INCEpTION User Guide

The INCEpTION Team

Version 0.13.0

Table of Contents

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.

Introduction

Getting started

In order to run INCEpTION, you need to have Java installed on your system in version 8 or higher. If you do not have Java installed yet, please the latest Oracle Java or OpenJDK.

Download the stand-alone JAR from the INCEpTION downloads page.

Start the application simply by **double-clicking** on the download JAR file in your file manager. After a moment, a splash screen will be displayed while the application is initializing. Once the initialization is complete, a dialog will appear which you can use to open the application in your default browser or to shut down the application.

Alternatively, you can start it on the command line using

```
$ java -jar inception-app-standalone-0.13.0.jar
```

The splash screen and dialog will not appear in this case and you have to manually point your browser at `http://localhost:8080`.

The first time you start the application, a default user with the name `admin` and the password `admin` is created. Use this username and password to log in to the application after opening it in your browser.

Your first annotation

After logging in, you will see the main menu. Click on **Projects** to go to the **project management page** and there click on **create** to start a new project. Enter a name for your project, select the type **annotation project** and press **save**.

Next, switch to the **Documents** tab and press **choose files** to select a plain text file from your local harddisk (the file should be in UTF-8 encoding). As part of the import process, INCEpTION automatically processes the file to identify sentence and token boundaries.

Use the **Home** link at the top of the screen to return to the main menu and select **Annotation** to open your text file in the annotation editor.

To create your first annotation, select **Named entity** from the **Layer** dropdown menu on the right side of the screen. Then, use the mouse to select a word in the **Annotation** area. When you release the mouse button, the annotation is immediately created and you can edit its details in the right sidebar.

Congratulations! You have created your first annotation project.

Where to go from here

To familiarize yourself with the functionalities of INCEpTION, try importing some of the INCEpTION example projects.

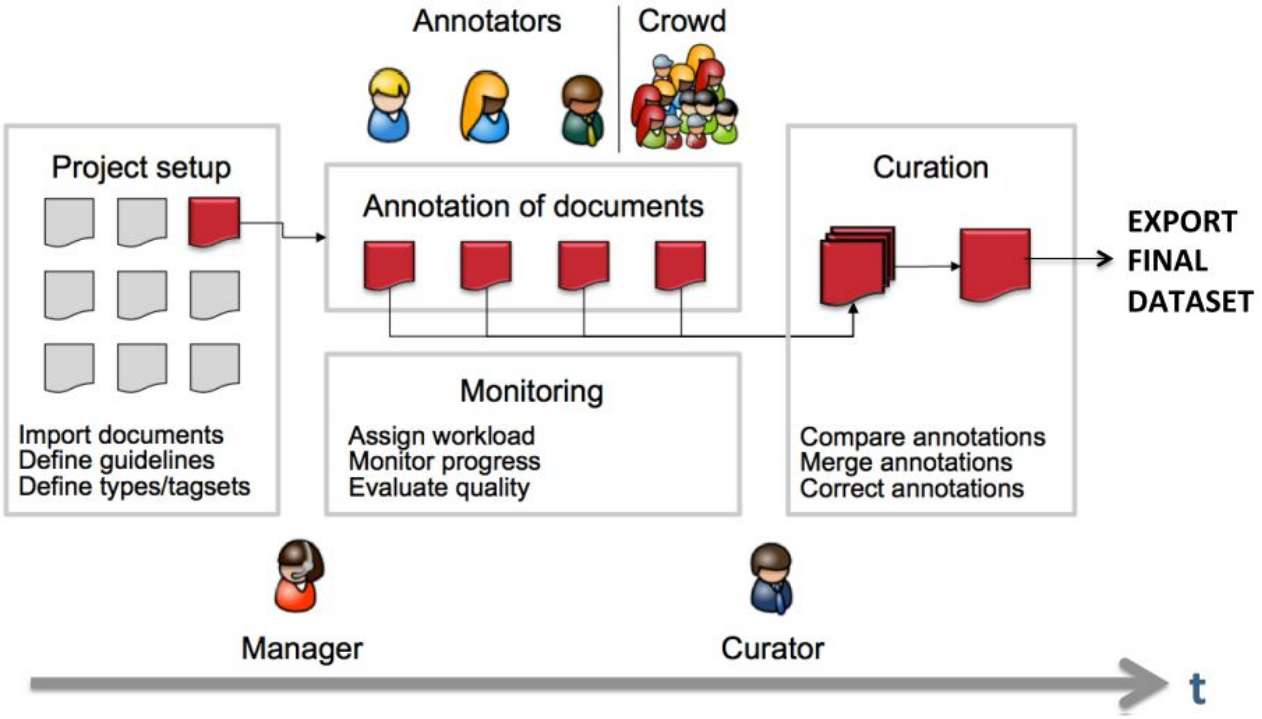
Running INCEpTION in the way you just did is a great way to get started and try out its capabilities, but it is not the best way of working with the application. If you like INCEpTION, please be sure to check out the Administrator Guide to learn how to set up a production-ready instance.

By default, INCEpTION creates and uses an embedded database and stores all its data in folder called `.inception` (*dot inception*) within your home folder. While this allows you to get started very quickly in trying out the application, it is not a recommended configuration for serious use. For production use, please configure INCEpTION to use a database server when using it in production. For more information, please refer to the Administrator Guide.

By default the server starts on port 8080 and you can access it via a browser at `http://localhost:8080` after you started it. But if you already have a service running on that port, 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).

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 managing 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.

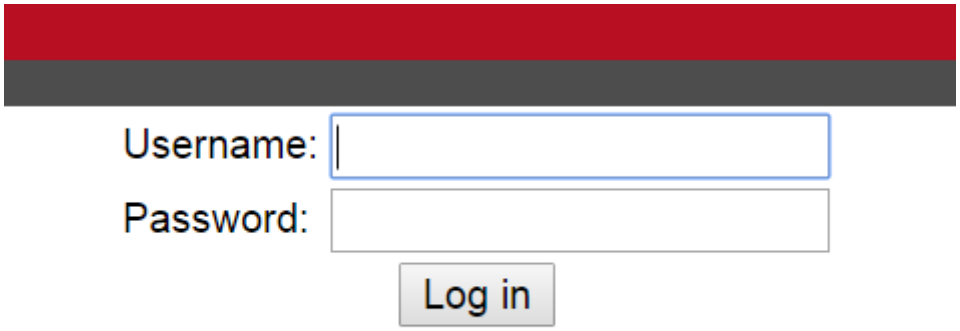
The document merged by the curator can be exported as soon as the curator clicked on **Done** for the document. The extraction of curated documents is also explained in Projects.

Core functionalities

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 password for this user after logging in (see User Management).



Dashboard

The dashboard allows you to navigate the functionalities of INCEpTION.

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.
- **Administration** - takes you to the administrator dashboard which allows configuring projects or managing users. This item is only available to administrators.
- **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.

Project overview

After logging in to INCEpTION, the first thing you see is the project overview. Here, you can see all the projects to which you have access. For every project, the roles you have are shown.

Using the filter toggle buttons, you can select which projects are listed depending on the role that you have in them. By default, all projects are visible.

Users with the role **project creator** can conveniently **create new projects** or **import project archives** on this page.

Users without a **manager** role can leave a project by clicking on the **Leave Project** button below the project name.

When uploading projects via this page, user roles for the project are not fully imported! If the importing user has the role **project creator**, then the **manager** role is added for the importing user. Otherwise, only the roles of the importing user are retained.

If the current instance has users with the same name as those who originally worked on the import project, the manager can add these users to the project and they can access their annotations. Otherwise, only the imported source documents are accessible.

Users with the role **administrator** who wish to import projects with all permissions and optionally create missing users have to do this through the Projects which can be access through the **Administration** link in the menu bar.

Project dashboard

Once you have selected a project from the Project overview, you are taken to this project's dashboard. Depending on the roles that a user has in the project, different functionalities can be accessed from here such as annotation, curation and project configuration. On the right-hand side of the page, some of the last activities of the user in this project are shown. The user can click on an activity to resume it e.g. if the user annotated a specific document, the annotation page will be opened on this document.

Annotation

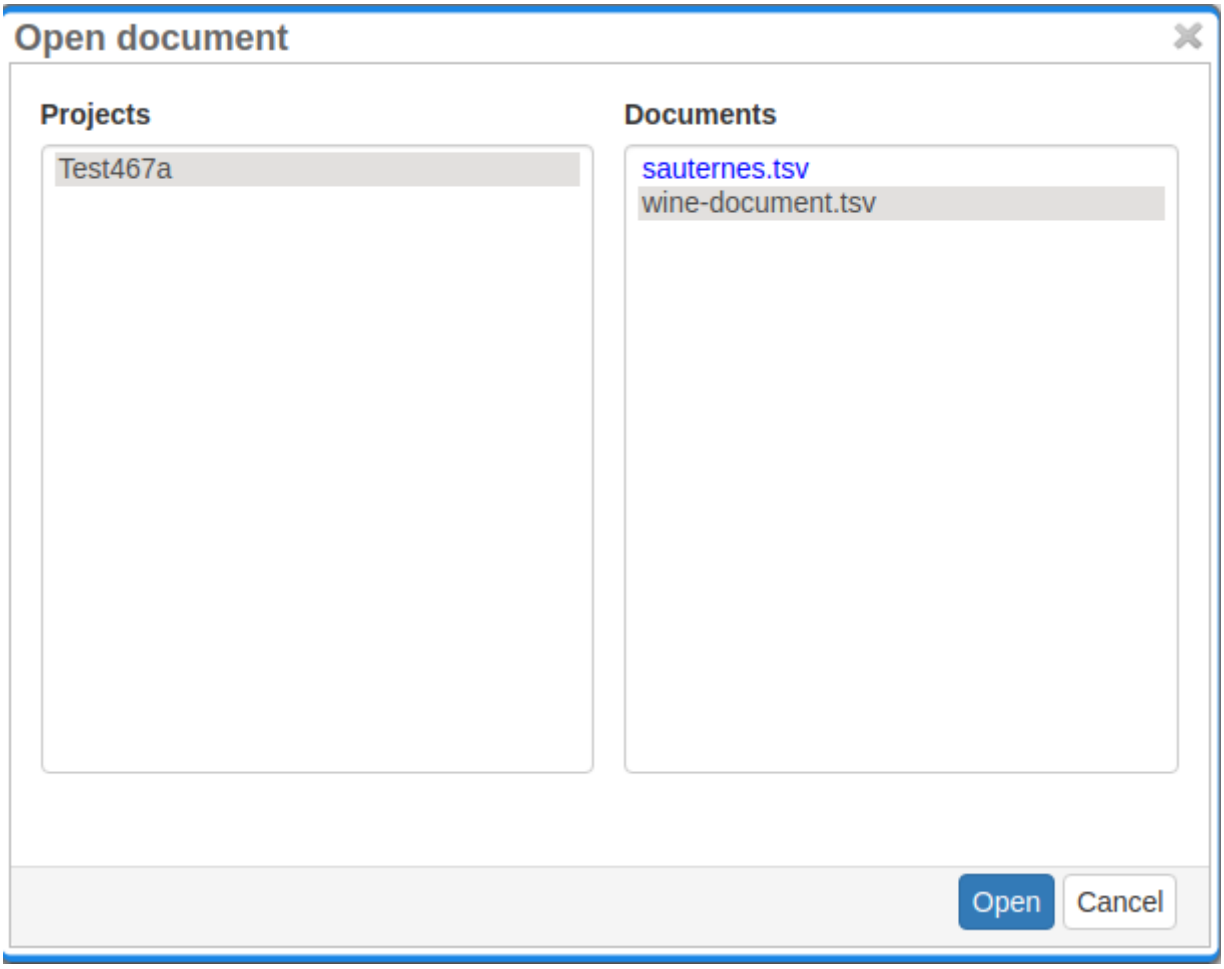
This functionality is only available to **annotators** and **managers**. Annotators and managers only see projects in which they hold the respective roles.

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

In addition to the default annotation view, PDF documents can be viewed and annotated using the PDF-Editor. Please refer to PDF Annotation Editor for an explanation on navigating and annotating in the PDF-view.

Opening a Document

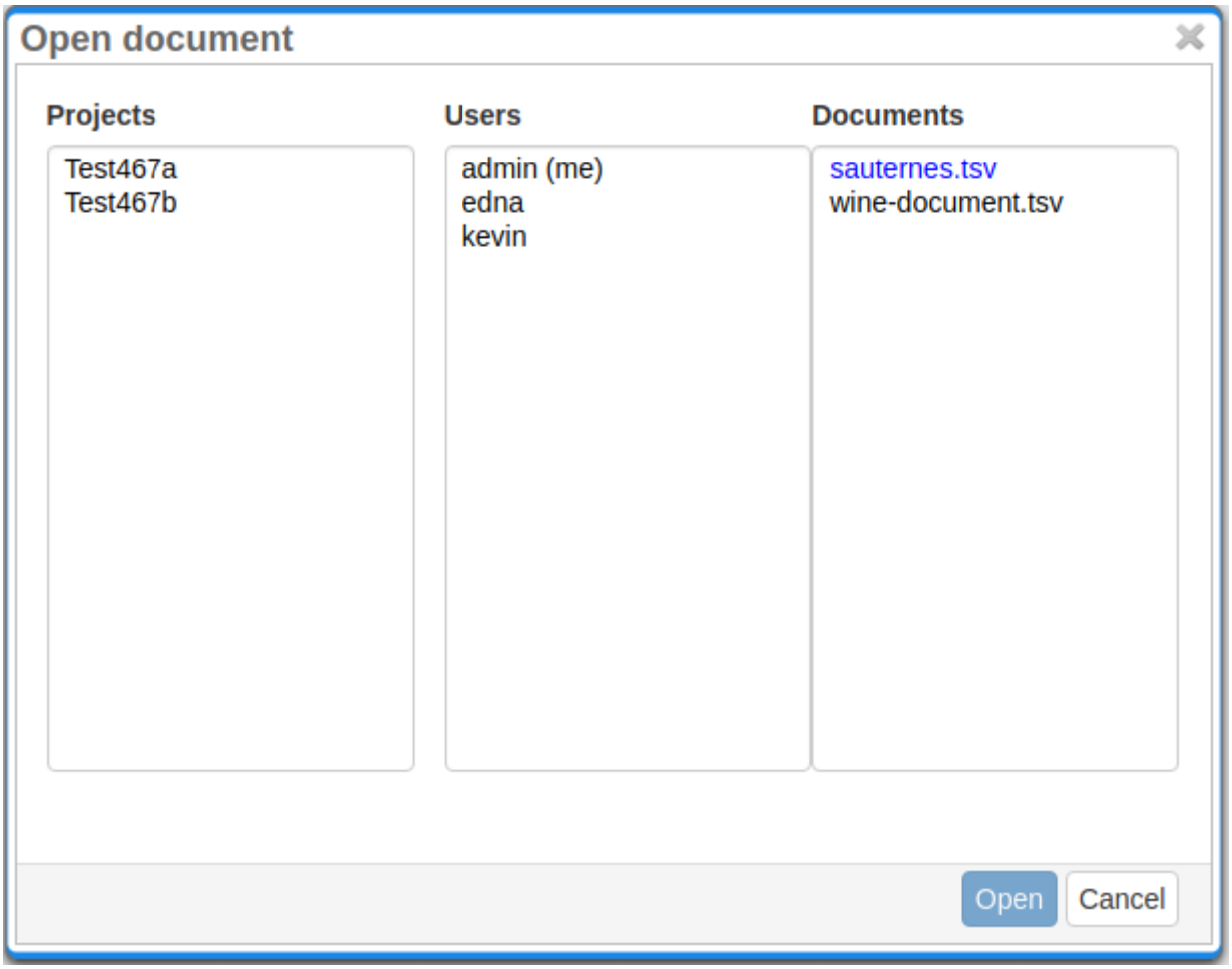
When navigating to the **Annotation** page, a dialogue opens that allows you to select a project, and a document within the project. If you want to open a different project or document later, click on **Open** to open the dialog.



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**.

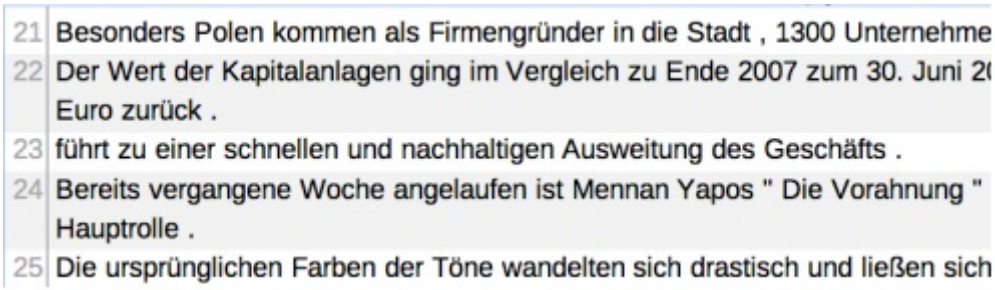
Users that are managers can additionally open other users' documents to view their annotations but cannot change them. This is done by selecting the project, user and then document in the

described dialogue. The user's own name is listed at the top and marked *(me)*.



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.

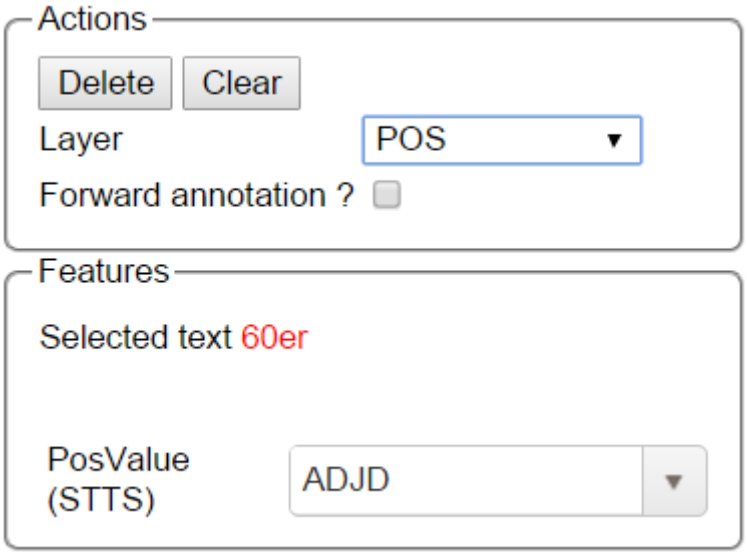
Table 1. Navigation key bindings

Key	Action
HOME	jump to first sentence

Key	Action
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

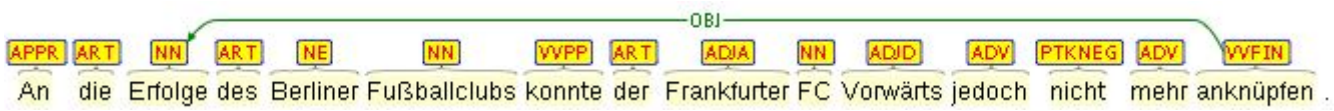
A click on the **Help** button displays the Guidelines for the tool and **The Annotator's Guide to NER-Annotation**. When you are finished with annotating or curating a document, please click on the **Done** button, so that the document may be further processed. If the button above the **Done** is a 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.

Once 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.

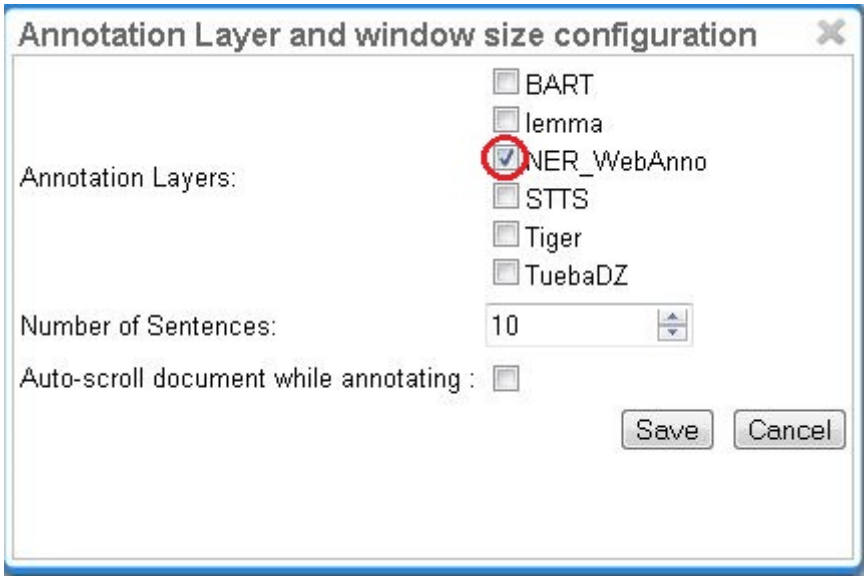
Spans

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**.

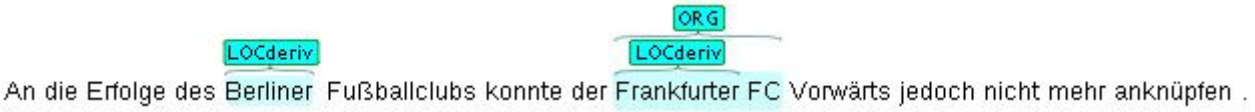
Depending on the layer behavior configuration, span annotations can have any length, can overlap, can stack, can nest, and can cross sentence boundaries.

Example

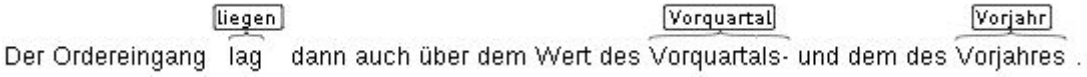
For example, for NE annotation, select the options as shown below (red check mark):



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 be rendered at the right side of `end` rather than at the left side of `.`.

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

Forward annotation mode

The **forward annotation** mode is useful for annotation tasks where every single token should be receiving an annotation - typical examples are part-of-speech of lemma annotation. When this mode is enabled, completing an annotation automatically creates another annotation on the next token.

The forward annotation mode is available for layers fulfilling the following conditions:

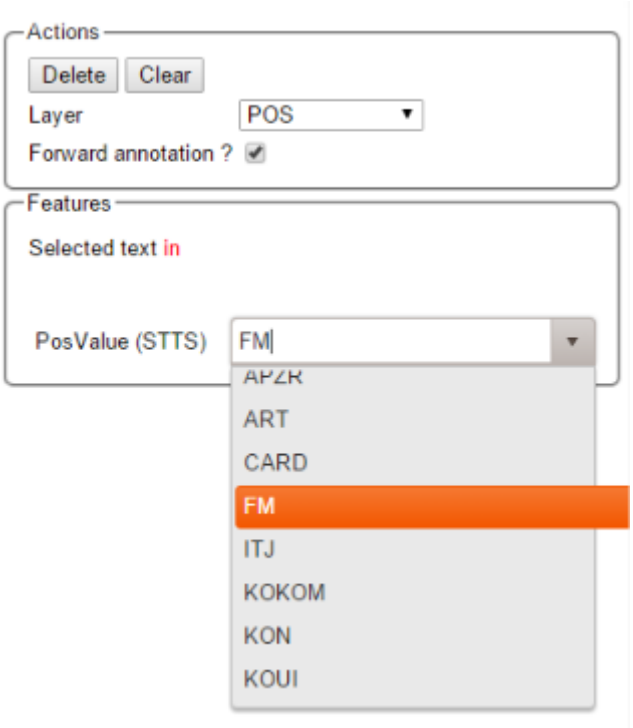
- the layer is a **span** layer
- the layer anchors to **single tokens**
- there is exactly one **enabled** and **visible String** feature
- if the feature uses a tagset, this tagset must be non-empty

If the feature is associated with a tagset, you will notice that the cursor does not jump to the feature editor when an annotation is created. This is intentional. Assume your tagset includes the tags `ADJ`, `ADV` and `NOUN`. When you press the `A` key once, the feature editor loads the first value starting with

that letter, i.e. ADJ. Press A again to move to the second tag ADV. Pressing a key multiple times cycles through all the tags starting with the respective letter. Thus, pressing A a third time loads ADJ again. Pressing N loads NOUN.

