Oyster: A Tool for Fine-Grained Ontological Annotations in Free-Text

Hamed Tayebikhorami[1,2], Alejandro Metke-Jimenez[1], Anthony Nguyen[1], and Guido Zuccon[2]

[1] Australian E-Health Research Centre (CSIRO)
{hamed.tayebikhorami,alejandro.metke,anthony.nguyen}@csiro.au
[2] Queensland University of Technology
{g.zuccon}@qut.edu.au
{hamed.tayebikhorami}@connect.qut.edu.au

Abstract. Oyster is a web-based annotation tool that allows users to annotate free-text with respect to concepts defined in formal knowledge resources such as large domain ontologies. The tool has been explicitly designed to provide (manual and automatic) search functionalities to identify the best concept entities to be used for annotation. In addition, Oyster supports features such as annotations that span across non-adjacent tokens, multiple annotations per token, the identification of entity relationships and a user-friendly visualisation of the annotation including the use of filtering based on annotation types. Oyster is highly configurable and can be expanded to support a variety of knowledge resources. The tool can support a wide range of tasks involving human annotation, including named-entity extraction, relationship extraction, annotation correction and refinement.

1 Introduction

The development of techniques and tools for information extraction and named-entity recognition from free-text cannot prescind from the collection of annotated data, either for training such techniques (e.g. when using supervised methods) or for evaluating them. A number of tools have been developed for the acquisition of such annotated data. For example, BRAT is a popular web-based annotation tool that provides a user-friendly interface and the possibility to collect rich structured annotations [2]. WordFreak is an annotation tool that supports both human-provided and computer-generated annotation; active learning plugins are available that can learn from labelled data and suggest annotations for unseen data that can in turn be manually corrected by the annotators [3]. Other annotation tools for free-text data are publicly available, e.g., Knowtator[3], MMAX2[4] and GATE[5]; they often share similar functionalities but are all somehow limited in the annotation classes and tasks that are supported.

[3] http://knowtator.sourceforge.net/
[4] http://mmax.eml-research.de/
[5] http://gate.ac.uk/

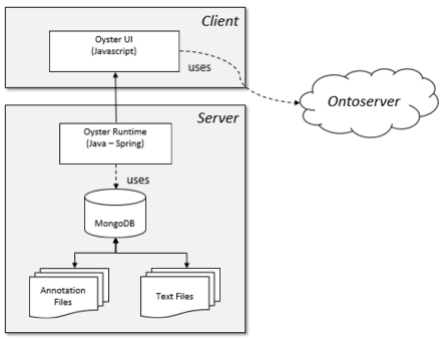

Fig. 1. The architecture of `Oyster`.

Despite the successful adoption of the tools mentioned above, tools like BRAT are difficult to use when numerous target annotation classes are considered. This is even more so when annotations need to be expressed against a large ontology, as it is the case when annotating clinical free-text data with the formal concepts defined in medical terminologies such as SNOMED CT. This paper introduces `Oyster`, an annotation tool that allows the fine-grained annotation of text with respect to large knowledge resources such as ontologies; it does so by supporting (both automated and manual) searching through the concepts defined within the knowledge resource. The version of `Oyster` described here is configured to provide annotations based on the SNOMED CT terminology via the Ontoserver platform [1], a terminology web server for clinical ontologies, and it is therefore tailored to the annotation of medical free-text data, such as discharge summaries, nurse handovers, radiology reports, etc. Nevertheless, `Oyster` can be easily extended to support the use of annotations from any knowledge resource, e.g. from Freebase (`https://www.freebase.com/`).

2 `Oyster`'s Architecture

Figure 1 shows the architecture of `Oyster`, which is divided into the client side and the server side. The client side is responsible to query the knowledge resource and identify candidate annotations within the resource itself. The server side is for serving the data and collecting annotations from the client and save it as JSON files.

The architecture is based on Spring MVC framework using JPA to connect with MongoDB database that is responsible for storing the free-text data. This data requires annotation, the annotations that are in turn collected, and the details associated to the users that provide the annotations. The server-side component is written in Java while the client-side component uses JavaScript, JSP and HTML, including libraries such as JQuery, Bootstrap and AnnotationJS, with this last library responsible for supporting the core annotation functionalities.

3 Features in Oyster

Next we describe the features developed in Oyster; we start from the key features that differentiate Oyster from existing annotation tools and we then turn our attention to other features present in the tool.

