.CoreferenceLink|referenceType|refer
enceRelation
```

Relation layers will come at last in the list and the very last entry in the features will be the type of the base (governor or dependent) annotations with a prefix `BT_`.

Example: Relation layers in file header

```
#T_RL=de.tudarmstadt.ukp.dkpro.core.api.syntax.type.dependency.Dependency|DependencyTy
pe|BT_de.tudarmstadt.ukp.dkpro.core.api.lexmorph.type.pos.POS
```

Here, the relation type `de.tudarmstadt.ukp.dkpro.core.api.syntax.type.dependency.Dependency` has a feature `DependencyType` and the relation is between a base type of `de.tudarmstadt.ukp.dkpro.core.api.lexmorph.type.pos.POS`.

File Body / Annotations

In this section we discuss the different representations of texts and annotation in WebAnno TSV3format

Reserved Characters

Reserved characters have a special meaning in the TSV format and must be are escaped with the **backslash** (\) character if they appear in text or feature values. Reserved characters are the following:

Reserved Characters

```
\,[,],|,_,->,;,\t,\n,*
```

The way that TSV is presently defined/implemented, it kind of considers → as a single "character"... and it is also escaped as a single unit, i.e. → becomes ->. It is something to be addressed in a future iteration of the format.

Sentence Representation

Sentence annotations are presented following the text marker #Text=, before the token annotations. All text given here is inside the sentence boundaries.

Example: Original text sections

```
#Text=Bell , based in Los Angeles , makes and distributes electronic , computer and
building products .
```

The text of an imported document is reconstructed from the sentence annotations. Additionally, the offset information of the sentence tokens are taken into account to determine whether padding needs to be added between sentences. The TSV format can presently not record text that occurs in between two sentences.

If a sentence spans multiple lines, the text is split at the line feed characters (ASCII 12) and multiple #Text= lines are generated. Note that carriage return characters (ASCII 13) are kept as escaped characters (\r).

Example: Original multi-line text

```
#Text=Bell , based in Los Angeles , makes and distributes
#Text=electronic , computer and building products .
```

Token and Sub-token Annotations

Tokens represent a span of text within a sentence. Tokens cannot overlap, although then can be directly adjacent (i.e. without any whitespace between them). The start offset of the first character of the first token corresponds to the start of offset of the sentence.

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: Token position

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

Here `1` indicates the sentence number, `2` indicates the token number (here, the second token in the first sentence) and `4` is the begin offset of the token and `8` is the end offset of the token while `Haag` is the token.

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

Example: Sub-token positions

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

Here, the sub-token `play` is indicated by sentence-token number `1-3.1` and the sub-token `s` is indicated by `1-3.2`.

While tokens may not overlap, sub-tokens may overlap.

Example: Overlapping sub-tokens

```
1-3 9-14  plays
1-3.1 9-12  pla
1-3.2 11-14 ays
```

Span Annotations

For every features of a span Annotation, annotation value will be presented in the same row as the token/sub-token annotation, separated by a TAB character. If there is no annotation for the given span layer, a `_` character is placed in the column. If the feature has no/null annotation or if the span layer do not have a feature at all, a `*` character represents the annotation.

Example: Span layer declaration in file header

```
#T_SP=de.tudarmstadt.ukp.dkpro.core.api.lexmorph.type.pos.POS|PosValue
#T_SP=webanno.custom.Sentiment|Category|Opinion
```

Example: Span annotations in file body

```
1-9 36-43   unhappy JJ  abstract    negative
```

Here, the first annotation at column 4, `JJ` is avalue for a feature **PosValue** of the layer **de.tudarmstadt.ukp.dkpro.core.api.lexmorph.type.pos.POS**. For the two features of the layer **webanno.custom.Sentiment** (**Category** and **Opinion**), the values `abstract` and `negative` are presented at column 5 and 6 resp.

Disambiguation IDs

Within a single line, an annotation can be uniquely identified by its type and stacking index. However, across lines, annotation cannot be uniquely identified easily. Also, if the exact type of the referenced annotation is not known, an annotation cannot be uniquely identified. For this reason, disambiguation IDs are introduced in potentially problematic cases:

- stacked annotations - if multiple annotations of the same type appear in the same line
- multi-unit annotations - if an annotations spans multiple tokens or sub-tokens
- un-typed slots - if a slot feature has the type `uima.tcas.Annotation` and may thus refer to any kind of target annotation.

The disambiguation ID is attached as a suffix `[N]` to the annotation value. Stacked annotations are separated by `|` character.

Example: Span layer declaration in file header

```
#T_SP=de.tudarmstadt.ukp.dkpro.core.api.lexmorph.type.pos.POS|PosValue
#T_SP=de.tudarmstadt.ukp.dkpro.core.api.ner.type.NamedEntity|value
```

Example: Multi-token span annotations and stacked span annotations

```
1-1 0-3 Ms. NNP PER[1]|PERpart[2]
1-2 4-8 Haag    NNP PER[1]
```

Here, `PER[1]` indicates that token `1-1` and `1-2` have the same annotation (multi-token annotations) while `PERpart[2]` is the second (stacked) annotation on token `1-1` separated by `|` character.

On chain layers, the number in brackets is **not** a disambiguation ID but rather a chain ID!

Slot features

Slot features and the target annotations are separated by TAB character (first the feature column then the target column follows). In the target column, the `sentence-token` id is recorded where the feature is drawn.

Unlike other span layer features (which are separated by `|` character), multiple annotations for a slot feature are separated by the `;` character.

Example: Span layer declaration in file header