If the feature is associated with a tagset, pressing the **BACKSPACE** key deletes the current annotation and moves on. Otherwise the current annotation is removed if the feature editor is empty (no value entered) when completing the annotation.

Press **ENTER** to complete the annotation and to move on to the next token. In the general case, a new annotation is created on the next token and it is loaded into the feature editor. However, if it is not permitted to stack annotations or make them overlap (i.e. if the value of the behaviour setting for **Overlap** is set to anything else than **Any**) and if there is already an annotation on that token, this would naturally fail. In this case, no new annotation is created and instead an existing annotation is opened for editing.



If the **Remember layer** setting is turned on, it is possible to select and edit an annotation which is not on the layer for which the forward-mode is enabled. In this case, the forward-mode is paused and the user can edit the annotation normally. The forward-mode resumes when the user creates a new annotation or selects an annotation on the forward-enabled layer.

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.

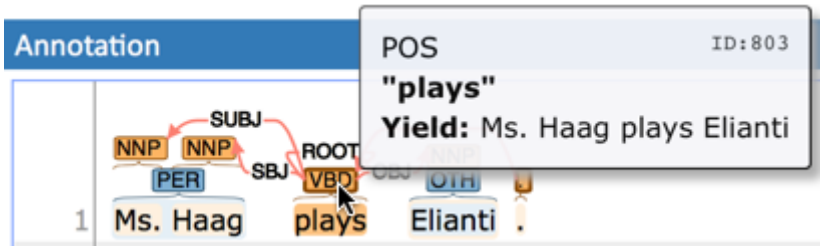


Figure 1. Example of the yield of a dependency relation

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.

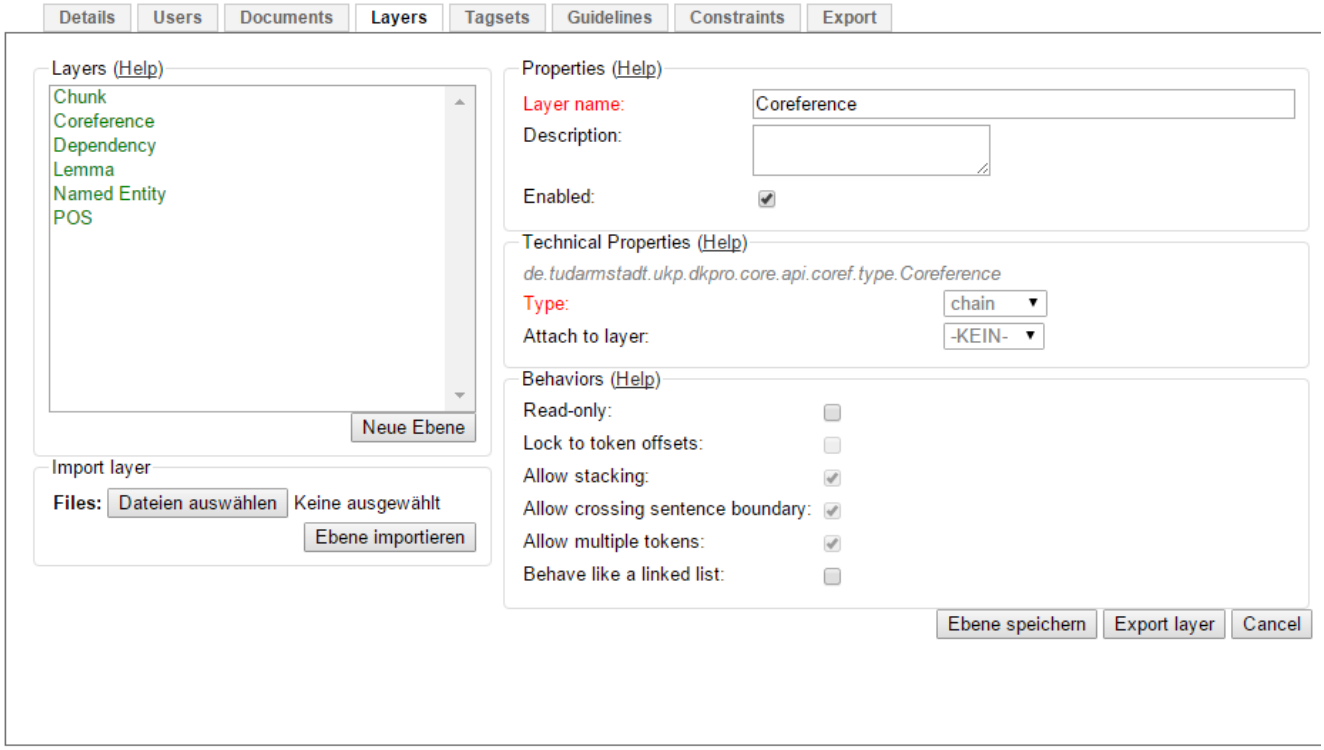


Figure 2. Configuration of a chain layer in the project settings

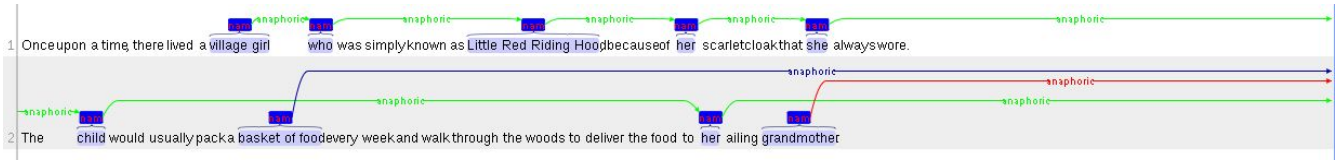


Figure 3. Example of chain annotations

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

Table 2. Chain behavior

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

Linked List	Condition	Result
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.

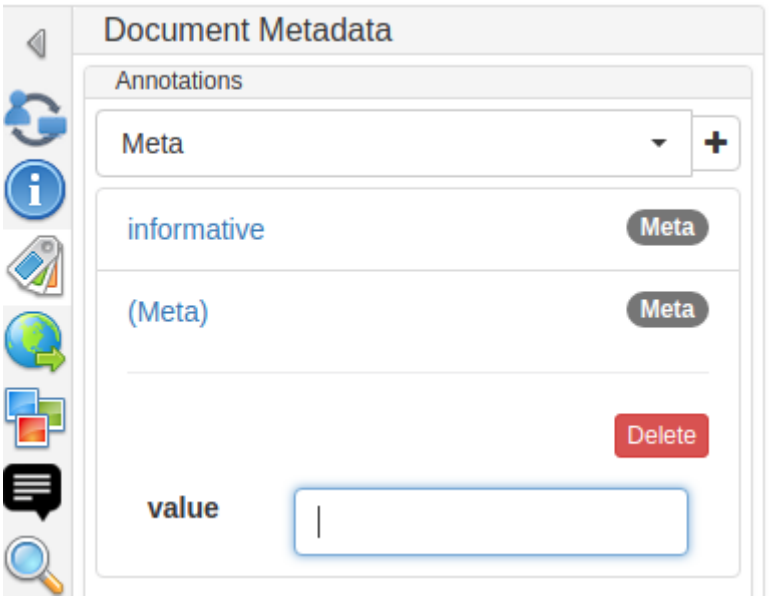
Document metadata

Before being able to configure document-level annotations, you need to define an annotation layer of type **Document metadata** on the project **Settings**, **Layers** tab. For this:

- Go to **Settings** → **Layers** and click the **Create** button
- Enter a name for the annotation layer (e.g. `Author`) and set its type to **Document metadata**
- Click **Save**
- On the right side of the page, you can now configure features for this annotation layer by clicking **Create**
- Again, choose a name and type for the feature e.g. `name` of type **Primitive: String**
- Click **Save**

On the annotation page, you can now:

- Open the **Document Metadata sidebar** (the tags icon) and
- Choose the newly created annotation layer in the dropdown.
- Clicking the plus sign will add a new annotation whose feature you can fill in.



This is an experimental feature not recommended for serious use. However, you can try it out and enable it by adding `documentmetadata.enabled=true` to the `settings.properties` file. In particular, curation of document metadata annotations is not possible and they are not supported by the TSV format. They can be im-/exported via XMI or Binary CAS though.

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. However if an integer feature is limited and the difference between the maximum and minimum is lower than 12 it can also be displayed with a radio button group instead. String features are displayed using a text field or a text area with multiple rows. The text area can be expanded if focused and collapses again if focus is lost. In case the string feature has a tagset it will instead use a combobox.

Link Features

Link features can be used to link one annotation to others. Before a link can be made, a slot must be added. If role labels are enabled 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.

Role labels

If role labels are disabled for the link feature layer they cannot be manually set by the user. Instead the UI label of the linked annotation is displayed. If role labels are enabled they can be changed by the user at any time. To change a previously selected role label, 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 label.

Image Features

Image URL features can be used to link a piece of text to an image. The image must be accessible via an URL. When the use edits an annotation, the URL is displayed it the feature editor. The actual images can be viewed via the image sidebar.

Concept features

Concept features are shown as an auto-complete field. Typing into the field triggers a query against the knowledge base and displays candidates in a dropdown list.

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.

The **Editor** setting can be used to switch between different modes of presentation. It is currently only available on the annotation page.

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 **Font zoom** setting controls the font size in the annotation area. This setting may not apply to all editors.

The **Page size** controls how many sentences are visible in the annotation area. The more sentences are visible, the slower the user interface will react. This setting may not apply to all editors.

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 **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. This setting may not apply to all editors.

The **Collaps arcs** setting controls whether long ranging relations can be collapsed to save space on screen. This setting may not apply to all editors.

The **Read-only palette** controls the coloring of annotations on read-only layers. This setting overrides any per-layer preferences.

Layer preferences

In this section you can select which annotation layers are displayed during annotation and how they are displayed.

Hiding layers 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 **Palette** setting for each layer controls how the layer is colored. There are the following options:

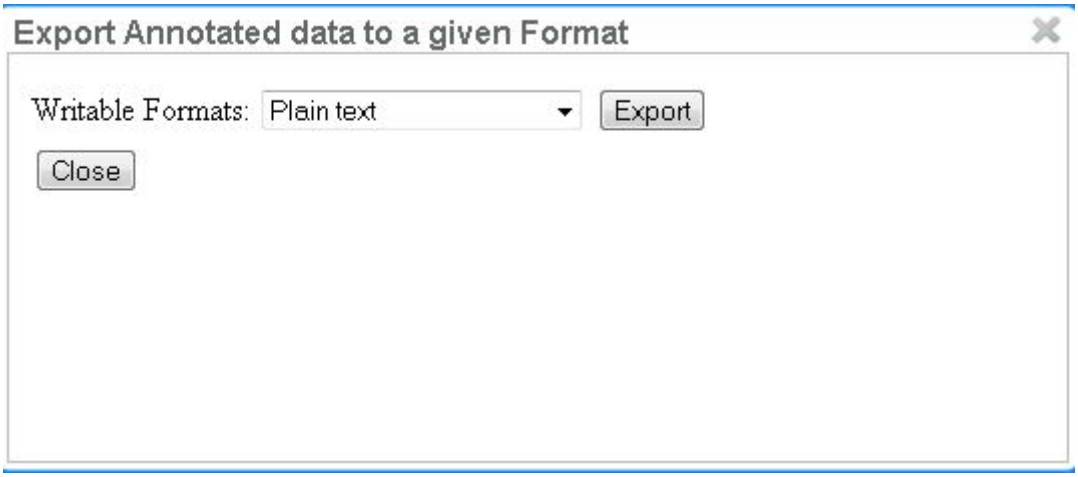
- **static** / **static pastelle** - all annotations receive the same color
- **dynamic** / **dynamic pastelle** - all annotations with the same label receive the same color. Note that this does not imply that annotations with different labels receive different colors.
- **static grey** - all annotations are grey.

Mind that there is a limited number of colors such that eventually colors will be reused.

Annotations on chain layers always receive one color per chain.

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:



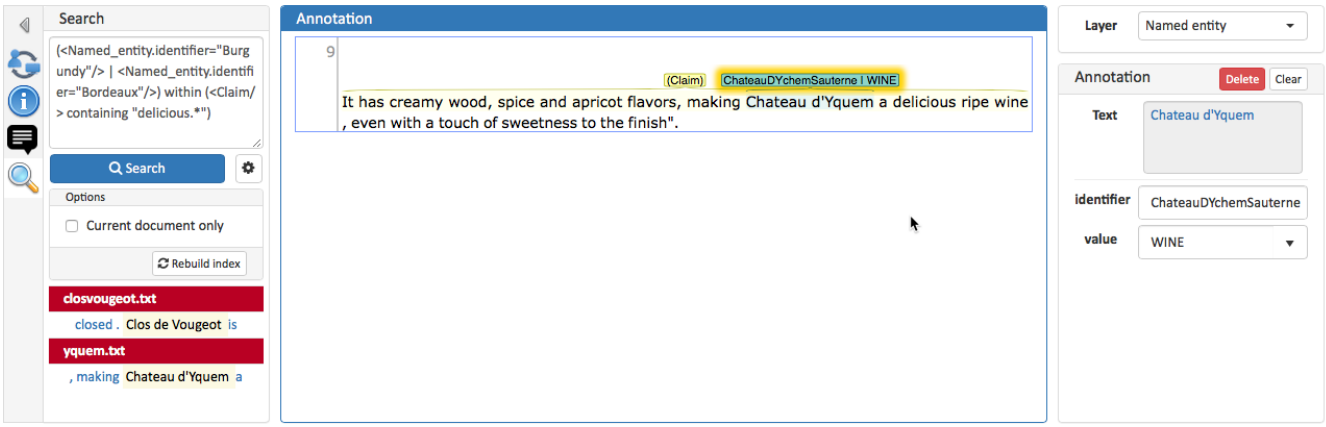
Choose your preferred format. Please take note of the facts that the plain text format does not contain any annotations and that the files in the binary format need to be unpacked before further usage. For further information the supported formats, please consult the corresponding chapter Formats.

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

Search

The search module allows to search for words, passages and annotations made in the documents of a given project. 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).

To perform a search, access the search sidebar located at the left of the screen, write a query and press the **Search** button. The results are shown below the query in a KWIC (keyword in context) style grouped by document. Clicking on a result will open the match in the main annotation editor.



The search only considers documents in the current project and only matches annotations made by the current user.

Very long annotations and tokens (longer than several thousand characters) are not indexed and cannot be found by the search.

Clicking on the search settings button (cog wheel) shows additional options:

- **Current document only** limits the search to the document shown in the main annotation editor. When switching to another document, the result list does not change automatically - the search button needs to be pressed again in order to show results from the new document.
- **Rebuild index** may help fixing search issues (e.g. no or only partial results), in particular after upgrading to a new version of INCEpTION. Note that this process may take quite some time depending on the number of documents in the project.
- **Grouping by** allows to group the search results by feature values of the selected annotation feature. By default the search results will be grouped by document title if no layer and no feature is selected.

Creating/Deleting Annotations for Search Results

The user can also use the search to create and/or delete annotations for a set of selected search results.

This means that annotations will be created/deleted at the token offsets of the selected search results. Search results can be selected via the checkbox on the left. Per default all search results are selected. If a result originates from a document which the user has already marked as **finished**, there is no checkbox since such documents cannot be modified anyway.

The currently selected annotation in the annotation editor serves as template for the annotations that are to be created/deleted. Note that selecting an annotation first is necessary for creating/deleting annotations in this way.

Clicking on the create settings button (cog wheel) shows additional options:

- **Override existing** will override an existing annotation of the same layer at a target location. If this option is disabled, annotations of the same layer will be stacked if stacking is enabled for this layer. Otherwise no annotation will be created.

Clicking on the delete settings button (cog wheel) shows additional options:

- **Delete only matching feature values** will only delete annotations at search results that exactly match the currently selected annotation including all feature values. If this option is disabled all annotations with the same layer as the currently selected annotation will be deleted regardless of their feature values.

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** .

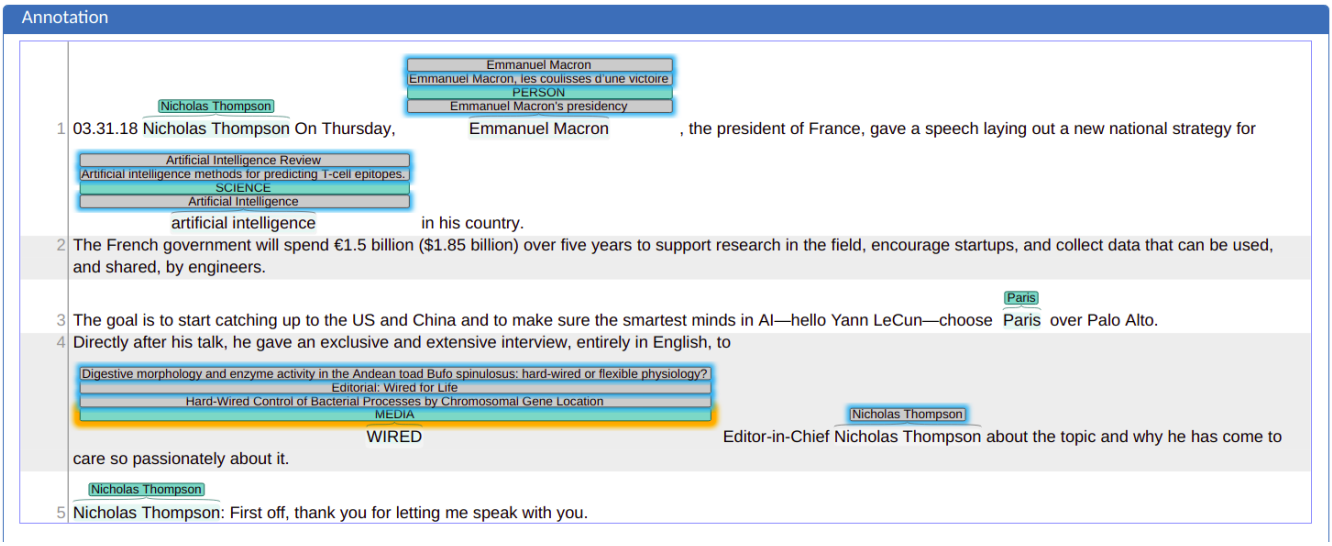
```
<Named_entity.identifier-exact="ChateauMorgonBeaujolais"/>
```

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

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

Recommenders

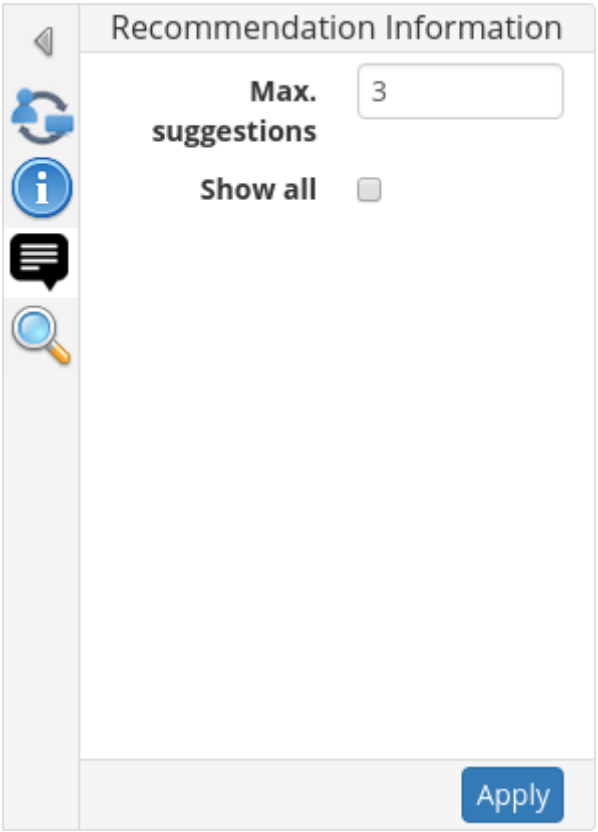
After configuring one or more recommender in the Project Settings, they can be used during annotation to generate predictions. In the annotation view, predictions are shown as grey bubbles next to normal annotations in blue. Predictions can be accepted by clicking once on them. In order to reject, use a double-click. For an example how recommendations look in action, please see the screenshot below.



Suggestions generated by a specific recommender can be deleted by removing the corresponding recommender in the Project Settings. Clicking **Reset** in the *Workflow* area will remove all predictions, however it will also remove all hand-made annotations.

Recommendation Sidebar

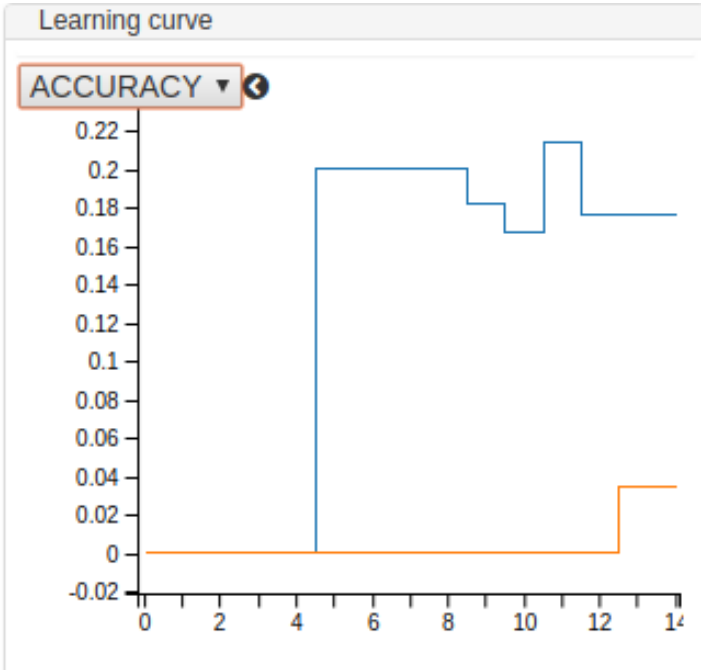
Clicking the speech bubble on the left opens the recommendation sidebar. There, one can set the maximum number of recommendations that are displayed for each token. Also, by ticking **Show all**, all recommendations are shown, even if an annotation at the given position already exists. In order to save the new values, press **Apply**.