3.1 Ontology-based Annotations

This is the most significant feature that distinguishes Oyster from other annotation tools. The tool allows the provision of annotations with respect to named-entities (concepts) as defined in a reference knowledge resource, such as an ontology. The current version of Oyster integrates the SNOMED CT terminology as a target reference knowledge resource for annotations, but Oyster can be easily configured to point to other knowledge resources, making it a general purpose tool.

Automated Annotation Suggestions based on the Ontology. Another feature developed within Oyster is the automatic suggestion of candidate concept annotations derived from the reference knowledge resource. This feature aims to simplify the annotation efforts of human coders and therefore speed up the annotation process. When a text span is highlighted, Oyster suggests the best matching candidates from the reference knowledge resource for the user to select the appropriate entity (or entities if multiple ones are appropriate), see Figure 2. To produce candidate suggestions, Oyster currently uses the Ontoserver web service [1], issuing as query the text span for which an annotation is to be assigned to.

Manual Annotation Search on the Ontology. If the highlighted textual content does not retrieve the desired annotation, the user is allowed to search manually on the ontology. A list of candidate annotations will be updated with the new search results. If the desired match still cannot be found, the user can map the annotation to a high-level parent concept and flag it as a supertype.

3.2 Annotating multiple spans

Oyster allows users to select and annotate multiple textual spans by holding the ctrl key while selecting the desired text spans. As can be seen in Figure 2, the two disjoint text spans "Fecal occult blood" and "negative" were selected for annotation[6]. After selecting the desired entity (or entities) from the suggested list or manually searched list, the user may save the annotation using the "Save" button (Figure 2).

3.3 Relationship

Oyster can link annotations to each other based on entity relationships in the ontology. For instance, consider this sentence from a medical report: "the patient is **suspected** to have **urinary track infection**". The two bolded spans refer to two SNOMED CT concepts. The user can relate the two concepts by creating an annotation for each of them and then linking the "suspected" annotation to the "urinary track infection" annotation through a relationship called "clinical context" which is a SNOMED CT concept itself.

[6] Note that this is only feasible in the Firefox browser as other browsers do not support multiple selections.

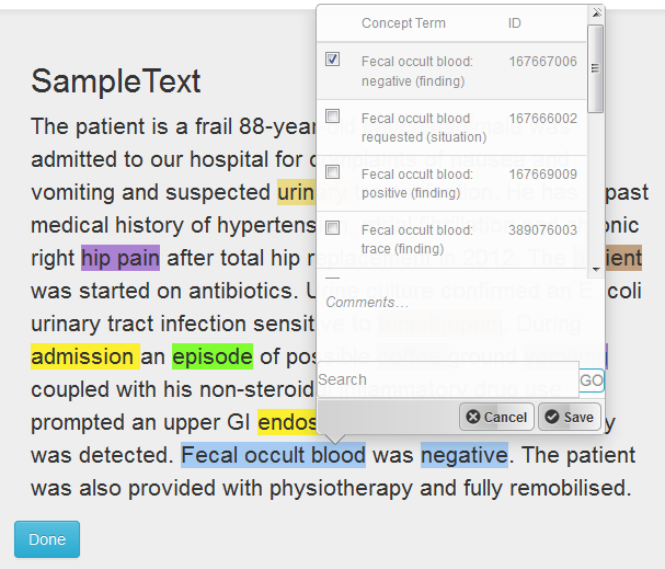

Fig. 2. The Oyster annotation process returning the best matching candidates from the reference knowledge resource for the user highlighted text spans.

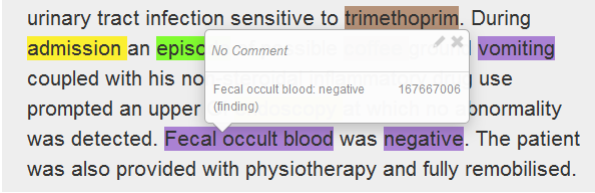

Fig. 3. Viewing annotations. Annotations can be updated or deleted suing the appropriate icons in the tooltip.

3.4 Category colour coding

`Oyster` allows administrators to choose color codes for ontology categories. Once the user creates an annotation, its span will be highlighted with the color code of the corresponding category. Figure 2 shows the highlighted spans that have colours corresponding to each top level hierarchy in SNOMED CT.