```
#T_SP=webanno.custom.Frame|FE|ROLE_webanno.custom.Frame:Roles_webanno.custom.FrameRole
sLink|webanno.custom.Lu
#T_SP=webanno.custom.Lu|luvalue
```

Example: Span annotations and slot features

```
2-1 27-30   Bob _   _   _   bob
2-2 31-40   auctioned   transaction seller;goods;buyer  2-1;2-3[4];2-6
2-3 41-44   the _   _   _   clock[4]
2-4 45-50   clock   _   _   _   clock[4]
2-5 52-54   to  _   _   _   _
2-6 55-59   John    _   _   _   john
2-7 59-60   .   _   _   _   _
```

Here, for example, at token `2-2`, we have three slot annotations for feature `Roles` that are `seller`, `goods`, and `buyer`. The targets are on token `2-1 `, `2-3[4]`, and `2-6` respectively which are on annotations of the layer `webanno.custom.Lu` which are `bob`, `clock` and `john`.

Chain Annotations

In the Chain annotation, two columns (TAB separated) are used to represent the `referenceType` and the `referenceRelation`. A chain ID is attached to the `referenceType` to distinguish to which of the chains the annotation belongs. The `referenceRelation` of the chain is represented by the relation value followed by `→` and followed by the `CH-LINK` number where `CH` is the chain number and `LINK` is the link number (the order the chain).

Example: Chain layer declaration in file header

```
#T_CH=de.tudarmstadt.ukp.dkpro.core.api.coref.type.CoreferenceLink|referenceType|refer
enceRelation
```

Example: Chain annotations

```
1-1 0-2 He  pr[1]   coref->1-1
1-2 3-7 shot    _   _
1-3 8-15    himself pr[1]   coref->1-2
1-4 16-20   with    _   _
1-5 21-24   his pr[1]   *->1-3
1-6 25-33   revolver    _   _
1-7 33-34   .   _   _
```

In this example, token `1-3` is marked as `pr[1]` which indicates that the **referenceType** is `pr` and it is part of the chain with the ID `1`. The relation label is `coref` and with the `CH-LINK` number `1-2` which means that it belongs to chain `1` and this is the second link in the chain.

Relation Annotations

Relation annotations comes to the last columns of the TSV file format. Just like the span annotations, every feature of the relation layers are represented in a separate TAB. Besides, one extra column (after all feature values) is used to write the token id from which token/sub-token this arc of a relation annotation is drawn.

Example: Span and relation layer declaration in file header

```
#T_SP=de.tudarmstadt.ukp.dkpro.core.api.lexmorph.type.pos.POS|PosValue
#T_RL=de.tudarmstadt.ukp.dkpro.core.api.syntax.type.dependency.Dependency|DependencyTy
pe|BT_de.tudarmstadt.ukp.dkpro.core.api.lexmorph.type.pos.POS
```

Example: Span and relation annotations

```
1-1 0-3 Ms. NNP SUBJ    1-3
1-2 4-8 Haag    NNP SBJ 1-3
1-3 9-14    plays   VBD P|ROOT  1-5|1-3
1-4 15-22   Elianti NNP OBJ 1-3
1-5 23-24   .   .   _   _
```

In this example (say token 1-1), column 4 (NNP) is a value for the feature PosValue of the **de.tudarmstadt.ukp.dkpro.core.api.lexmorph.type.pos.POS** layer. Column 5 (SUBJ) records the value for the feature **DependencyType** of the **de.tudarmstadt.ukp.dkpro.core.api.syntax.type.dependency.Dependency** relation layer, where as column 6 (1-3) shows from which governor (VBD) the dependency arc is drawn.

For relations, a single disambiguation ID is not sufficient. If a relation is ambiguous, then the source ID of the relation is followed by the source and target disambiguation ID separated by an underscore (_). If only one of the relation endpoints is ambiguous, then the other one appears with the ID 0. E.g. in the example below, the annotation on token 1-5 is ambiguous, but the annotation on token 1-1 is not.

Example: Disambiguation IDs in relations

```
#FORMAT=WebAnno TSV 3.2
#T_SP=de.tudarmstadt.ukp.dkpro.core.api.ner.type.NamedEntity|value
#T_RL=webanno.custom.Relation|value|BT_de.tudarmstadt.ukp.dkpro.core.api.ner.type.Name
dEntity


#Text=This is a test .
1-1 0-4 This  * _ _
1-2 5-7 is  _ _ _
1-3 8-9 a _ _ _
1-4 10-14 test  _ _ _
1-5 15-16 . *[1]|*[2] * 1-1[0_1]
```

Troubleshooting

If the tool is kept open in the browser, but not used for a long period of time, you will have to log in again. For this, press the reload button of your browser.

If the tool does not react for more than 1 minute, please also reload and re-login.

We are collecting error reports to improve the tool. For this, the error must be reproducible: If you find a way how to produce the error, please open an issue and describe it.

[[Configurable Settings]] == Configurable Settings

There are multiple settings that can be configured in the file `settings.properties`. The file must be created by the user and put under the root directory under `.inception` For details, please refer to the Administrator Guide.