Learning Curve Diagram

On the bottom of the recommendation sidebar, a learning curve diagram is shown. It plots a line for

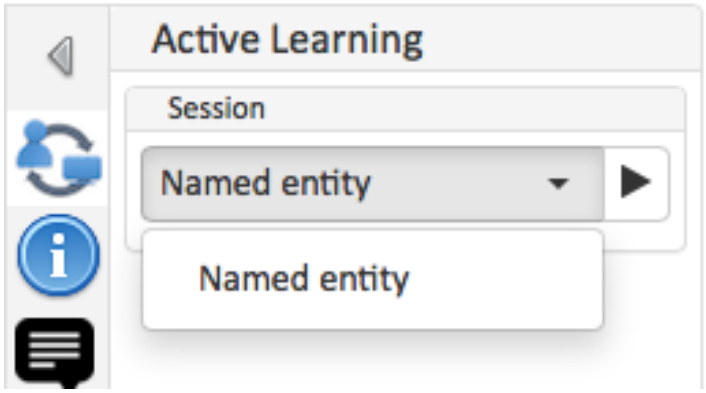
every recommender associated with the project. The learning curve line depicts the scores of a number of recent evaluations of a recommender. The y-axis of the chart represents the scores and the x-axis represents the time indices. For instance, if the recent 50 evaluations are plotted, the x-axis would be 0-50 with 50 being the latest point in time relative to the other points. Also, if the number of evaluations calculated so far are less than 50 then less points are shown on the x-axis. The shown score can be changed from accuracy to other scores via the retractable dropdown above the y-axis. The lines are only plotted for the enabled recommenders.



Active Learning

Active learning is a family of methods which seeks to optimize the learning rate of classification algorithms by soliciting labels from a human user in a particular order. This means that recommenders should be able to make better suggestions with fewer user interactions, allowing the user to perform quicker and more accurate annotations. Note that Active Learning only works if there are recommenders and if these recommenders actually generate recommendations which are usually shown as grey bubbles over the text.

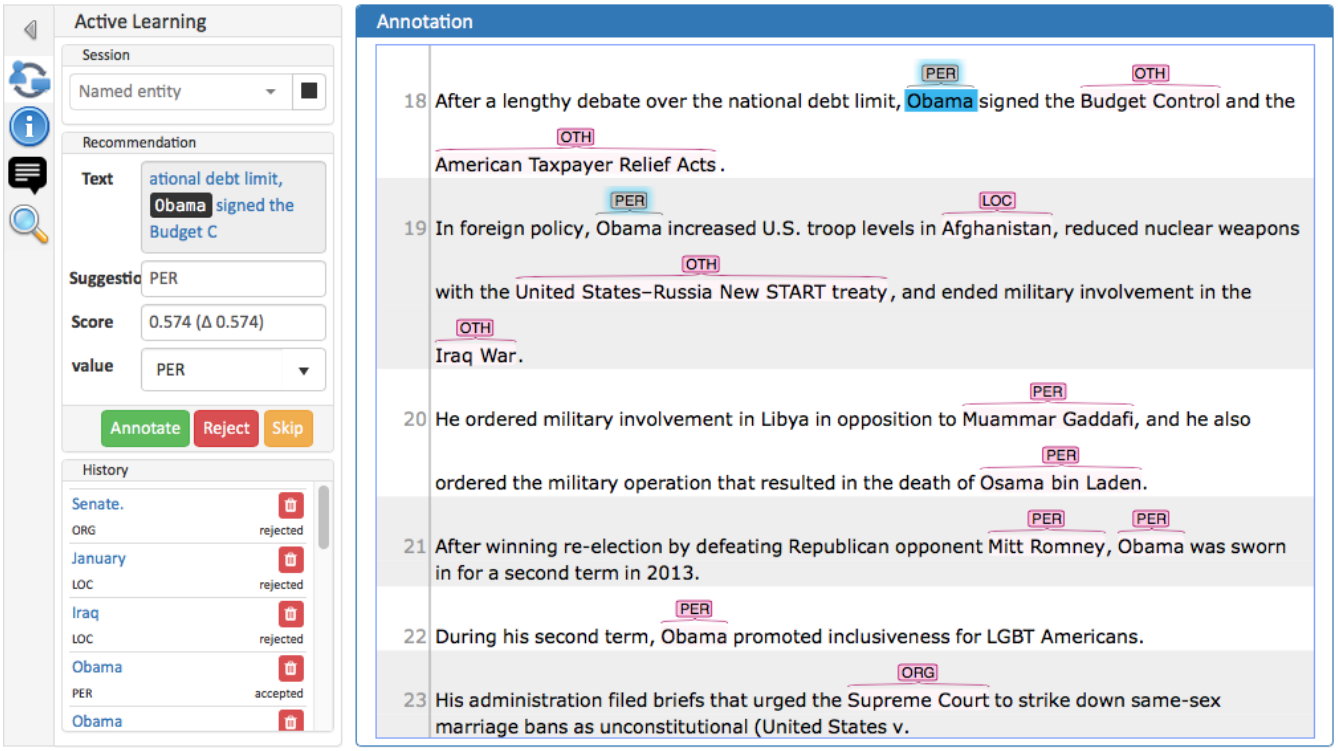
Open the Active Learning sidebar on the left of the screen. You can choose from a list of all layers for which recommenders have been configured and then start an active learning session on that layer.



The system will 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. Additionally, there is a field which shows the suggested label and which allows changing that label - i.e. to correct the suggestion provided by the system. The recommendation is also highlighted in the central annotation editor.

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



When using the *Annotate*, *Reject* or *Skip* buttons, the system automatically jumps to the next suggestion for the user to inspect. However, at times it may be necessary to go back to a recently inspected suggestion in order to review it. The **History** panel shows the 50 most recent actions. Clicking on the text of an item loads it in the main annotation editor. It is also possible to delete items from the history, e.g. wrongly rejected items.

The history panel displays whether a given suggestion was accepted, corrected or rejected, but this information can only be indicative. It represents a snapshot of the moment where the user made the choice. As the recommender is continuously updated by the system, the suggestions constantly change. It may happen that a suggestion which is shown as *rejected* in the sidebar is at a later time not even generated anymore by the recommender. Thus, deleting an item from the history will not always cause the suggestion from which it was generated to reappear. Resetting a document also clears the Active Learning history.

INCEpTION allows the user to create annotations as usual in the main annotation editor panel, even when in an Active Learning session. However, there is only a limited interaction with actions performed in the main annotation editor. If a suggestion is accepted or rejected in the main annotation editor, this is recorded in the history. However, if a user manually creates an annotation which causes a suggestion to disappear by overlapping with it, the history does not record this as a correction. For example, if the system generates a suggestion for `Paul.` (including the final sentence punctuation) but the user manually creates an annotation only for `Paul` (without the punctuation),

the system does not recognize it as a correction.

Accepting/correcting, rejecting and skipping a suggestion in the sidebar cause the main annotation editor to move to the next suggestion. However, when a suggestion is accepted or rejected via the main editor, the generated annotation is opened in the annotation detail editor panel on the right side and the editor does not move to the next suggestion. For actions made in the main editor, it is assumed that the user may want to perform additional actions (e.g. set features, create more annotations in the vicinity) - jumping to the next suggestion would interfere with such intentions. That said, the next suggestion is loaded in the active learning sidebar and the user can jump to it by clicking on the suggestion text in the sidebar.

When removing an accepted/corrected item from the history and the annotation which was generated from this item is still present (i.e. it has not been deleted by other means), the user is asked whether the associated annotation should also be deleted.

Suggestions that are skipped disappear at first. However, once all other suggestions have been processed, the system asks whether the skipped suggestions should now be reviewed. Accepting will remove all skipped items from the history (even those that might no longer be visible in the history because of its on-screen size limit).

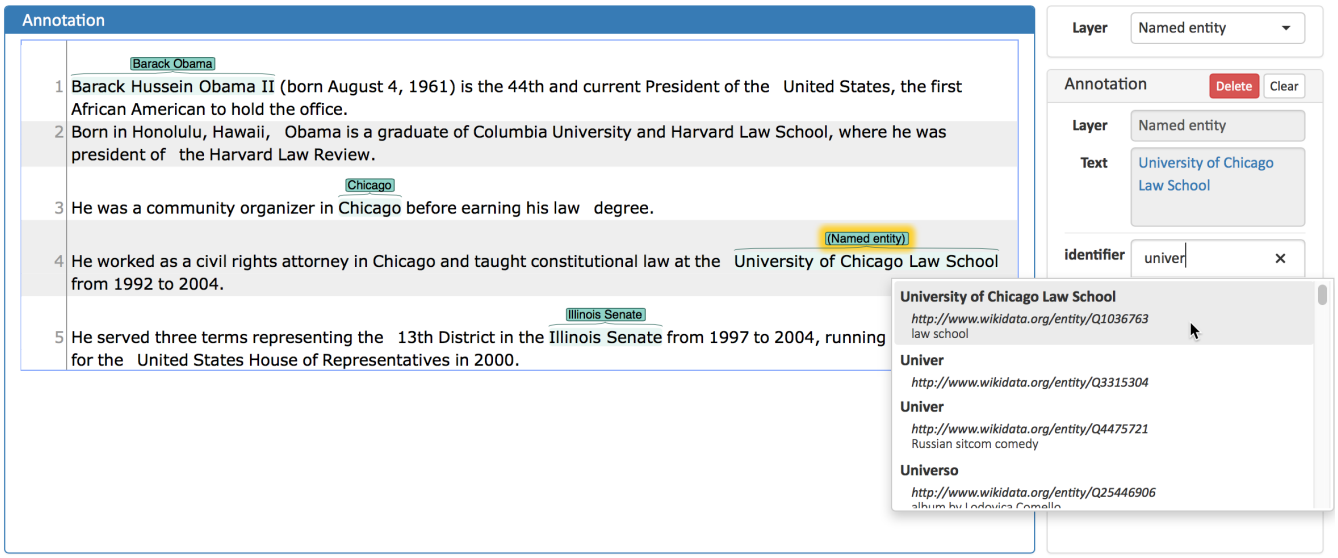
Concept Linking

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

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 annotation 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.

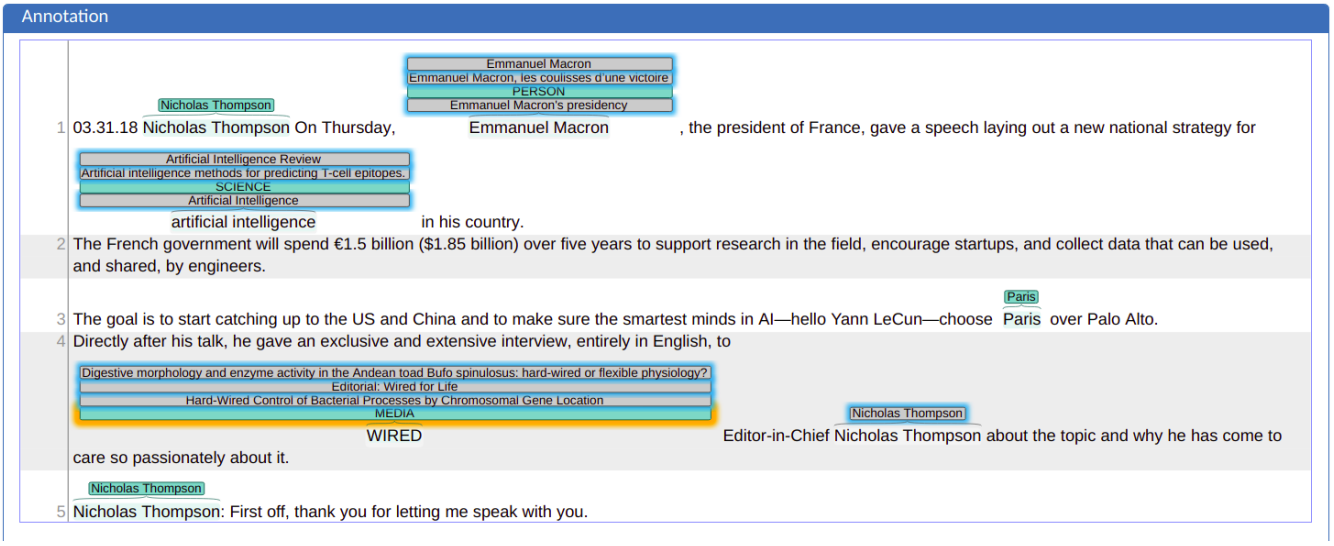
To link a concept mention to the knowledge base, first select the mention annotation, then select the concept feature in the right sidebar of the annotation editor and start typing the name of a concept. A ranked list of candidates is then displayed in the form of a drop-down menu. In order to make the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

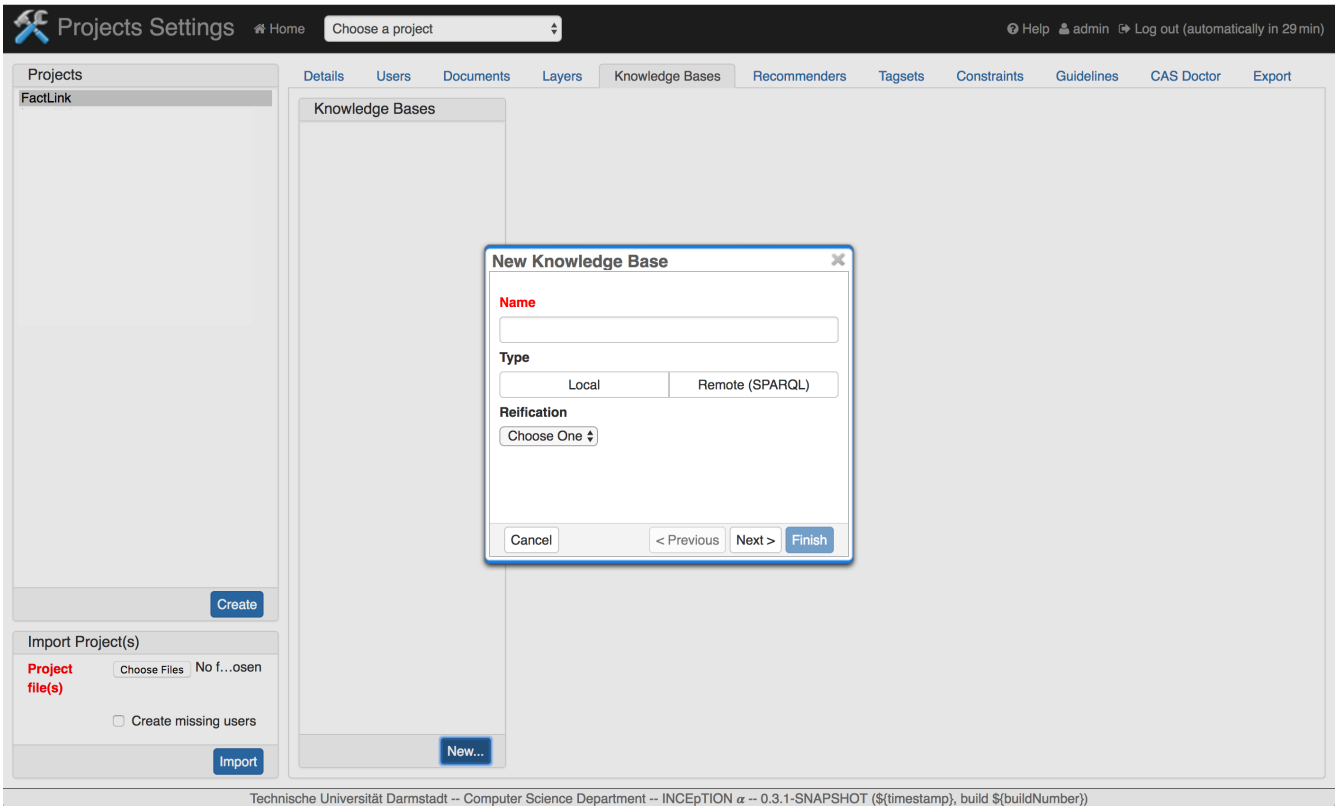
Fact extraction is the task of extracting facts from documents. It is defined that a fact consists of a subject, a predicate, an object and qualifiers. Fact extraction in INCEpTION includes annotating the mentions for each fact component, and linking these mentions to instances of concept classes or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

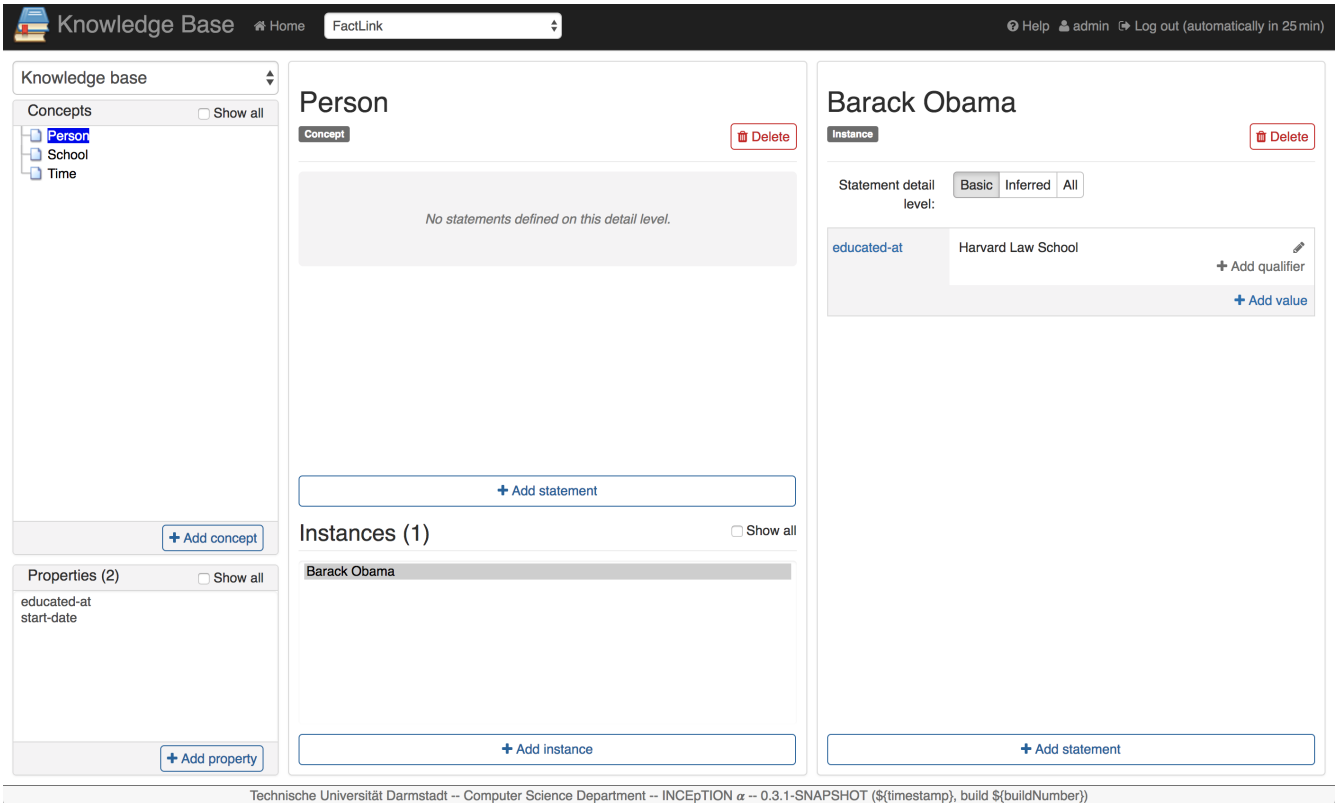
In the **Projects Settings**, switch to the **Knowledge Bases** tab, then click **New...** on the bottom and a dialogue box shows as in the figure below.



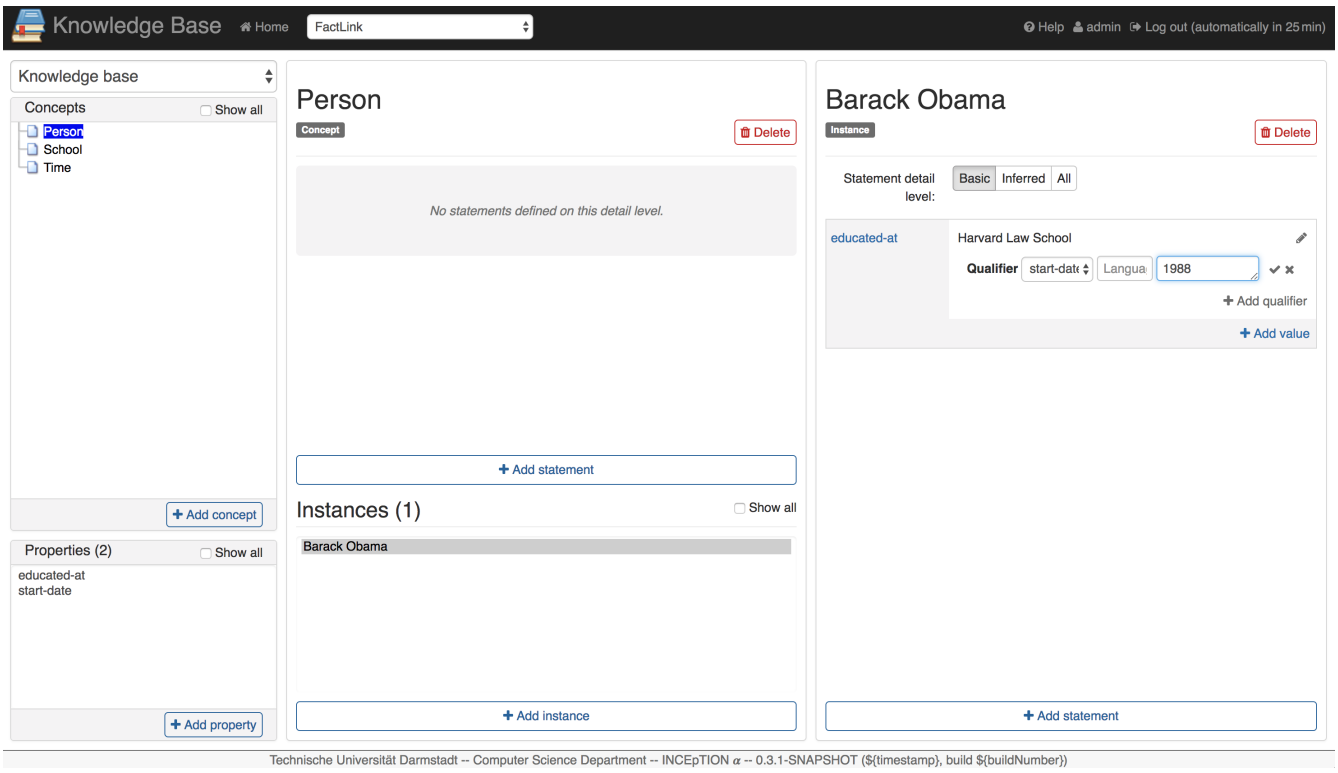
To create a local knowledge base, one needs to choose **Local** for the type. For the reification, **NONE** is the defaut case, but one cannot add or view qualifiers in the knowledge base with **NONE**. So, to support qualifiers, one needs to choose **WIKIDATA** for the Reification. One can then follow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



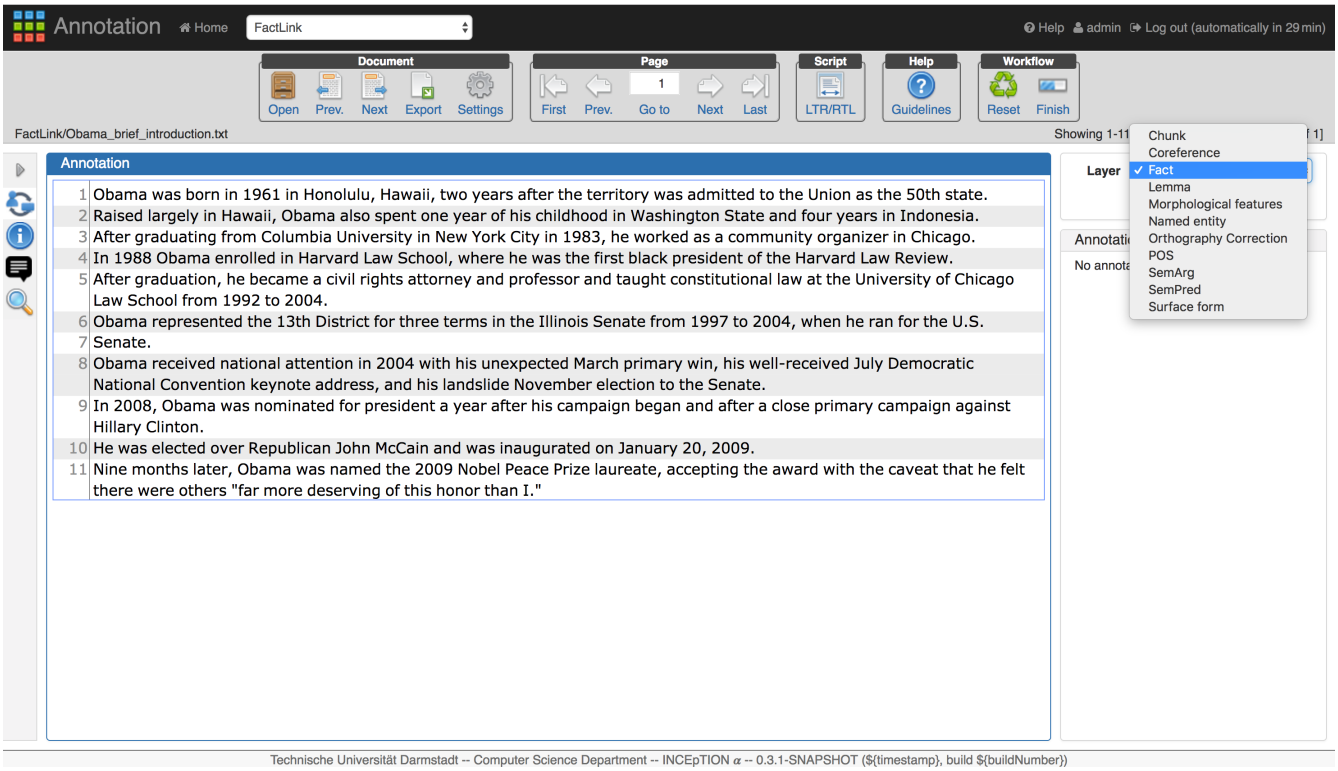
One can click the button + **Add qualifier**, choose a property *start-date* for this qualifier name, enter *1988* for this qualifier value, and click √ to save this qualifier.