3.5 View and edit

After creating annotations for specific spans, the user can view the details of annotations by hovering the mouse on the highlighted span. Icons in the tooltip allow for the revision or deletion of the annotations (Figure 3).

3.6 Filtering

Users may filter their annotations based on the category or categories that they wish to view. This is handled by the filtering feature that allows users to filter annotations by categories (Figure 4).

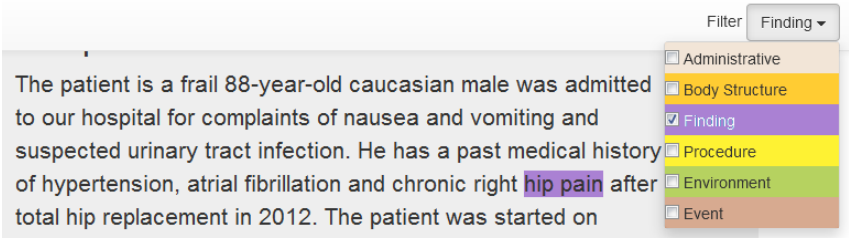

Fig. 4. Filtering annotations.

3.7 Timing

The time to annotate each file is measured in the application's background. Timing information can be used, for example, to study the complexity of certain annotation tasks and for estimating the duration of subsequent annotation tasks. `Oyster` measures the annotation time for each file without involving the user in time controls such as start, pause and resume (see Section 3.9 for more details).

3.8 File manager and user access

`Oyster` enables an administrator to upload documents (in plain text format; otherwise they are ignored) using drag and drop multiple file selection. Then the administrator can assign the uploaded text files to existing users. The files can be shared or private. If the file is shared, users can contribute together and view or edit each other's annotations, while if private, then each user has their own independent annotations.

3.9 Text file life cycle

After logging into the system, users can see a list of documents that have been assigned to them by the administrator. These documents or text files are shown along with three icons: not commenced (☐), in progress (🗹) and completed (✔). By selecting any document from the in-progress list, the time for that particular file will start (or resume if it has already started before). This time will be paused if another text file is selected or if the user has no activities (mouse or keyboard events) in the `Oyster` page for a pre-specified idle time. The timing will resume by selecting a document again or resuming activities in the `Oyster` page. The user can complete the annotation process by pushing the "Done" button which stops the timing and moves the document from the in-progress list to the completed list. The completed documents can only be viewed, which means that the user cannot edit or delete any annotations while the text file is completed. The user can continue editing a completed file by pushing the "Edit" button which moves the document back to the in-progress list (Figure 5).

4 Conclusion

In this paper we have presented `Oyster`, a tool for supporting the collection of annotation from free-text data. Unlike other tools for data annotation, `Oyster` allows annotations to be defined with respect to named entities contained in large formal knowledge resources; we showed examples that used `Oyster` to

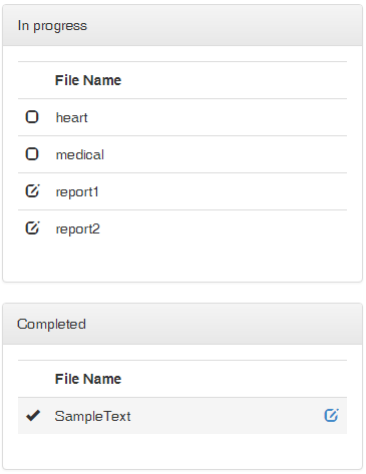

Fig. 5. Document list.

annotate text using SNOMED CT concepts. One of the key features of `Oyster` are the automated suggestion of candidate annotations based on named-entities' descriptions contained in the reference knowledge resource and the possibility of manually searching for alternative named-entities to be used for annotation. Future work will consider implementing plugins for `Oyster` to allow annotation using a number of different popular knowledge resources as reference target, e.g. Freebase. In addition, we plan to extend the current search functionalities to allow (1) searching through the assigned annotations (e.g., for consistency checking or further analysis), and (2) loading pre-annotated information along with the input documents, e.g. for annotation checking.

References

1. Simon McBride, Michael Lawley, Hugo Leroux, and Simon Gibson. Using australian medicines terminology (amt) and snomed ct-au to better support clinical research. *Studies in health technology and informatics*, 178:144–149, 2012.
2. Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107. Association for Computational Linguistics, 2012.
3. Jeremy LaCivita Thomas Morton. Wordfreak: an open tool for linguistic annotation. In *'03 Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, volume 4, pages 17–18. University of Pennsylvania, 2003.