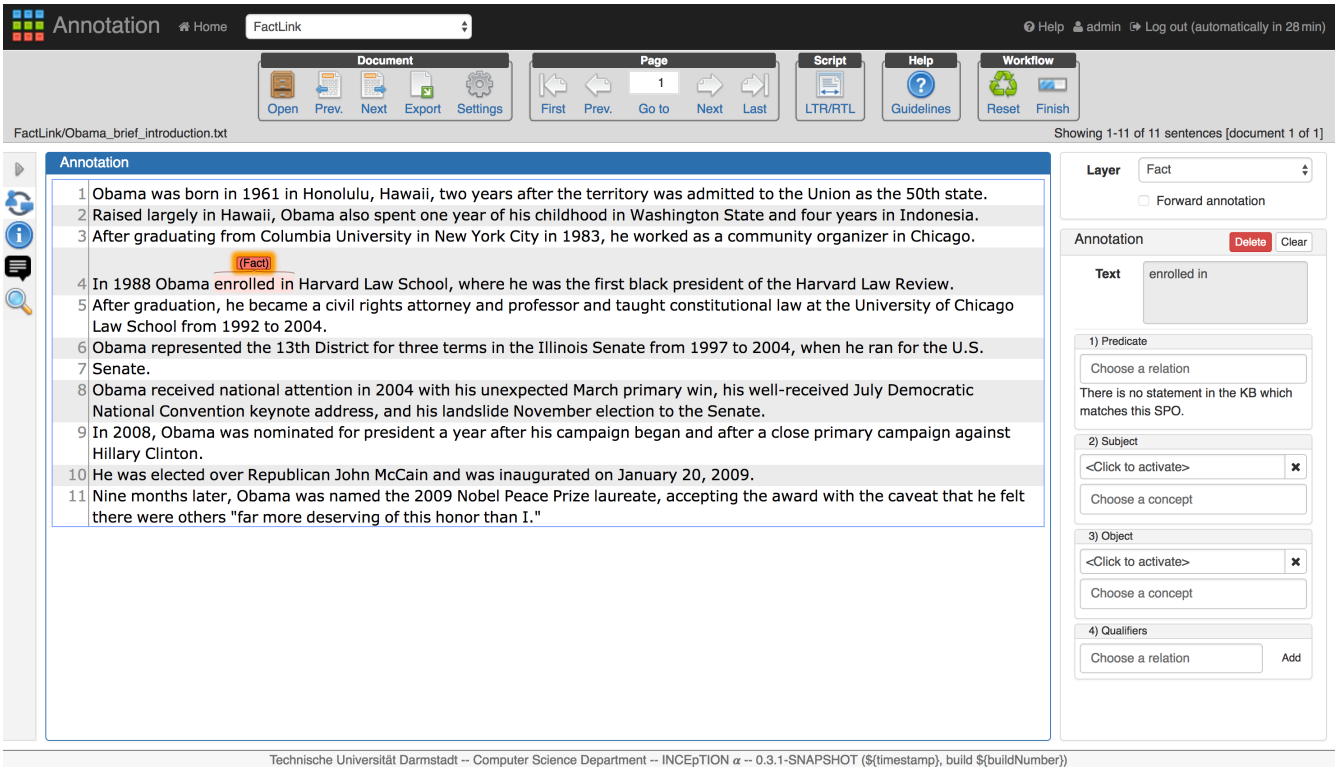
One can click the pencil icon on the same line as this qualifer to edit and delete this qualifier.

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:

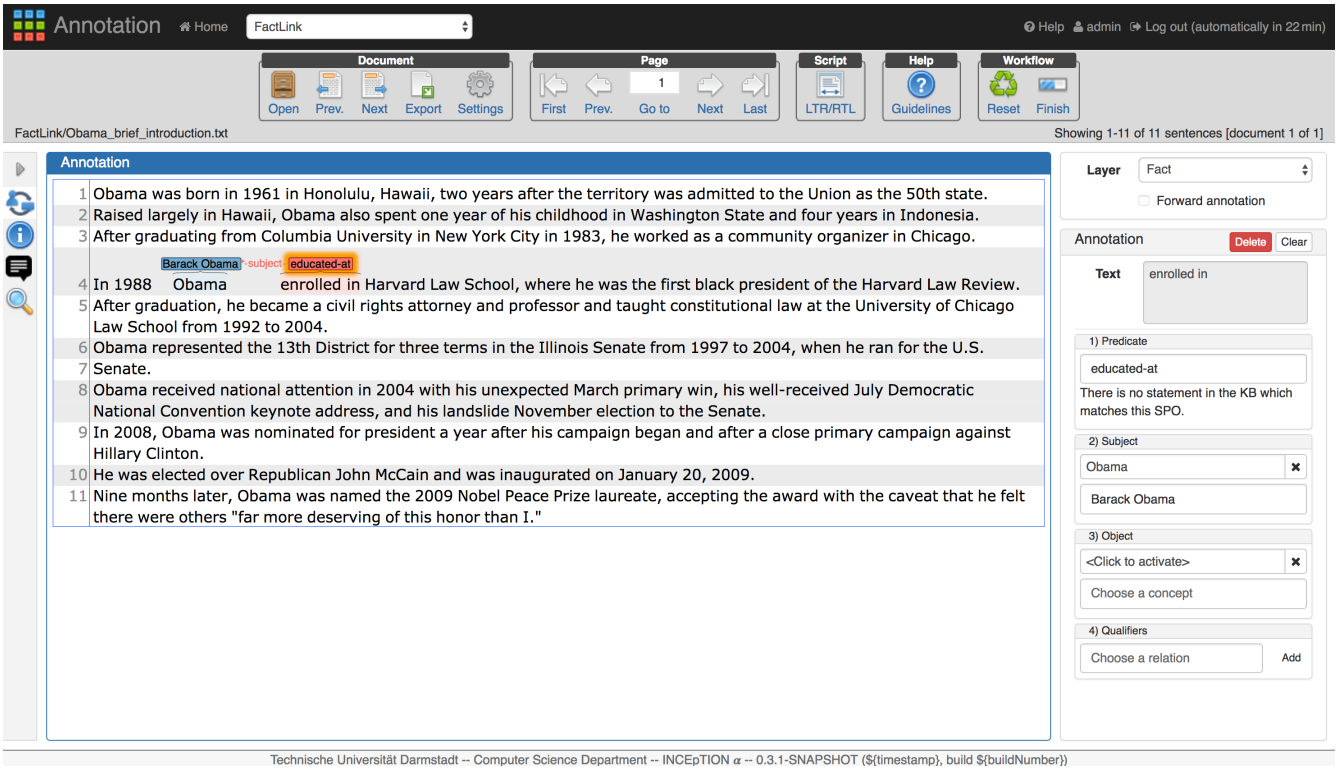


One can now mark the predicate mention of a fact. In the screenshot below, *enrolled in* is selected. The right sidebar shows feature editors for the predicate, the subject, the object and the qualifiers. One can then choose *educated-at* in the dropdown field of the predicate feature editor to link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

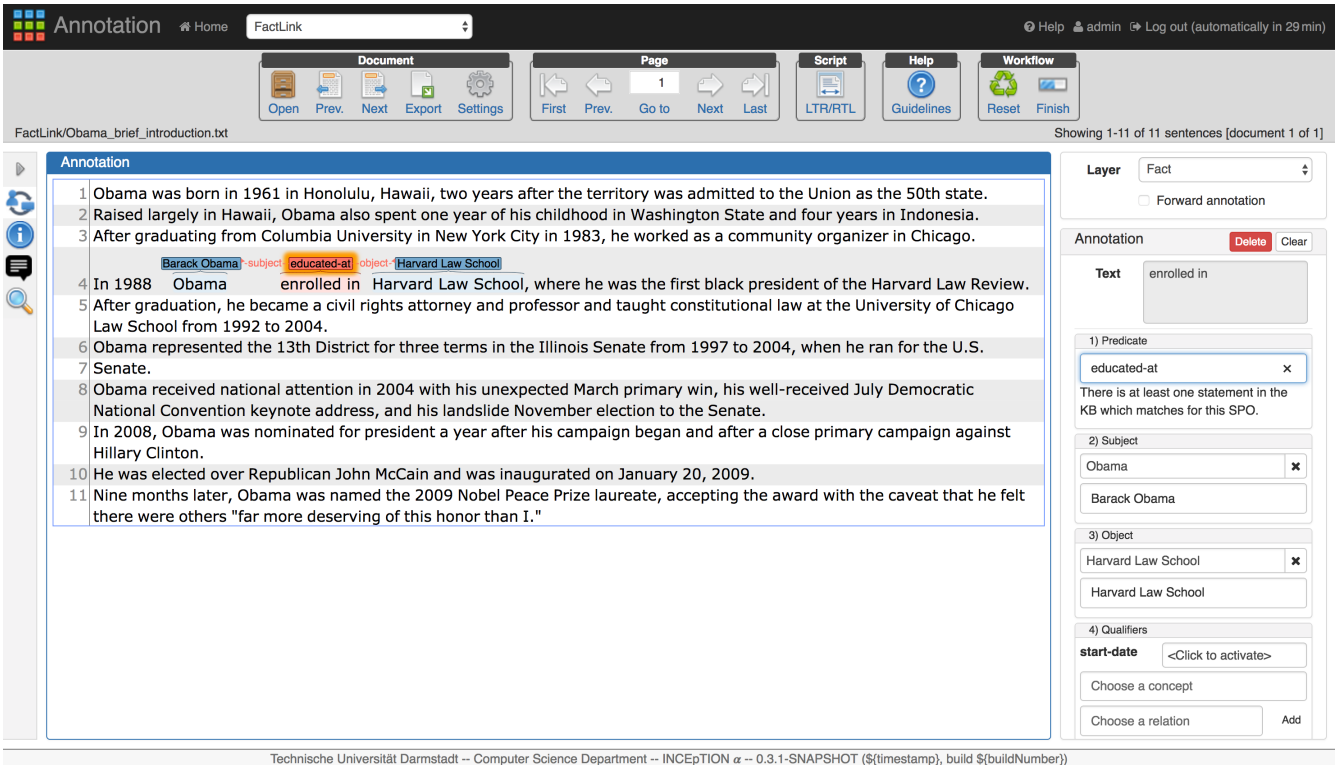
dropdown field is a list of instances in the knowledge base. The default setting returns all the instances from all the knowledge base in this project. See Fact linking with multiple knowledge bases to select a specific knowledge base. After linking the subject to the knowledge base instance, in the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



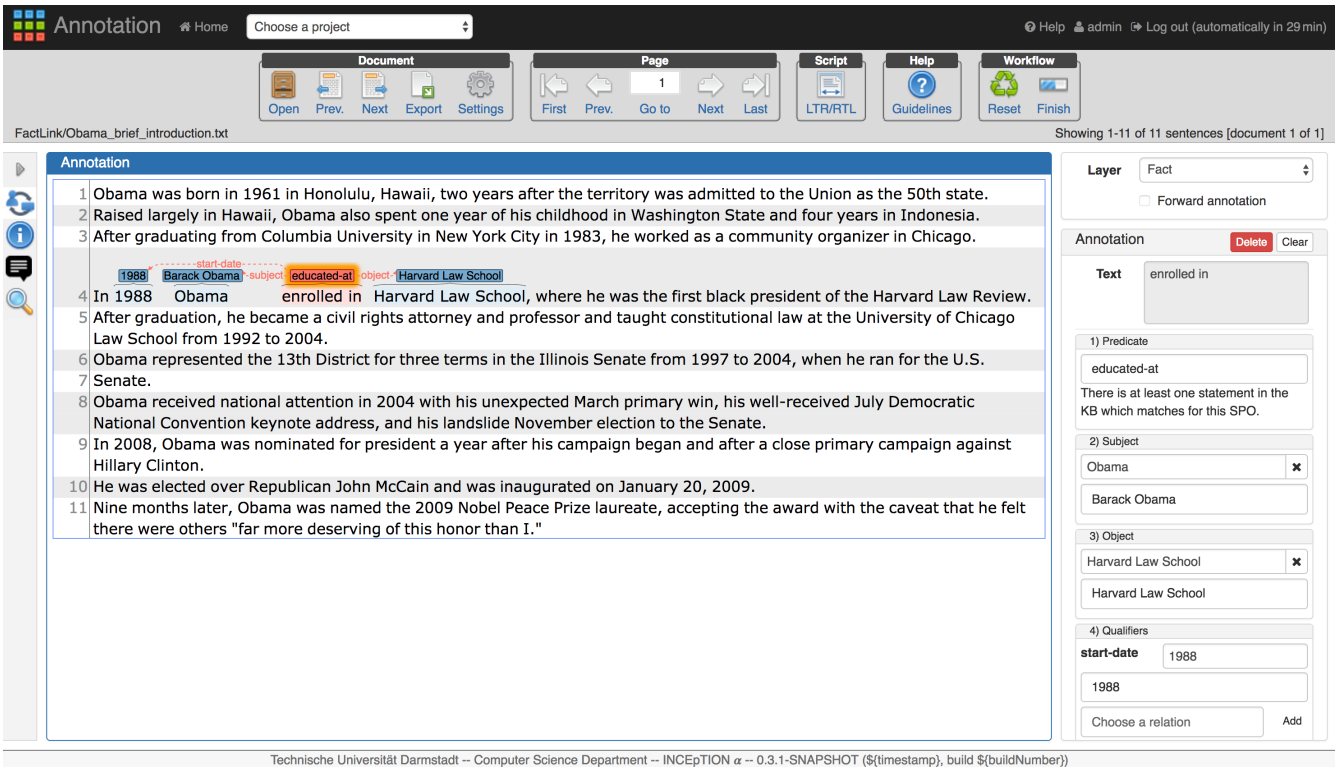
Extracting the object mention is same as extracting the subject.

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.

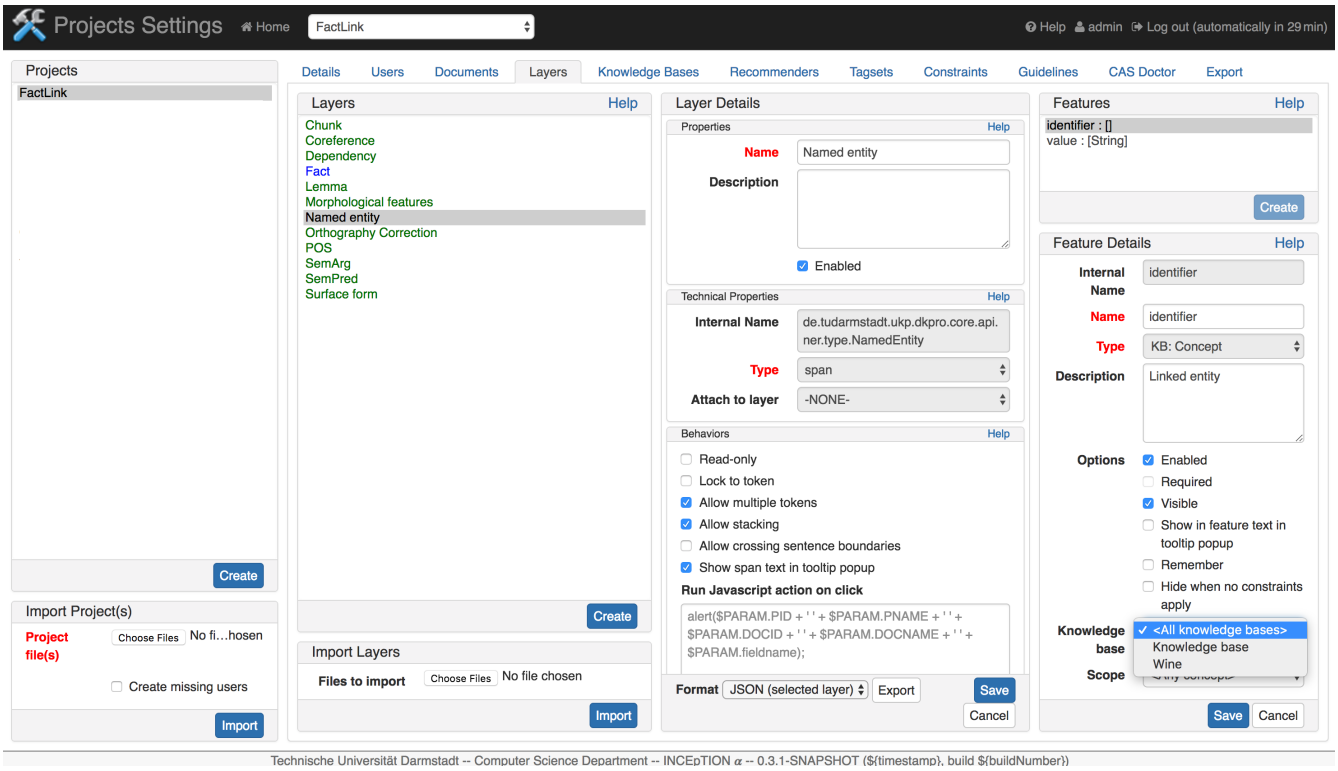


After annotating this qualifer, in the main editor, there is an arrow from the label above the predicate to the label above the qualifier value, with the qualifier name *start-date*.

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.



Images

Linking text to images can be useful e.g. when dealing with OCRed data or with text describing images. To support such cases, INCEpTION supports **image features**. Image features can be to annotation layers just like and other type of features. When selecting an annotation containing an image feature, the a text field is used as the feature editor. Enter an image URL into this field in order to link the annotation to an image. It is presently not possible to upload images to INCEpTION - the image must be accessible via an URL, e.g. from an IIIF server.

Open the **images sidebar** to get an overview over all images linked to any of the annotations currently visible on screen.

The sidebar attempts to add a border of an appropriate color to each image. For light images, a dark border is added and for dark images, a light border is added. However, this is only possible if the image server supports cross-origin resource sharing (CORS). The website enable-cors.org provides tips on how to configure your image server to support CORS. If CORS is not supported by the image server, rendering performance will degrade as there is an attempt to re-load the image without CORS and without trying to determine the border color - but the application should still work and the images should still show.

Curation

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

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.

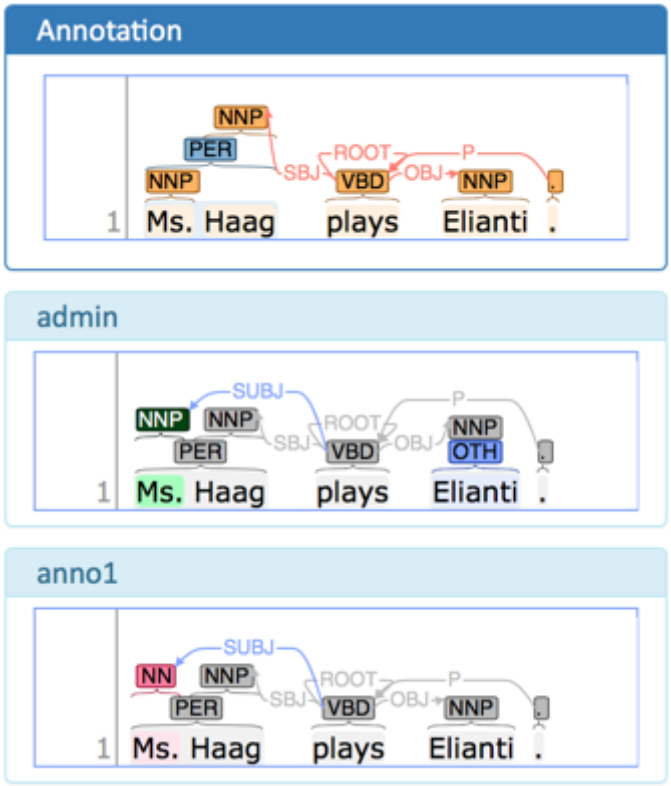
Table 3. Explanation of the project colors in the curation open document dialog

No curatable documents	Red
Curatable documents	Green

Table 4. 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 panels. If the annotators do not agree, the respective annotations are show in dark blue in the lower panels. By default, they are not taken into the merged file (cf. Merging strategies).

Left-click on an annotation in one of the lower panels to merge it. This action copies the annotation to the upper panel. The merged annotation will turn green in the lower panel from which it was selected. If other annotators had a conflicting opinion, these will turn red in the lower panels of the respective annotators.

Right-click on an annotation in the lower panels to bring up a menu with additional options.

- **Merge all XXX**: merge all annotations of the given type from the selected annotator. Note that this overrides any annotations of the type which may previously have been merged or manually created in the upper panel.

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

Table 5. 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

Merging strategies

By default, the merging strategy only considers annotations if all annotators made the same annotation at the same location (i.e. complete and agreeing annotations) - i.e. it considers any annotations not provided by all annotators as a disagreement between the annotators.

However, there are situations where it is desirable to merge annotations from all annotators, even if some did not provide it. For example, if your project has two annotators, one working on POS tagging and another working on lemmatization, then as a curator, you might simply want to merge the annotators from the two. This can be done by using the **Re-merge** action and activating the checkbox **Merge incomplete annotations**. This will re-merge the current document (i.e. discard

the entire state of curation and merge from scratch).

Anonymized curation

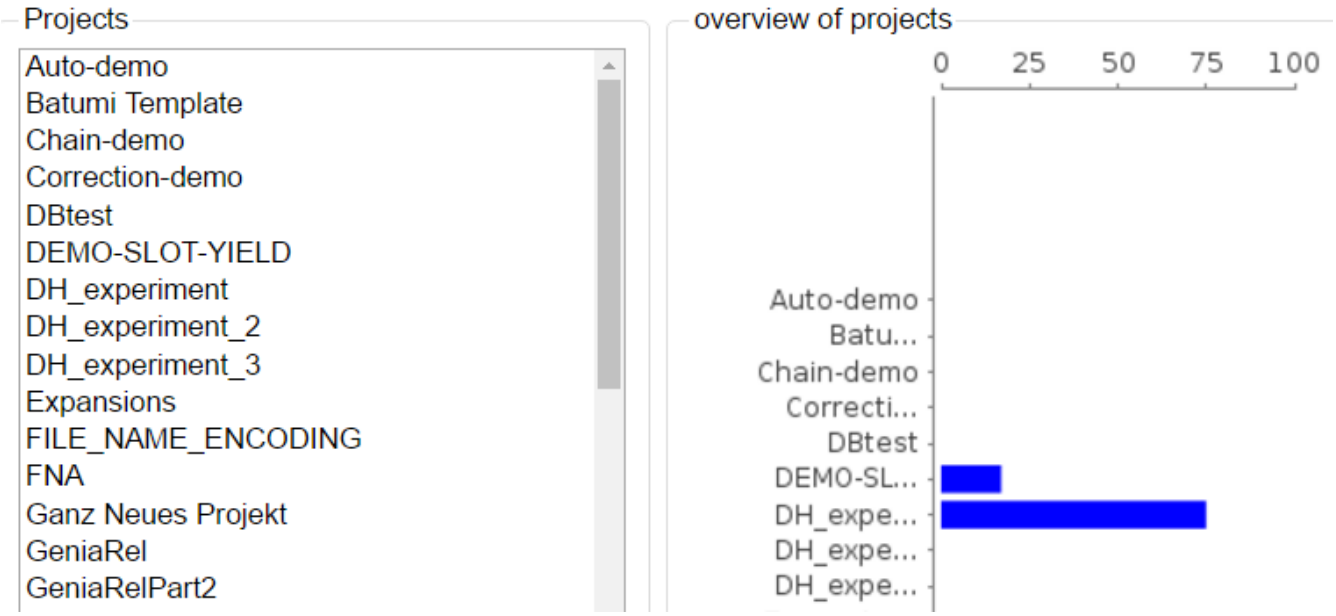
By default, the curator can see the annotators names on the curation page. However, in some cases, it may not be desirable for the curator to see the names. In this case, enable the option **Anonymous curation** in the project detail settings. Users with the curator role will then only see an anonymous label like **Anonymized annotator 1** instead of the annotator names. Users who are project managers can still see the annotator names.

The order of the annotators is not randomized - only the names are removed from the UI. Only annotators who have marked their documents as **finished** are shown. Thus, which annotator recieves which number may changed depending on documents being marked as finished or put back into progress.

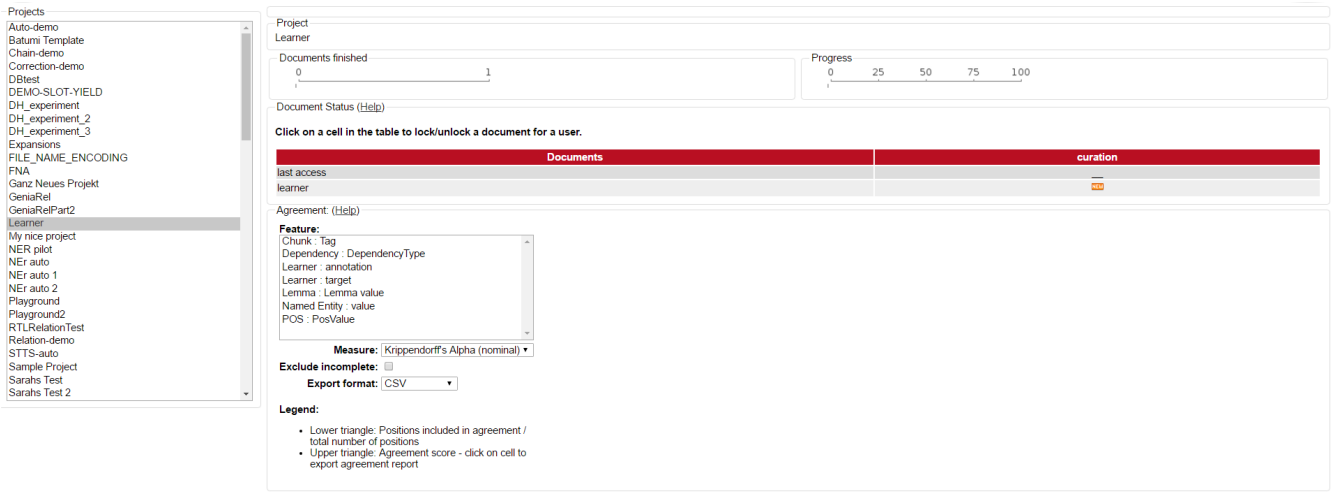
Monitoring

This functionality is only available to **curators** and * managers*.

This page allows 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:

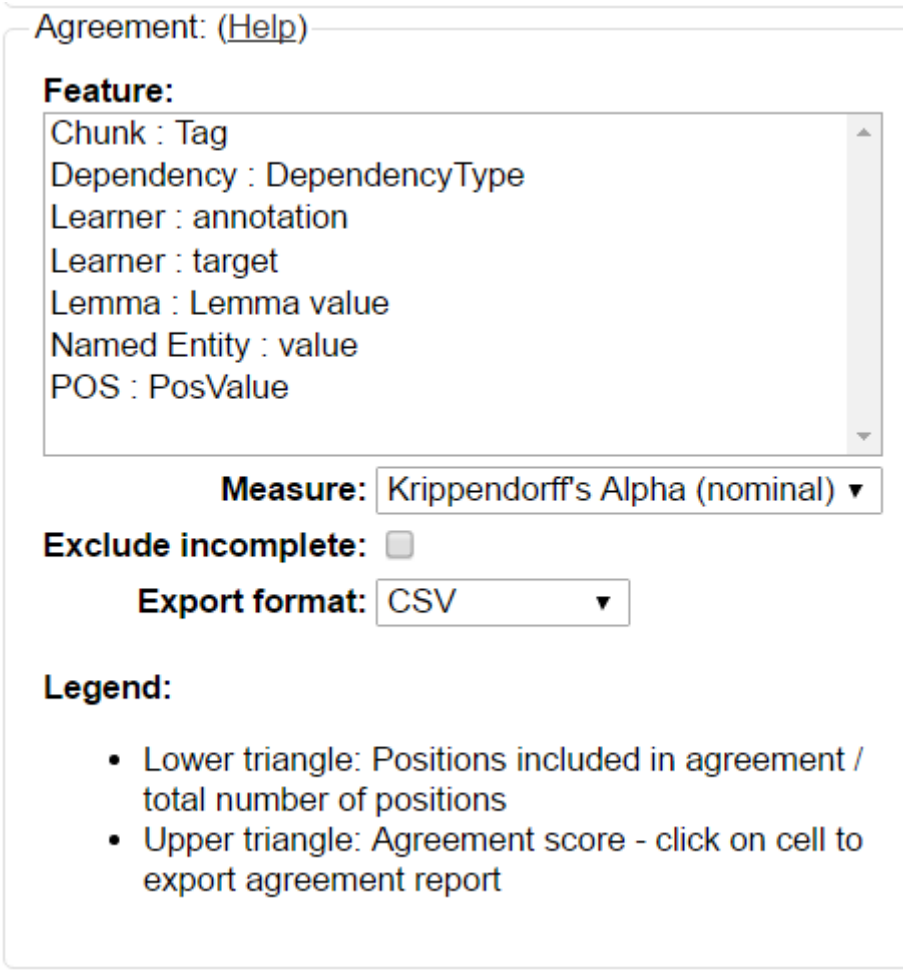


In the right frame, the overall progress of all projects is displayed. In the left frame one sees all projects that you are allowed to curate. By clicking on one of the projects on the left, it may be selected and the following view is opened:



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 6. 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.

users	anno5	anno6	anno7	darina
anno5	1.0	0.74	0.75	0.0
anno6		1.0	0.73	0.0
anno7			1.0	0.0
darina				0.0

Agreement

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

Measures

Several agreement measures are supported.

Table 7. Supported agreement measures

Measure	Type	Short description
Cohen’s kappa	Coding	Chance-corrected inter-annotator agreement for two annotators. The measure assumes a different probability distribution for all raters. Incomplete annotations are always excluded.
Fleiss' kappa	Coding	Generalization of Scott’s pi-measure for calculating a chance-corrected inter-rater agreement for multiple raters, which is known as Fleiss' kappa and Carletta’s K. The measure assumes the same probability distribution for all raters. Incomplete annotations are always excluded.
Krippendorff’s alpha (nominal)	Coding	Chance-corrected inter-rater agreement for multiple raters for nominal categories (i.e. categories are either equal (distance 0) or unequal (distance 1). The basic idea is to divide the estimated variance of within the items by the estimated total variance.

Measure	Type	Short description
Krippendorff's alpha (unitizing)	Unitizing	Chance-corrected inter-rater agreement for unitizing studies with multiple raters. As a model for expected disagreement, all possible unitizations for the given continuum and raters are considered. Note that units coded with the same categories by a single annotator may not overlap with each other.

Coding vs. Unitizing

Coding measures are based on positions. I.e. two annotations are either at the same position or not. If they are, they can be compared - otherwise they cannot be compared. This makes coding measures unsuitable in cases where partital overlap of annotations needs to be considered, e.g. in the case of named entity annotations where it is common that annotators do not agree on the boundaries of the entity. In order to calculate the positions, 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.

Unitizing measures basically work by internally concatenating all documents into a single long virtual document and then consider partial overlaps of annotations from different annotations. I.e. there is no averaging over documents. The partial overlap agreement is calculated based on character positions, not on token positions. So if one annotator annotates **the blackboard** and another annotator just **blackboard**, then the partial overlap is comparatively high because **blackboard** is a longish word. Relation and chain layers are presently not supported by the unitizing measures.

Incomplete annotations

When working with coding measures, there is the concept of **incomplete annotations**. For a given position, the annotation is incomplete if at least one annotator has **not** provided a label. In the case of the pairwise comparisons that are used to generate the agreement table, this means that one annotator has produced a label and the other annotator has not. Due to the way that positions are generated, it also means that if one annotator annotates **the blackboard** and another annotator just **blackboard**, we are actually dealing with two positions (**the blackboard**, offsets 0-15 and **blackboard**, offsets 4-14), and both of them are incompletely annotated. Some measurs cannot deal with incomplete annotations because they require that every annotator has produced an annotation. In these cases, the incomplete annotations are **excluded** from the agreement calculation. The effect is that in the **(the) blackboard** example, there is actually no data to be compared. If we augment that example with some other word on which the annotators agree, then only this word is considered, meaning that we have a perfect agreement despite the annotators not having agreed on **(the) blackboard**. Thus, one should avoid measure that cannot deal with

incomplete annotations such as Fleiss' kappa and Cohen's kappa except for tasks such as part-of-speech tagging where it is known that positions are the same for all annotators and all annotators are required (not expected) to provide an annotation.

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. Thus, an **incomplete** annotation is not one where the label is missing, but rather one where the entire annotation is missing.

In general, it is a good idea to use at least a measure that supports incomplete data (i.e. missing labels) or even a unitizing measure which is able to produce partial agreement scores.

Table 8. 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
null	**empty**	yes	yes
empty	**no annotation**	no	no

Stacked annotations

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.

Pairwise agreement matrix

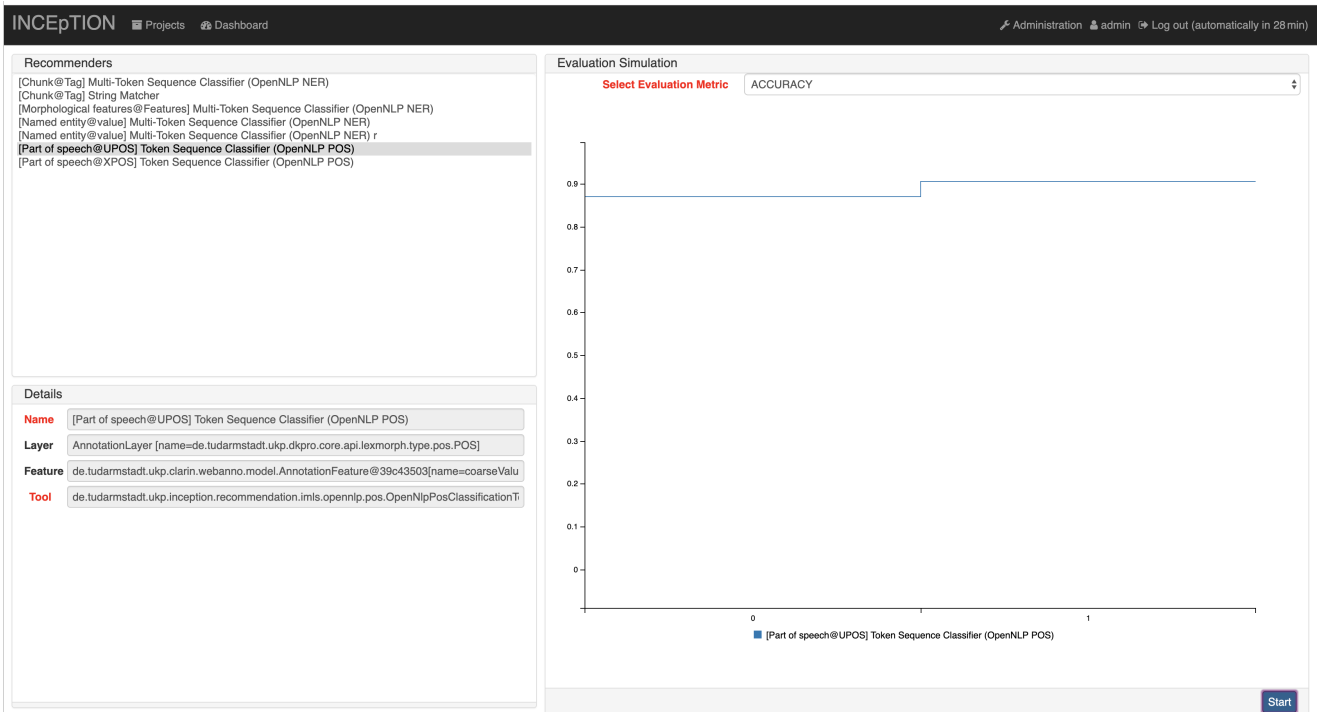
The lower part of the agreement matrix displays how many configuration sets were used to calculate agreement and how many were found in total. The upper part of the agreement matrix displays the pairwise agreement scores.

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.

Evaluation Simulation

The evaluation simulation panel provides a visualization of the performance of the selected recommender with the help of a learning curve diagram. On the bottom right of the panel, the start button performs evaluation on the selected recommender using the annotated documents in the project and plots the evaluation scores against the training data size on the graph. The evaluation score can be one of the four metrics, Accuracy, Precision, Recall and F1. There is a drop down panel to change the metric. The evaluation might take a long time.

For the evaluation, the training and test data is used from all the imported documents of the project. Moreover, the data is split into **80% training data** and **20% test data**. The training data is **incremented by 250** per evaluation.



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.

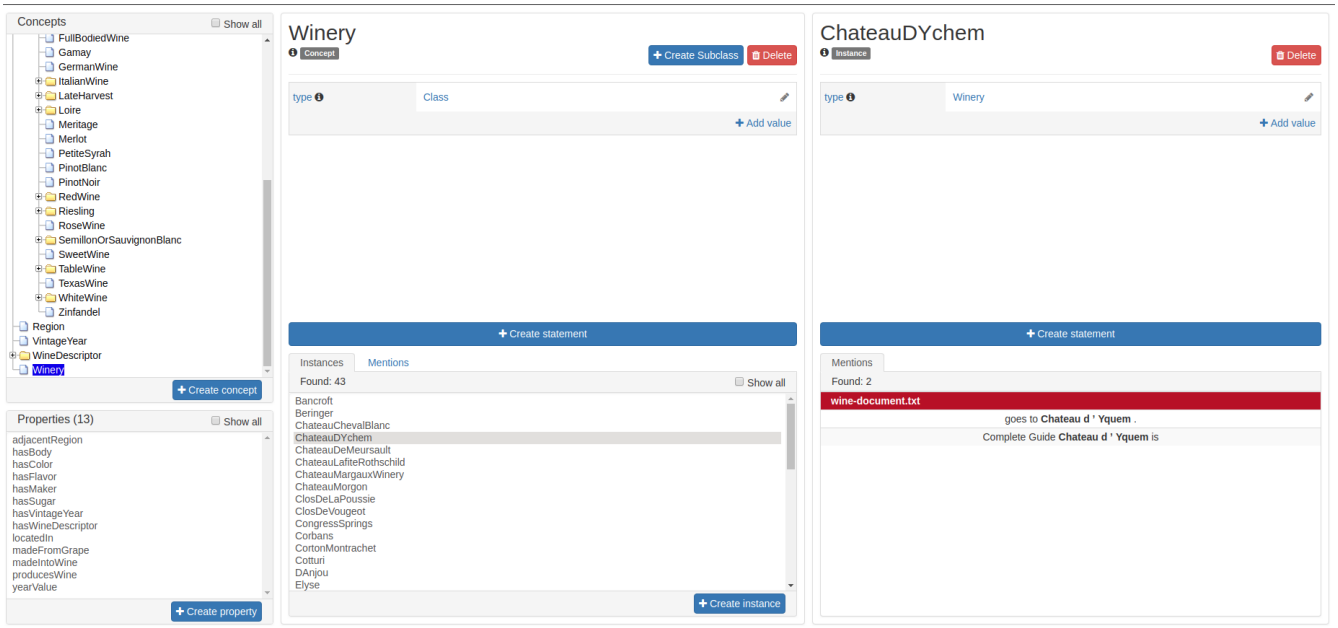
This section briefly describes how to set up a KB in the KB management page on **Projects Settings**, explains the functionalities provided by the **Knowledge Base** page and covers the **concept** and **property** feature types.

In order for a knowledge base to be searchable (e.g. from the Knowledge Base page), the configured knowledge base needs to have labels for all items (e.g. concepts, instances, properties) that should be found.

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.

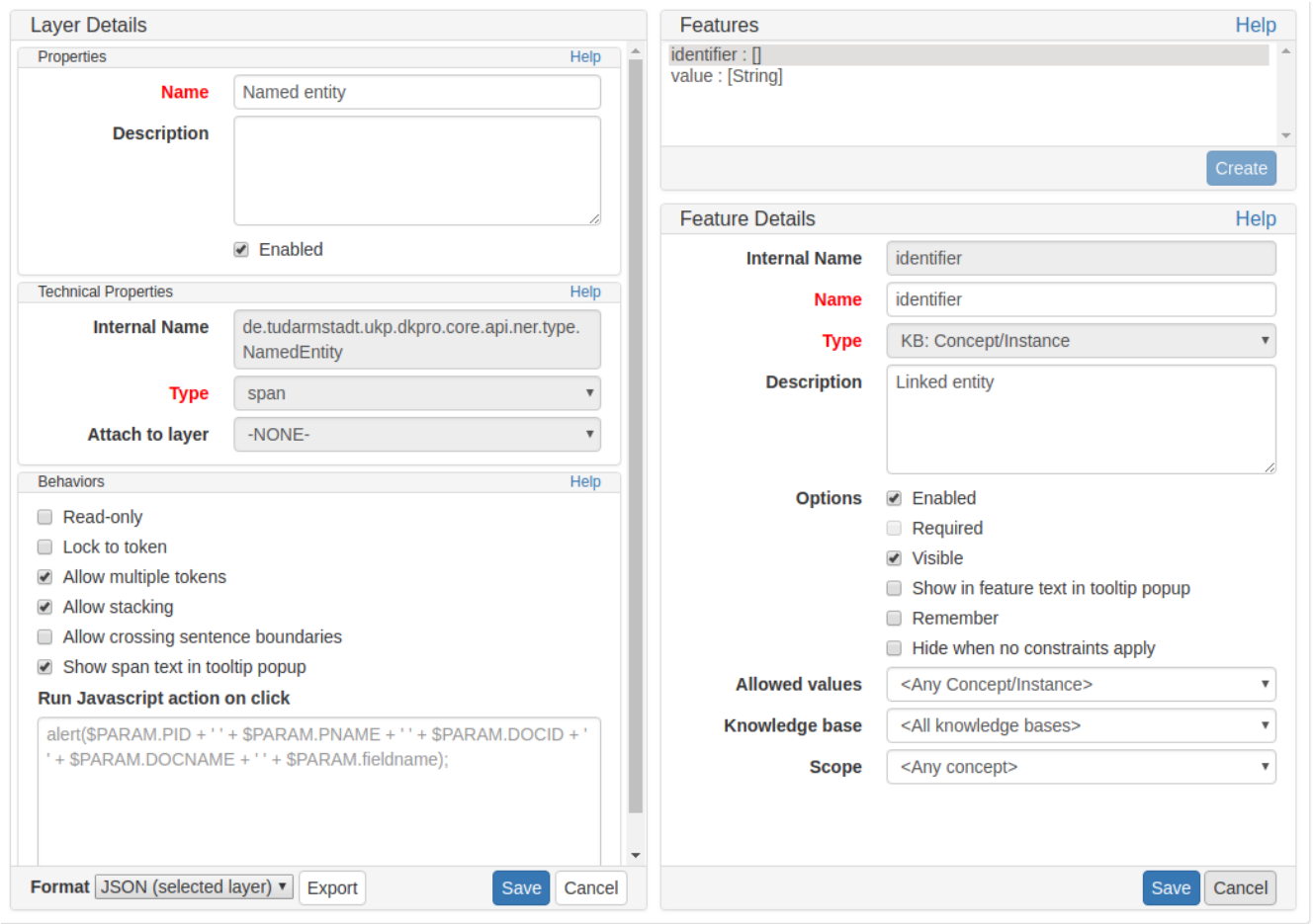
The same is true for the object of a statement: After choosing the property for a concept, the object has to be specified. The possible objects are limited by the range of the property if given. Right now, four different editors are available to specify features for:

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.

The **scope** setting allows to limit linking candidates to a subtree of the knowledge base.

Selecting scope means that full-text search cannot be used. This means that queries may become very slow if the scope covers a large number concepts or instances. Therefore, it is best not to choose too broad scopes.

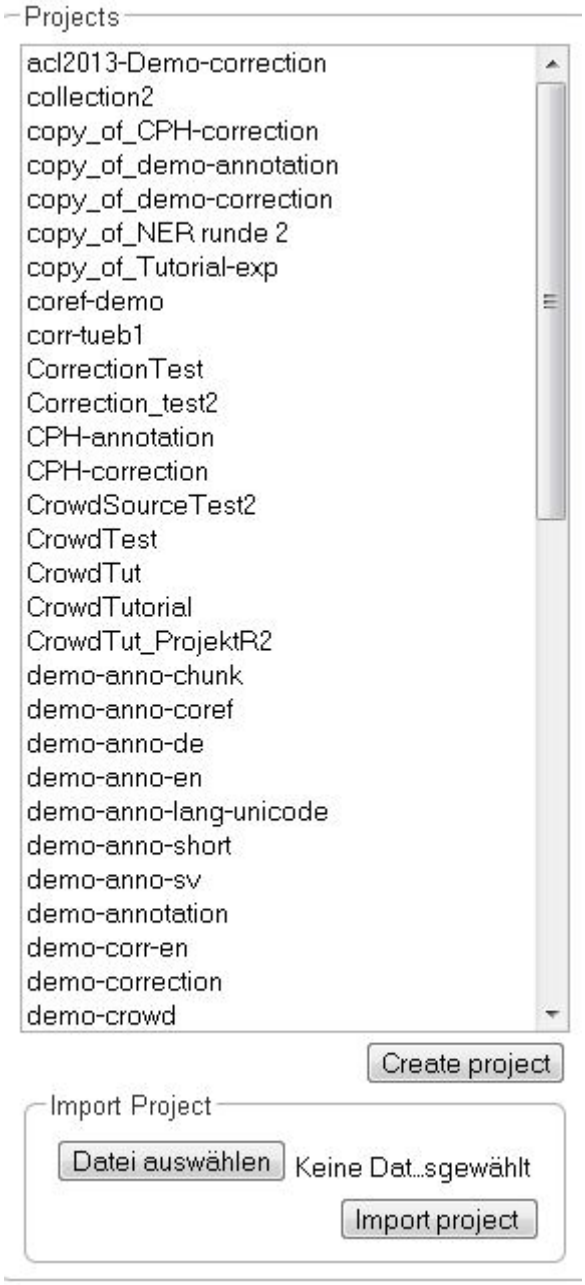
Projects

This functionality is only available to **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].

Click on **Create Project** to create a new project.

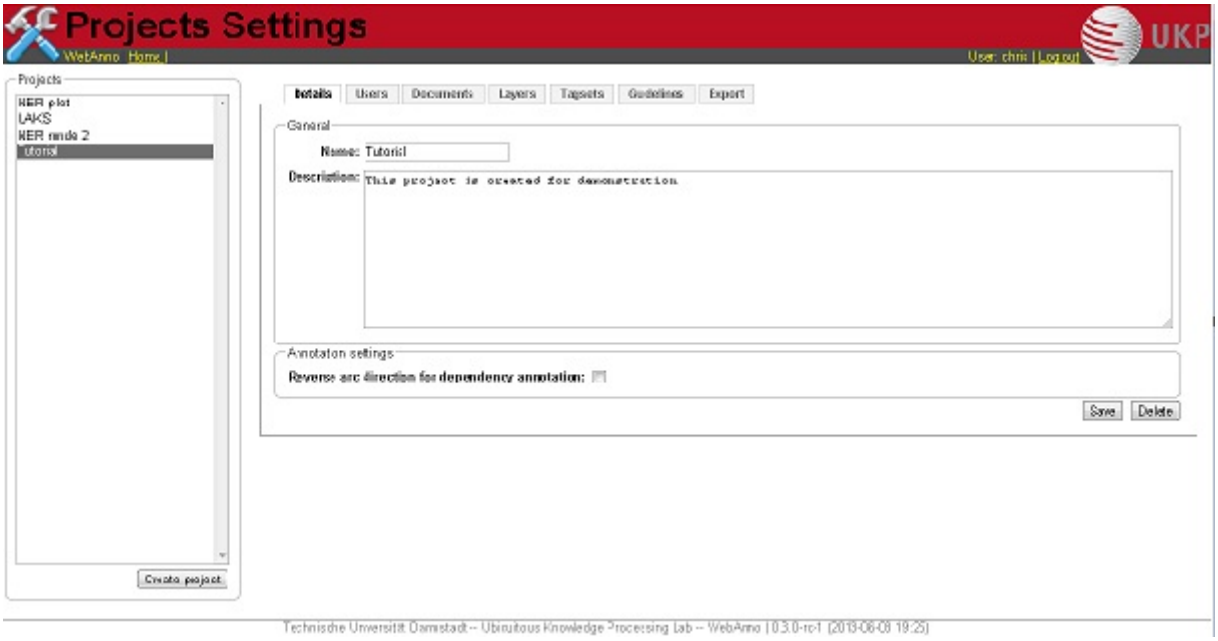


After doing so, a new pane is displayed, where you can name and describe your new project. It is also important to chose the kind of project you want to create. You have the choice between annotation, automation, and correction. Please do not forget to save.

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

Here, you can import project archives such as the example projects provided on our website or projects exported from the Export tab.

When a user with the role **project creator** imports a project, that user automatically becomes a **manager** of the imported project. However, no permissions for the project are imported!

If the current instance has users with the same name as those who originally worked on the import project, the manager can add these users to the project and they can access their annotations. Otherwise, only the imported source documents are accessible.

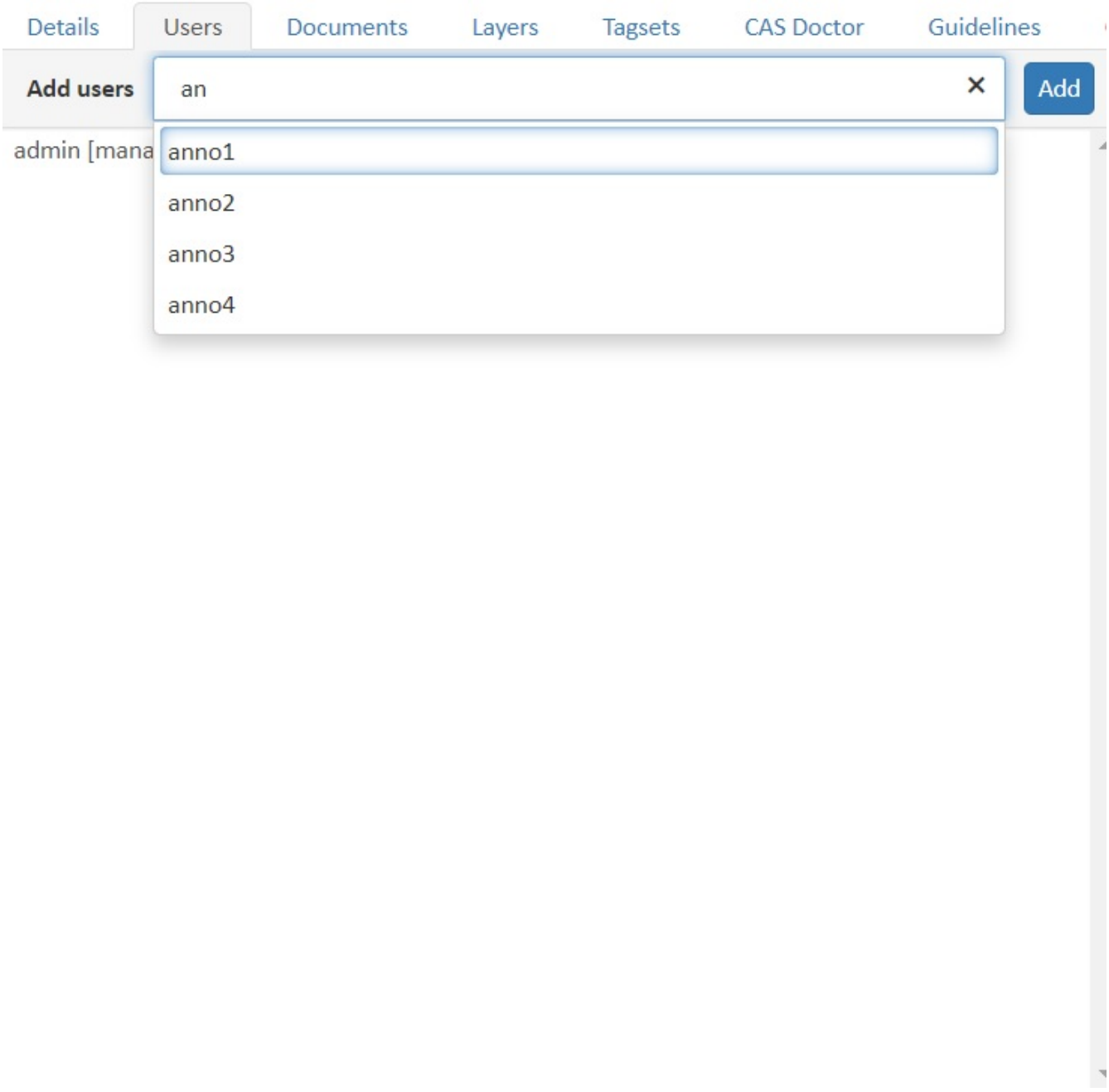
When a user with the role **administrator** imports a project, the user can choose whether to **import the permissions** and whether to **automatically create users** who have permissions on the imported project but so far do not exist. If this option to create missing users disabled, but the option to import permissions is enabled, then 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.

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

Users

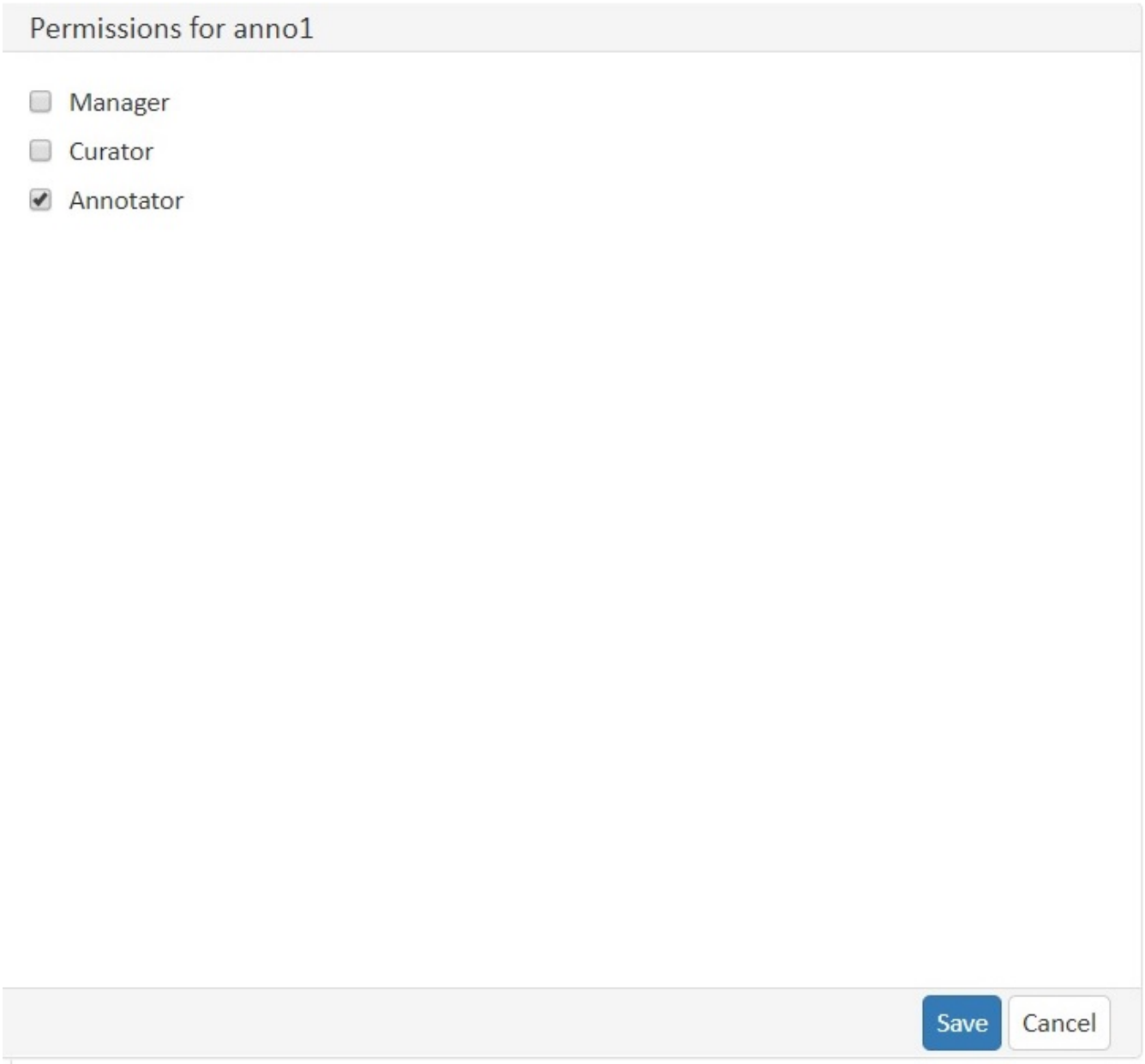
After clicking on the **Users** tab, you are displayed with a new pane in which you can add new users by clicking on the **Add users** text field. You get a dropdown list of enabled users in the system which can be added to the project. Any users which are already part of the project are not offered. As you type the dropdown list with the users is filtered to match your input. By clicking on a

username or by pressing enter you can select the corresponding user. You can keep typing to add more users to the project. When you press the **Add** button the selected users are added to your project.



For privacy reasons, the administrator may choose to restrict the users shown in the dropdown. If this is the case, you have to enter the full name of a user before it appears in the dropdown and can be added.

By default, the users are added to the project as annotators. If you want to assign additional roles, you can do so by clicking on the user and then on **Permissions** pane select the appropriate permissions.



After ticking the wished permissions, click on **Save**. To remove a user, remove all the permissions and then click on **Save**.

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 walk-through 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 9. Built-in layers

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

The coloring of the layers signal the following:

Table 10. 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.

Exporting layers

At times, it is useful to export the configuration of a layer or of all layers, e.g. to copy them to another project. There are two options:

- **JSON (selected layer)**: exports the currently selected layer as JSON. If the layer depends on other layers, these are included as well in the JSON export.
- **UIMA (all layers)**: exports a UIMA type system description containing all layers of the project. This includes built-in types (i.e. DKPro Core types) and it may include additional types required to allow loading the type system description file again. However, this type system description is usually not sufficient to interpret XMI files produced by INCEpTION. Be sure to load XMI files

together with the type system description file which was included in the XMI export.

Both types of files can be imported back into INCEpTION. Note that any built-in types that have have been included in the files are ignored on import.

Properties

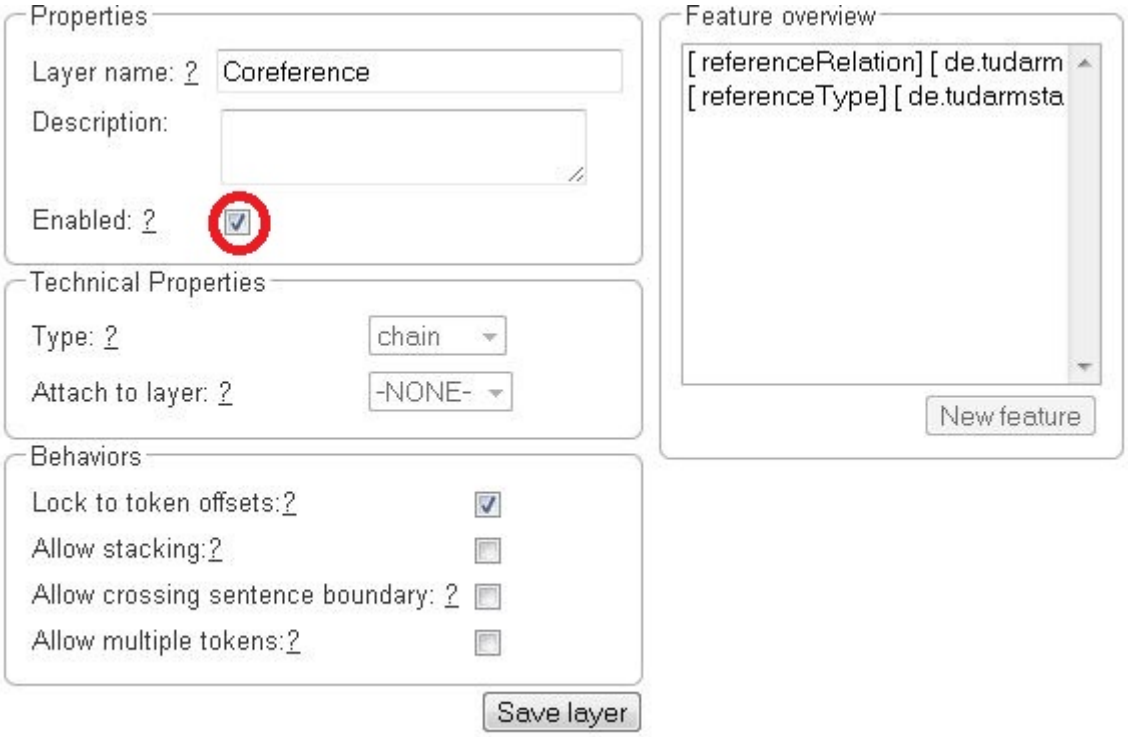
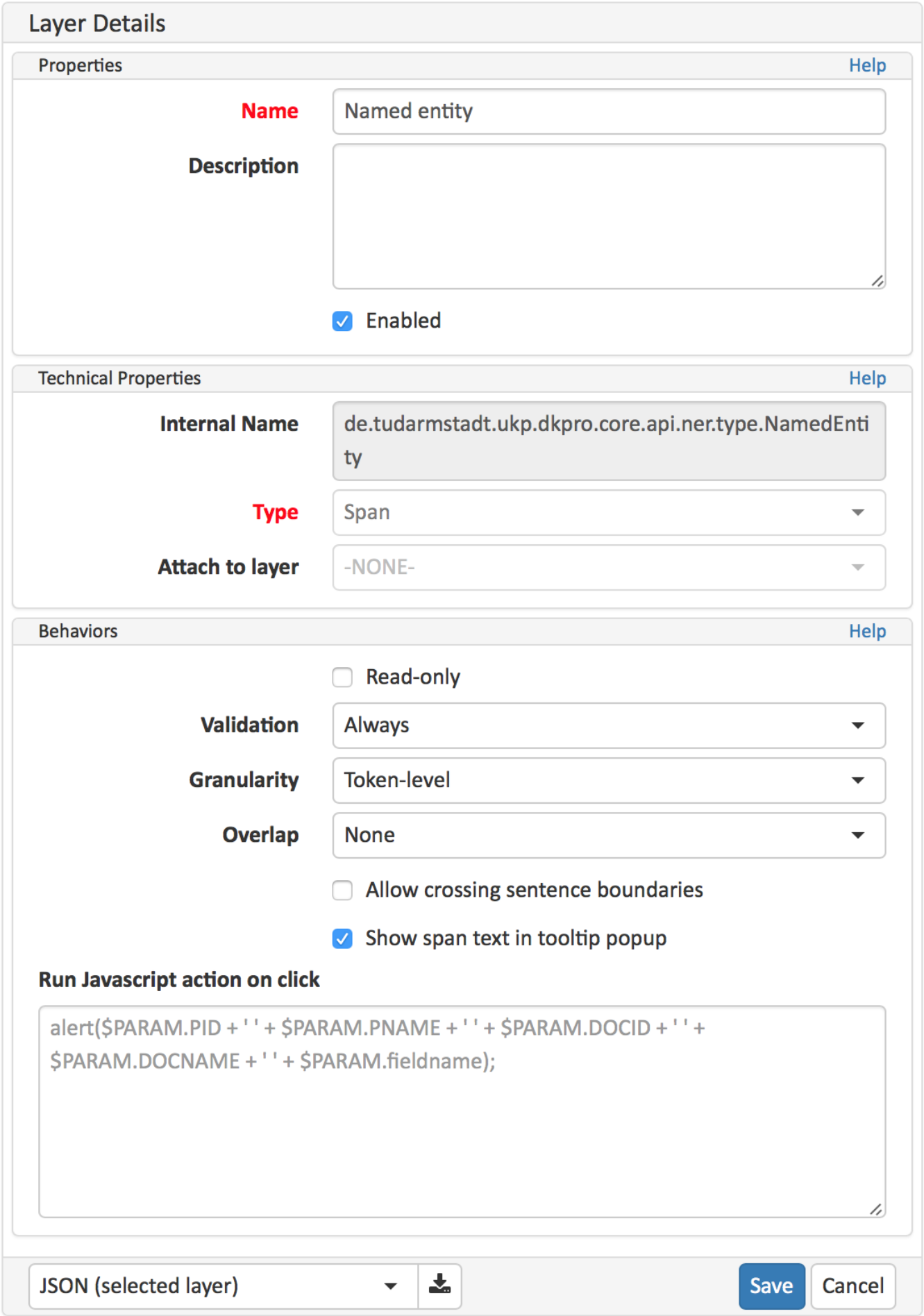


Table 11. Properites

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.


Layer Details
Properties
Help
Name
Named entity
Description
Enabled
Technical Properties
Help
Internal Name
de.tudarmstadt.ukp.dkpro.core.api.ner.type.NamedEntity
Type
Span
Attach to layer
-NONE-
Behaviors
Help
Read-only
Validation
Always
Granularity
Token-level
Overlap
None
Allow crossing sentence boundaries
Show span text in tooltip popup
Run Javascript action on click
alert($PARAM.PID + ' ' + $PARAM.PNAME + ' ' + $PARAM.DOCID + ' ' + $PARAM.DOCNAME + ' ' + $PARAM.fieldname);
JSON (selected layer)
Save
Cancel

Technical Properties

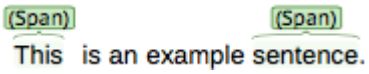
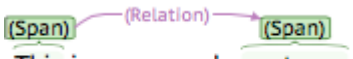
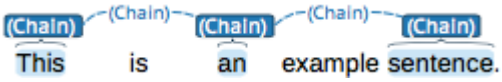
In the frame **Technical Properties**, the user may select the type of annation that will be made with this layer: span, relation, or chain.

Table 12. 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 13. 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.

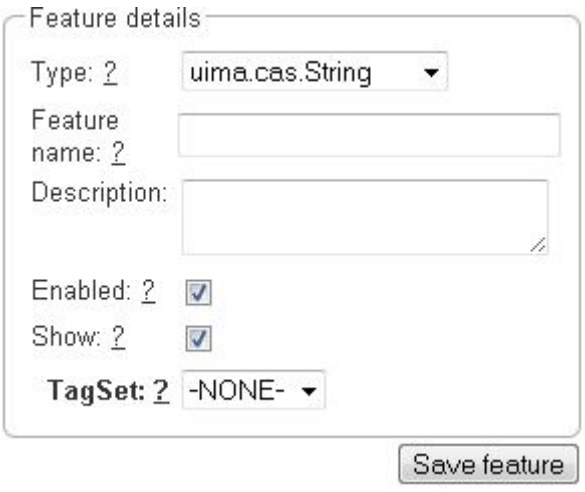
Behaviours

Table 14. Behaviors

Behavior	Description
Read-only	The layer may be viewed but not edited.
Validation	When pre-annotated data is imported or when the behaviors settings are changed, it is possible that annotations exist which are not conforming to the current behavior settings. This setting controls when a validation of annotations is performed. Possible settings are **Never** (no validation when a user marks a document as finished) and **Always** (validation is performed when a user marks a document as finished). Mind that changing the document state via the Monitoring page does not trigger a validation. Also, problematic annotations are highlighted using an error marker in the annotation interface. **NOTE:** the default setting for new projects/layers is **Always**, but for any existing projects or for projects imported from versions of INCEpTION where this setting did not exist yet, the setting is initialized with **Never**.
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.

Behavior	Description
Overlap	This setting controls if and how annotations may overlap. For **span layers**, overlap is defined in terms of the span offsets. If any character offset that is part of span A is also part of span B, then they are considered to be **overlapping**. If two spans have exactly the same offsets, then they are considered to be **stacking**. For **relation layers**, overlap is defined in terms of the end points of the relation. If two relations share any end point (source or target), they are considered to be **overlapping**. If two relations have exactly the same end points, they are considered to be **stacking**. Note that some export formats are unable to deal with stacked or overlapping annotations. E.g. the CoNLL formats cannot deal with overlapping or stacked named entities.
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 15. 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 16. 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.

Table 17. String feature properties

Property	Description
Multiple Rows	If enabled the textfield will be replaced by a textarea which expands on focus. This also enables options to set the size of the textarea and disables tagsets.
Collapsed Rows	Set the number of rows for the textarea when it is collapsed and not focused.

Property	Description
Expanded Rows	Set the number of rows for the textarea when it is expanded and not focused.

Table 18. Number feature properties

Property	Description
Limited	If enabled a minimum and maximum value can be set for the number feature.
Minimum	Only visible if `Limited` is enabled. Determines the minimum value of the limited number feature.
Maximum	Only visible if `Limited` is enabled. Determines the maximum value of the limited number feature.
Editor Type	Select which editor should be used for modifying this features value.

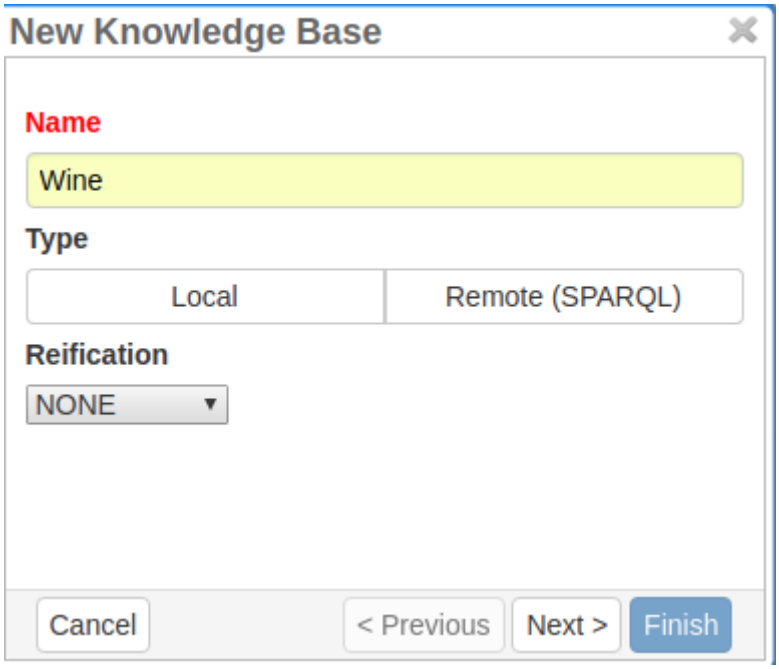
Table 19. Link feature properties

Property	Description
Enable Role Labels	Allows users to add a role label to each slot when linking anntations. If disabled the UI labels of annotations will be displayed instead of role labels. This property is enabled by default.

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

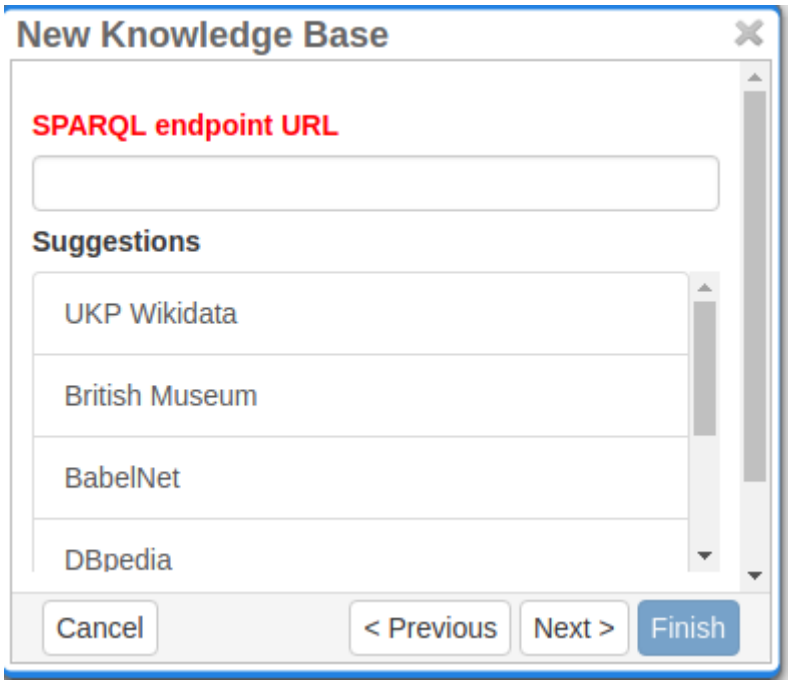
In the **Projects Settings**, switch to the **Knowledge Bases** tab, then click **New...** on the bottom and a dialogue box shows as in the figure below.



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**.

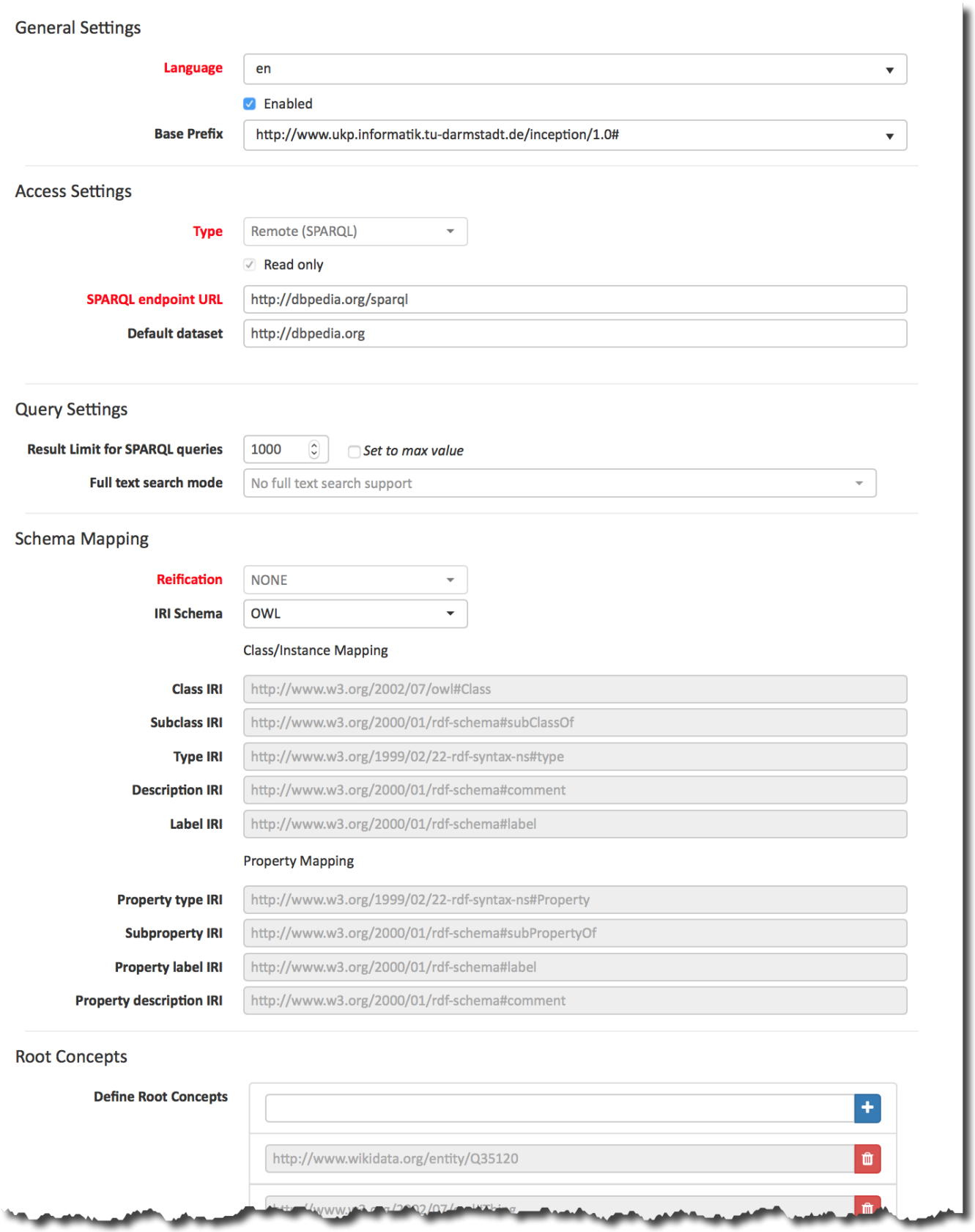
For the local KB, the user can optionally choose a RDF file from which to import the initial data. Alternatively, the user can skip the step to create an empty KB to create a knowledge base from scratch. It is also always possible to import data from an RDF file after the creation of a KB. It is also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



Settings

There are various settings for knowledge bases.



Remote KBs

The remote knowledge bases, there are the following settings:

- **SPARQL endpoint URL:** The SPARQL URL used to access the knowledge base
- **Default dataset:** A SPARQL endpoint may server multiple datasets. This setting can be used to restrict queries to a specific one. Consult with the operator of the SPARQL server to see which datasets are available.
- **Result limit for SPARQL queries:** this is used to limit the amount of data retrieved from the remote server, e.g when populating dropdown boxes for concept search.
- **Full text search mode:** a SPARQL endpoint capable of full-text searching makes the knowledge based considerably more responsive. Use `bif:contains` for Virtuoso servers and `lucenesail` for RDF4J-based servers. Note that full-text search must be enabled on the server by the server operator.

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:

- `bif:contains`: use with remote Virtuoso SPARQL endpoints.
- `text:query`: use with remote Fuseki SPARQL endpoints.
- `https://www.mediawiki.org/ontology#API/`: use with the official Wikidata SPARQL endpoint.
- `lucenesail#matches`: use with local knowledge bases or possibly with remote knowledge bases using the RDF4J Lucene SAIL.
- `FTS:NONE`: use if there is not full text search support in the triple store - this will be very slow in particular for large KBs.

Importing RDF

KBs can be populated by importing RDF files. Several formats are supported. The type of the file is determined by the file extension. So make sure the files have the correct extension when you import them, otherwise nothing might be imported from them despite a potentially long waiting time. The application suppports GZIP compressed files (ending in `.gz`, so e.g. `.ttl.gz`), so we recommend compressing the files before uploading them as this can significantly improve the import time due to a reduced transfer time across the network.

Format	Extension
RDF	`.rdf`

Format	Extension
RDF Schema	`.rdfs`
OWL	`.owl`
N-Triples	`.nt`
Turtle	`.ttl`

Recommenders

Recommenders provide annotation support by predicting potential labels. These can be either accepted or rejected by the user. A recommender learns from this interaction to further improve the quality of its predictions.

Recommenders are trained every time an annotation is created, updated or deleted. In order to determine whether the annotations are good enough, recommenders are evaluated on the annotation data. During recommender evaluation a score for each recommender is calcultated and if this score does not meet the configured threshold, the recommender will not be used.

Recommenders can be configured in the **Project Settings** under the **Recommenders** tab. To create a new recommender, click **Create**. Then, the layer, feature and the classifier type has to be selected.

By default, the name of new recommenders are auto-generated based on the choice of layer, feature and tool. However, you can deactivate this behavior by unchecking the **auto-generate** option next to the name field.

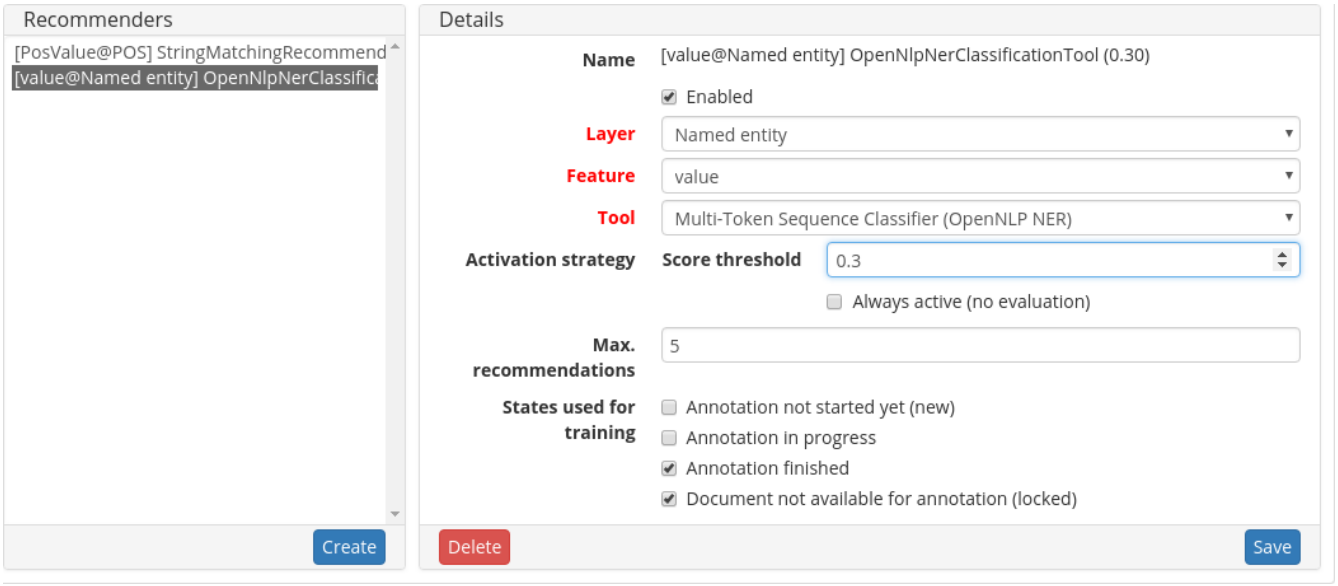
Recommenders can be enabled and disabled. This behaviour is configured by the **Enabled** checkbox. Recommenders that are disabled are not used for training and prediction and are not evaluated.

The **Activation strategy** describes when a recommender should be used for prediction. Right now, there are two options: either set a threshold on the evaluation score (if the evaluation score is lower than the threshold, the recommender is not used for predicting until annotations have changed) or always enable it. If the option **Always active** is disabled and the score threshold is set to 0, the recommender will also be always executed, but internally it is still evaluated.

Some recommenders are capable of generating multiple alternative suggestions per token or span. The maximum number of suggestions can be configured by the **Max. recommendations** field.

Sometimes it is desirable to not train on all documents, but only on e.g. finished documents. In order to control documents in which state should be used for training, the respective ones can be selected from the **States used for training**.

To save a recommender, click **Save**. To abort, click **Cancel**. To edit an existing recommender, it can be selected from the left pane, edited and then saved. Recommenders can be deleted by clicking on **Delete**. This also removes all predictions by this recommender.



String Matcher

The string matching recommender is able to provide a very high accuracy for tasks such as named entity identification where a word or phrase always receives the same label. If an annotation is made, then the string matching recommender projects the label to all other identical spans, therefore making it easier to annotate repeated phenomena. So if we annotate *Kermit* once as a *PER*, then it will suggest that any other mentions of *Kermit* should also be annotated as *PER*. When the same word or phrase is observed with different labels, then the matcher will assign the relative frequency of the observations as the score for each label. Thus, if *Kermit* is annotated twice as *PER* and once as *OTH*, than the score for *PER* is 0.66 and the score for *OTH* is 0.33.

The recommender can be used for span layers that anchor to single or multiple tokens and where cross-sentence annotations are not allowed. It can be used for string features or features which get internally represented as strings (e.g. concept features).

Gazeteers

It is possible to pre-load gazeteers into string matching recommenders. A gazeteer is a simple text file where each line consists of a text and a label separated by a tab character. The order of items in the gazeteer does not matter. Suggestions are generated considering the longest match. Comment lines start with a #. Empty lines are ignored.

Gazeteer example

```
# This is a comment
Obama   PER
Barack Obama    PER
Illinois    LOC
Illinois State Senate   ORG
Hawaii  LOC
Indonesia   LOC
```

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, it considers only the first overlapping annotation and then skips all annotation until it reaches one that does not overlap with it.

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.

LAPPS Grid

This recommender allows to use LAPPS Grid web-services to generate predictions. Only services taking the inputs are supported:

Inputs

- Text
- Tokens
- Sentences

Furthermore, the services must produce annotations on one of the following built-in layers:

Outputs

- Part-of-speech
- Lemma
- Named entity

The recommender comes with a list of pre-configured services that can be conveniently chosen from. However, note that these services use versioned URLs and as LAPPS Grid evolves, these the services might be upgrade and become available under different URLs.

WebLicht

The WebLicht recommender allows you to use CLARIN WebLicht services to generate annotation recommendations. In order to do so, first need to obtain an API key here.

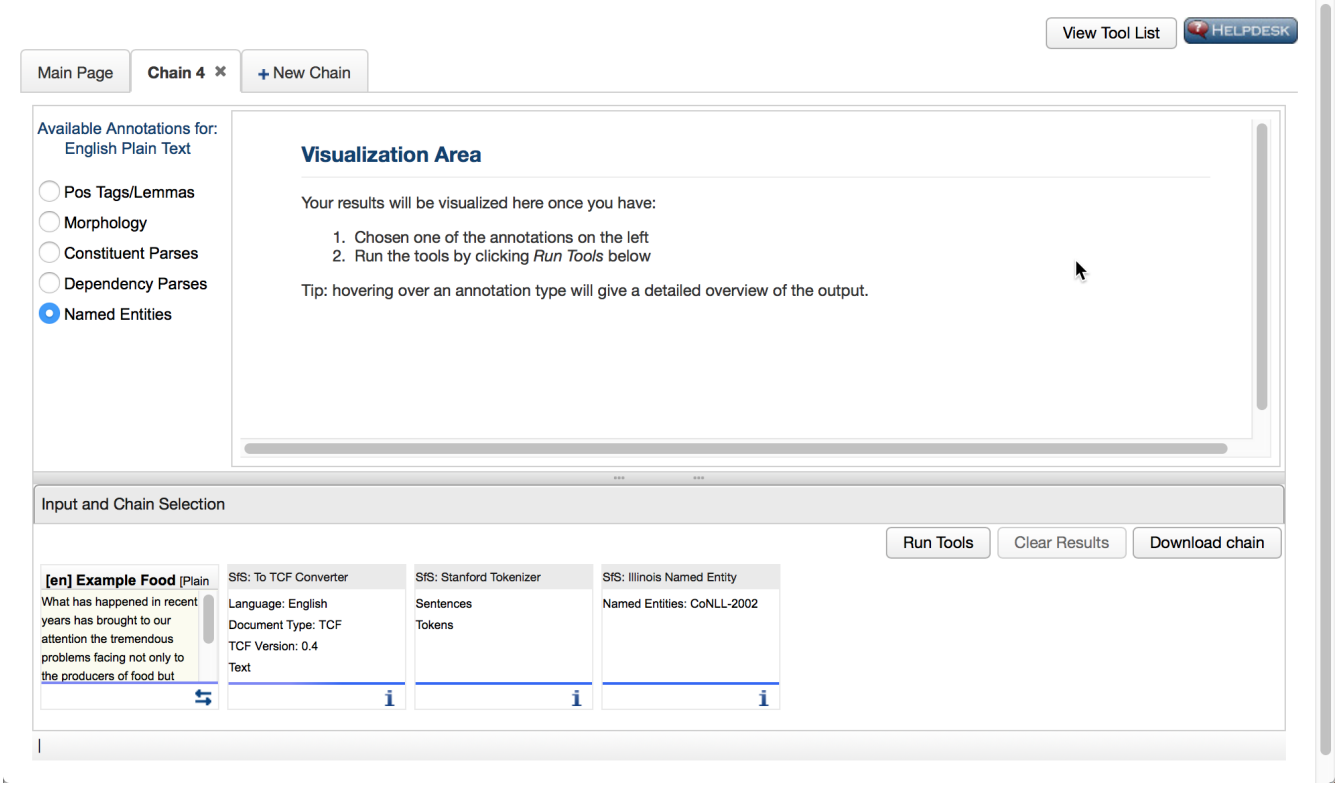
After making the basic settings and entering the API key, **Save** the recommender. Doing so allows you to attach a processing chain definition file. With out such a file, the recommender will not work. We will provide some example settings here to go along with the example processing chain that we will be building below:

- **Layer**: Named entity
- **Feature**: value
- **Tool**: WebLicht recommender
- **URL**: Do not change the default value unless you really know what you are doing (e.g. developing custom WebLicht services).
- **Input format:** Plain text

Next, log in to WebLicht to build a processing chain.

The simplest way to build a chain is this:

- **Choose a sample input.** Make sure the language of the input matches the language of the documents in your INCEpTION project. WebLicht will only allow you to add NLP services to the chain which are compatible with that language. For our example, we will choose `[en] Example Food`. Press **OK**.
- **Choose easy mode.** This allows you to conveniently select a few common types of annotations to generate.
- **Choose Named Entities.** For our example, we choose to generate named entity annotations, so we select this from the list on the left.
- **Download chain**. Click this button to download the chain definition. Once downloaded, it is a good idea to rename the file to something less cryptic than the default auto-generated ID, e.g. we might rename the file to `WebLicht_Named_Entities_English.xml`.



Back in the recommender settings, click **Browse** in the **Upload chain** field and select the processing chain definition file you have just generated. Then click the **Upload** button that appears in the field.

For good measure, **Save** the whole settings once more. When you now open a document in the annotation page, you should be able to see recommendations.

Supported annotations

The WebLicht recommender can currently be used with the following built-in annotation layers:

- Part of speech
- Lemma
- Named entities (only the `value` feature)

Using the TCF format

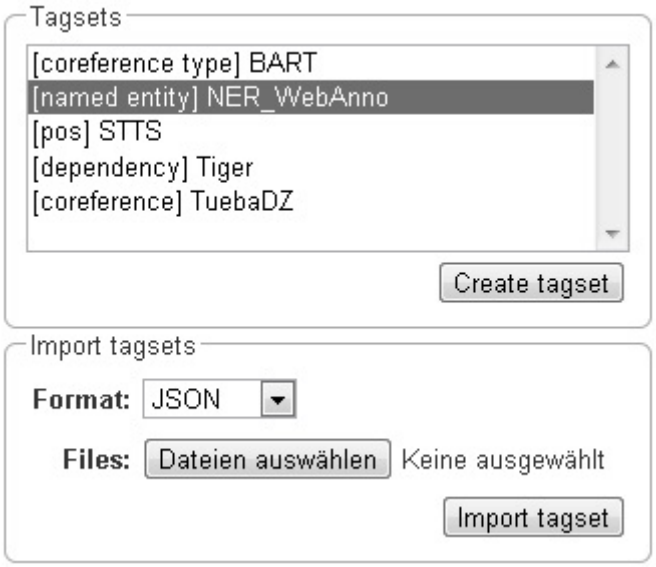
By default, the recommender sends data as plain text to WebLicht. This means that the processing chain needs to run a tokenizer and sentence splitter. Since these might generate boundaries different from the one you have in INCEpTION, some of the recommendations might look odd or may not be displayed at all. This can be avoided by sending data in the WebLicht TCF format. If you select this format, the tokens and sentence boundaries will be sent to WebLicht along with the text. You will then also need to specify the language of the documents that you are going to be sending. Note that even when selecting the TCF format, only text, language, tokens and sentences are sent along - no other annotations. Also, only the target layer and feature will be extracted from the processing chain's results - no other annotations.

However, building a processing chain that takes TCF as input is a bit more difficult. When building the chain, you need to upload some TCF file containing tokens, sentences, and the proper language in the **Input selection** dialog of WebLicht. One way to get such a file is to open one of your documents in the annotation page, export it in the TCF format, then opening the exported file in a

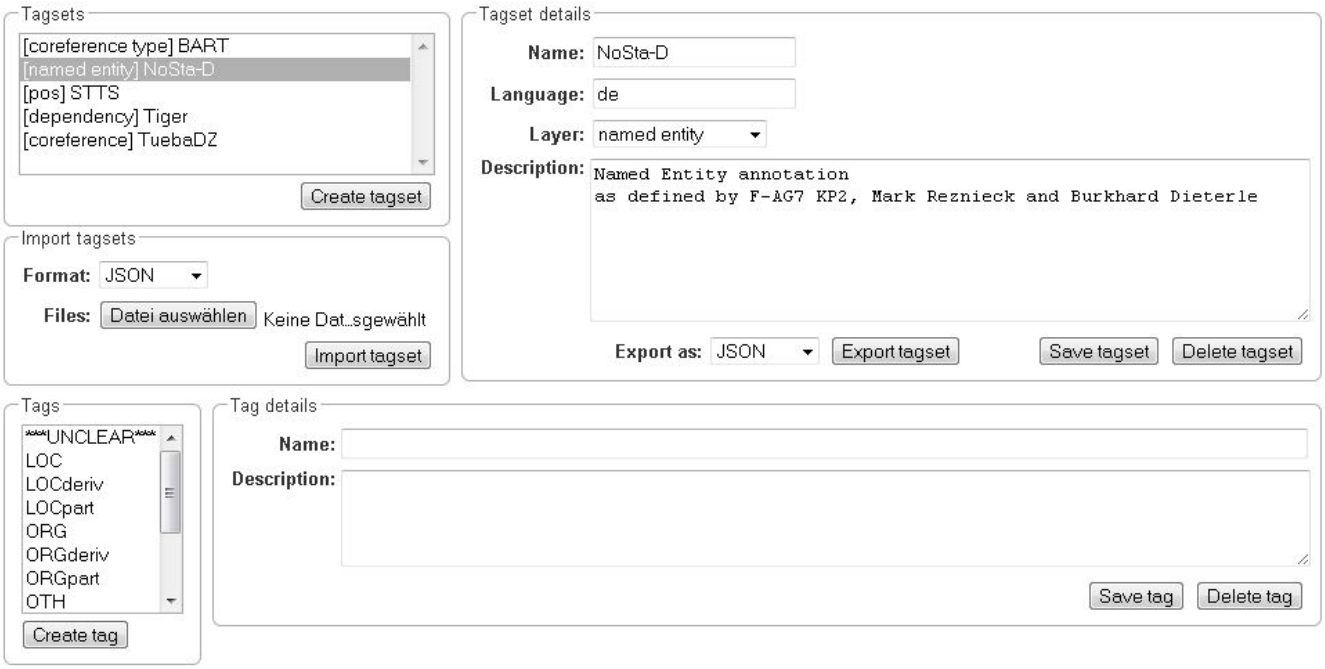
text editor an manually fixing the `lang` attribute on the `tc:TextCorpus` XML element. We know that this is a bit inconvenient and try to come up with a better solution.

Tagsets

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

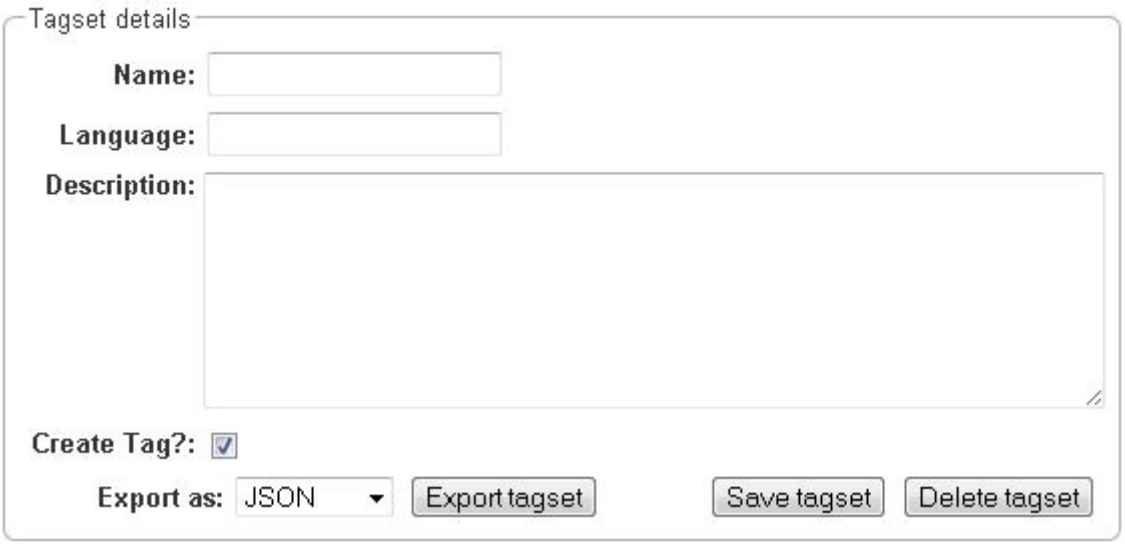


To edit one of the existing tagsets, select it by a click. Then, the tagset characteristics are displayed.



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**.

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. A re-import of these archives is not possible.

A **whole project** export always serves as an archive which can be re-imported again since it includes the annotations in the format internally used by the application. In addition to the internal format, the annotations can be included in a secondary format in the export. This format is controlled by 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.

The **AUTO** format export annotated files in the format of the originally imported file. If the original file format did not contain any annotations (e.g. plain text files) or only specific types of annotations (e.g. CoNLL files), the secondary annotation files will also have none or limited annotations.

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

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**.

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

The `CORRECTION_USER.XXX` and `CURATION_USER.ser` may be located in the `curation` and `curation_ser` folders for old exported projects.

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.

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.

Advanced functionalities

Corpus building

In order to annotate text, it is first necessary to actually have text documents. Not every text documented is worth annotating. For this reason, INCEpTION allows connecting to external document repositories, to search these repositories for interesting documents, and to import relevant documents.

Search page

If document repositories have been configured in a project, the **Search** page becomes accessible through the project dashboard. On the top left of the search page you can select in a dropdown menu which document repository you want to query. All document repositories that were created in the project settings should be selectable here and are identified by their **Name**. The field next to it is the query text field in which the search queries are entered. After entering a query, search by pressing the **Enter** key or by a clicking on the **Search** button. The documents in the document repository which match the search query are returned as the search results and then shown as a table. The table displays 10 results at a time and more can be accessed through the paging controls which are located above the table. Depending on the repository, you may see a document title or ID, text snippets with highlights indicating matches of your query in the document, and a score which represents the relevance of the document to the query. If a document has not yet been imported into your project, there is an **Import** button which extracts the document from the repository and adds it to the project, thereby making it available for annotation. If the document has already been imported, there is an **Open** button instead. Clicking on the document title or ID opens a preview page where the document text can be viewed before importing it.

Normally the ability to add new documents to a project is limited to project managers and it is only possibly via the **Documents** tab in the project settings. However, any user can import a document from an external repository.

External search sidebar

The external search functionality can be used in the sidebar of the annotation page as well and can be opened by clicking on the globe-logo on the sidebar at the left of the annotation page. It essentially offers the same functionality as the external search page accessible via the project dashboard. Being able to search directly from the annotation page may be more convenient though because the user does not have to keep switching between the search page and the annotation page. Additionally, clicking on a search result in the external search sidebar automatically imports the document into your project and opens it in the annotation view.

Document repositories

Document repositories can be added via the **Document repository** tab in the project settings.

ElasticSearch

Selecting the **ElasticSearch** repository type allows connecting to remote ElasticSearch instances.

In order to set up a connection to an ElasticSearch repository, the following information needs to be provided:

- **Remote URL**: the URL where the ElasticSearch instance is running (e.g. `http://localhost:9200/`)
- **Index Name**: the name of the index within the instance (e.g. `mycorpus`)
- **Search path**: the suffix used to access the searching endpoint (usually `_search`)
- **Object type**: the endpoint used to download the full document text (usually `texts`)
- **Field**: the field of the documents in the ElasticSearch repository that is used for matching the search query (default `doc.text`)

From this information, two URLs are constructed:

- the search URL: `<URL>/<index name>/<search path>`
- the document retrieval URL as: `<URL>/<index name>/<object type>/<document id>`

The individual documents should contain following two fields as their source:

- **doc**: should contain the subfield *text* which is the full text of the document
- **metadata**: should contain subfields like *language*, *source*, *timestamp* and *uri* to provide further information about the document

The **Random Ordering** setting allows to switch the ranking of results from the default ranking used by the ElasticSearch server to a random order. The documents returned will still match the query, but the order does not correspond to the matching quality anymore. When random ordering is enabled, no score is associated with the search results. If desired, the **random seed** used for the ordering can be customized.

The **Result Size** setting allows to specify the number of document results that should be retrieved when querying the document repository. The possible result sizes lie between 1 and 10000 documents.

If the default **Field** setting `doc.text` is used, then the JSON structure for indexed documents should look as follows:

```
{
  "doc": {
    "text": "Here goes the document text."
  }
}
```

PubAnnotation

PubAnnotation is a repository through which anyone can share their texts and annotations with others. It can be added as an external document repository by selecting the **PubAnnotation** repository type.

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.

Importing 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.

Implementing constraint sets

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.

All identifiers used in constraint statements are **case sensitive**.

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. Conditions can be defined on features with string, integer or boolean values, but in any case, the value needs to be put into quotes (e.g. `someBooleanFeature="true"`, `someIntegerFeature="2"`).

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. **Restrictions can only be defined on String-valued features that are associated with a tagset.**

```
<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
```

CAS Doctor

The CAS Doctor is an essential development tool. When enabled, it checks the CAS for consistency when loading or saving a CAS. It can also automatically repair inconsistencies when configured to do so. This section gives an overview of the available checks and repairs.

It is safe to enable any checks. However, active checks may considerably slow down the application, in particular for large documents or for actions that work with many documents, e.g. curation or the calculation of agreement. Thus, checks should not be enabled on a production system unless the application behaves strangely and it is necessary to check the documents for consistency.

Enabling repairs should be done with great care as most repairs are performing destructive actions. Repairs should never be enabled on a production system. The repairs are executed in the order in which they are appear in the `debug.casDoctor.repairs` setting. This is important in particular when applying destructive repairs.

When documents are loaded, CAS Doctor first tries to apply any enabled repairs and afterwards applies enabled checks to ensure that the potentially repaired document is consistent.

Additionally, CAS Doctor applies enabled checks **before** saving a document. This ensures that a bug in the user interface introduces inconsistencies into the document on disk. I.e. the consistency of the persisted document is protected! Of course, it requires that relevant checks have been implemented and are actually enabled.

By default, CAS Doctor generates an exception when a check or repair fails. This ensures that inconsistencies are contained and do not propagate further. In some cases, e.g. when it is known that by its nature an inconsistency does not propagate and can be avoided by the user, it may be convenient to allow the user to continue working with the application while a repair is being developed. In such a case, CAS Doctor can be configured to be non-fatal. Mind that users can always continue to work on documents that are consistent. CAS Doctor only prevents loading inconsistent documents and saving inconsistent documents.

Configuration

Setting	Description	Default	Example
debug.casDoctor.fatal	If the extra checks trigger an exception	true	false
debug.casDoctor.checks	Extra checks to perform when a CAS is saved (also on load if any repairs are enabled)	*unset*	comma-separated list of checks
debug.casDoctor.repairs	Repairs to be performed when a CAS is loaded - order matters!	*unset*	comma-separated list of repairs

Setting	Description	Default	Example
debug.casDoctor.forceReleaseBehavior	Behave as like a release version even if it is a beta or snapshot version.	false	true

Checks

All feature structures indexed

ID `AllFeatureStructuresIndexedCheck`

Related repairs Remove dangling chain links, Remove dangling relations, Re-index feature-attached spans, Remove dangling feature-attached span annotations

This check verifies if all reachable feature structures in the CAS are also indexed. We do not currently use any un-indexed feature structures. If there are any un-indexed feature structures in the CAS, it is likely due to a bug in the application and can cause undefined behavior.

For example, older versions of INCEpTION had a bug that caused deleted spans still to be accessible through relations which had used the span as a source or target.

This check is very extensive and slow.

Feature-attached spans truly attached

ID `FeatureAttachedSpanAnnotationsTrulyAttachedCheck`

Related repairs Re-attach feature-attached spans, Re-attach feature-attached spans and delete extras

Certain span layers are attached to another span layer through a feature reference from that second layer. For example, annotations in the POS layer must always be referenced from a Token annotation via the Token feature `pos`. This check ensures that annotations on layers such as the POS layer are properly referenced from the attaching layer (e.g. the Token layer).

Links reachable through chains

ID `LinksReachableThroughChainsCheck`

Related repairs Remove dangling chain links

Each chain in a chain layers consist of a **chain** and several **links**. The chain points to the first link and each link points to the following link. If the CAS contains any links that are not reachable through a chain, then this is likely due to a bug.

No multiple incoming relations

ID `NoMultipleIncomingRelationsCheck`

Check that nodes have only one in-going dependency relation inside the same annotation layer. Since dependency relations form a tree, every node of this tree can only have at most one parent node. This check outputs a message that includes the sentence number (useful to jump directly to the problem) and the actual offending dependency edges.

No 0-sized tokens and sentences

ID `NoZeroSizeTokensAndSentencesCheck`

Related repairs Remove 0-size tokens and sentences

Zero-sized tokens and sentences are not valid and can cause undefined behavior.

Relation offsets consistency

ID `RelationOffsetsCheck`

Related repairs Repair relation offsets

Checks that the offsets of relations match the target of the relation. This mirrors the DKPro Core convention that the offsets of a dependency relation must match the offsets of the dependent.

CASMetadata presence

ID `CASMetadataTypeIsPresentCheck`

Related repairs Upgrade CAS

Checks if the ìnternal type `CASMetadata is defined in the type system of this CAS. If this is not the case, then the application may not be able to detect concurrent modifications.

Repairs

Re-attach feature-attached spans

ID `ReattachFeatureAttachedSpanAnnotationsRepair`

This repair action attempts to attach spans that should be attached to another span, but are not. E.g. it tries to set the `pos` feature of tokens to the POS annotation for that respective token. The action is not performed if there are multiple stacked annotations to choose from. Stacked attached annotations would be an indication of a bug because attached layers are not allowed to stack.

This is a safe repair action as it does not delete anything.

Re-attach feature-attached spans and delete extras

ID `ReattachFeatureAttachedSpanAnnotationsAndDeleteExtrasRepair`

This is a destructive variant of Re-attach feature-attached spans. In addition to re-attaching unattached annotations, it also removes all extra candidates that cannot be attached. For example, if there are two unattached Lemma annotations at the position of a Token annotation, then one will be attached and the other will be deleted. Which one is attached and which one is deleted is undefined.

Re-index feature-attached spans

ID `ReindexFeatureAttachedSpanAnnotationsRepair`

This repair locates annotations that are reachable via a attach feature but which are not actually indexed in the CAS. Such annotations are then added back to the CAS indexes.

This is a safe repair action as it does not delete anything.

Repair relation offsets

ID `RelationOffsetsRepair`

Fixes that the offsets of relations match the target of the relation. This mirrors the DKPro Core convention that the offsets of a dependency relation must match the offsets of the dependent.

Remove dangling chain links

ID `RemoveDanglingChainLinksRepair`

This repair action removes all chain links that are not reachable through a chain.

Although this is a destructive repair action, it is likely a safe action in most cases. Users are not able see chain links that are not part of a chain in the user interface anyway.

Remove dangling feature-attached span annotations

ID `RemoveDanglingFeatureAttachedSpanAnnotationsRepair`

This repair action removes all annotations which are themselves no longer indexed (i.e. they have been deleted), but they are still reachable through some layer to which they had attached. This affects mainly the DKPro Core POS and Lemma layers.

Although this is a destructive repair action, it is sometimes a desired action because the user may know that they do not care to resurrect the deleted annotation as per Re-index feature-attached spans.

Remove dangling relations

ID `RemoveDanglingRelationsRepair`

This repair action removes all relations that point to unindexed spans.

Although this is a destructive repair action, it is likely a safe action in most cases. When deleting a span, normally any attached relations are also deleted (unless there is a bug). Dangling relations are not visible in the user interface.

Remove 0-size tokens and sentences

ID `RemoveZeroSizeTokensAndSentencesRepair`

This is a destructive repair action and should be used with care. When tokens are removed, also any attached lemma, POS, or stem annotations are removed. However, no relations that attach to lemma, POS, or stem are removed, thus this action could theoretically leave dangling relations behind. Thus, the Remove dangling relations repair action should be configured **after** this repair action in the settings file.

Upgrade CAS

ID `UpgradeCasRepair`

Ensures that the CAS is up-to-date with the project type system. It performs the same operation which is regularly performed when a user opens a document for annotation/curation.

This is considered to be safe repair action as it only garbage-collects data from the CAS that is no longer reachable anyway.

Annotation Guidelines

Providing your annotation team with guidelines helps assuring that every team member knows exactly what is expected of them.

Annotators can access the guidelines via the **Guidelines** button on the **annotation page**.

Project managers can provide these guidelines via the **Guidelines** tab in the **project settings**. Guidelines are provided as files (e.g. PDF files). To upload guidelines, click on **Choose files**, select a file from your local disc and then click **Import guidelines**. Remove a guideline document by selecting it and pressing the **Delete** button.

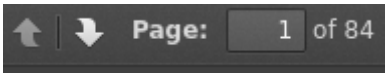
PDF Annotation Editor

Opening the PDF Editor

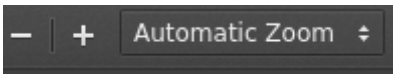
To switch to the PDF editor for an opened document, click on `Settings` in the `Document` panel located at the top. In the section `General Display Preferences` select `PDF` for the `Editor` field. Save your settings. The PDF editor will open.

Navigation

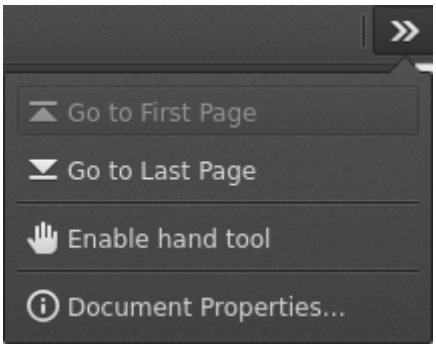
Once the editor is loaded you can see the opened PDF document. To navigate through the document you can hover your mouse over the PDF panel and use the mouse wheel for scrolling. After clicking in the PDF panel it is also possible to use the `Up` and `Down` keys of the keyboard. For a faster navigation the `Page Up` and `Page Down` keys can be used. In the PDF panel on the top left there are buttons for switching to the previous or next page in the document. Next to the buttons you can enter a page number to jump to a specific page.



In the top center of the PDF panel the zoom level of the document can be adjusted.



The button on the top right in the PDF panel opens a small menu which provides functionality to go to the first or last page. You can also use the `Home` and `End` keys instead. The menu also contains an option to enable the hand tool to navigate through the document via clicking and dragging the pages.



When moving through the document annotations will not show immediately. Once the movement stops for a short period of time the annotations for the previous, current and next page will be loaded. The loading process might take a few seconds.

Creating Span Annotations

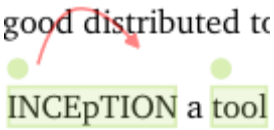
To create a span annotation first select the desired layer for it. This can be done in the `Layer box` on the right sidebar. If another annotation is already selected press the `Clear` button on the right sidebar.

Once you have chosen a layer, select the desired text to create a span annotation. This can be done by clicking at the beginning of the text span, dragging until the end and then releasing the mouse button. In the upper right corner you can see a moving circle which indicates that the creation of the annotation is in process.

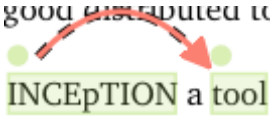
The creation might take a few seconds. Once finished the new span annotation will be rendered if it was created successfully. If it was not possible to create the annotation an error message is shown. After the span annotation is created it will be automatically selected in the `Layer box`.

Creating Relation Annotations

To create a relation annotation click on the knob of a span annotation and drag and drop on another span annotation knob. In order to create a relation annotation between two spans an according layer must exist.



After releasing the mouse button the creation process starts which is indicated by a moving circle in the upper right corner. This might take a few seconds. Once finished the new relation annotation will be rendered if it was created successfully. If it was not possible to create the annotation an error message is shown. After the relation annotation is created it will be automatically selected in the `Layer box`.



Currently long distance relation annotations are not creatable as annotations are rendered dynamically when moving through the pages.

Selecting Span and Relation Annotations

Span annotations can be selected by clicking on their span annotation knob.

To select relation annotations click on the relation arc.

Modifying Span and Relation Annotations

To modify an existing span or relation annotation you first need to select it.

Once an annotation is selected it will be shown in the `Layer box`.

You can now edit the selected annotation.

Deleting Span and Relation Annotations

First select the annotation that will be deleted.

The selected annotation will be shown in the `Layer box`.

To delete the annotation click on the `Delete` button on the right sidebar.

Accepting Recommendations

To accept and convert a recommendation to an actual span annotation select it.

The recommendation is converted to an actual annotation and is selected automatically.

Rejecting Recommendations

To reject a recommendation quickly double click the span annotation knob.

The recommendation then will be removed from the editor.

Appendices

Appendix A: 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.

When serializing a span annotation starts or ends in a space between tokens, then the annotation is truncated to start at the next token after the space or to end at the last token before the space. For example, if you consider the text `[one two]` and there is an some span annotation on `[one ]` (note the trailing space), the extent of this span annotation will be serialized as only covering `[one]`. It is not possible in this format to have annotations starting or ending in the space between tokens because the inter-token space is not rendered as a row and therefore is not addressable in the format.

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]
```

Appendix B: Formats

Table 20. 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

Appendix C: 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.