Starting environment for human annotation of software mentions

Deliverable information	
Deliverable number and name	D4.1
Due date	M3
Actual delivery date	01.04.24
Work Package	WP4
Lead Partner for deliverable	Inria
Authors	Patrice Lopez, Samuel Scalbert, Alain Montei, Laurent Romary
Reviewers	
Approved by	
Dissemination level	Public
Version	0.2

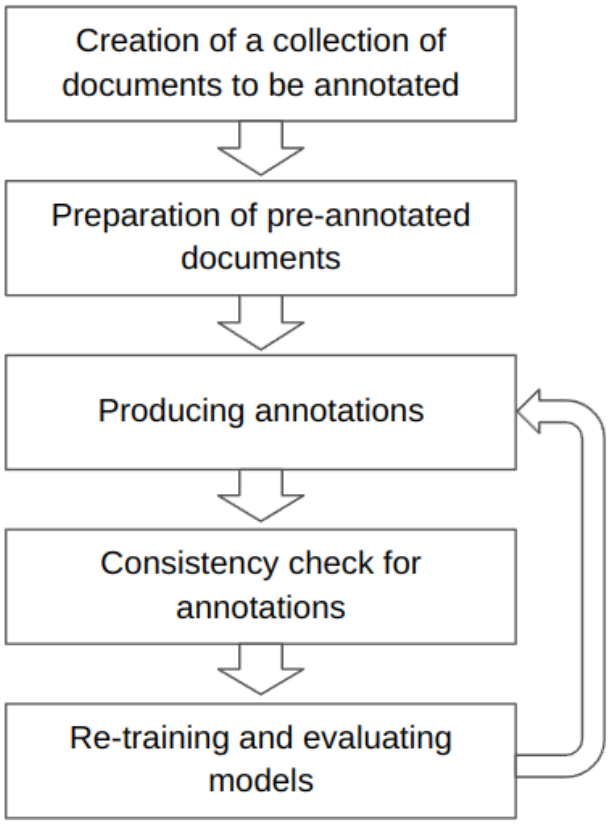
Document revision history			
Issue Date	Version	Author	Comments
19/02/2024	0.1		First draft
12/03/2023	0.1	Cezary Rosiński (Reviewer)	

Abbreviations

Executive summary

Overview

The annotation process can be streamlined as follows



Introduction

This deliverable presents the available environment for producing manual annotations of software mentions usable by the Softcite Software mention recognizer. It describes the requirements related to the documents to be annotated, to the manual annotations to be produced, and to the expected quality and formats. It also describes the different tools available to support the production of the annotations.

1 Creation of a collection of documents to be annotated

1.1 Nature of the documents

The documents considered here are scholarly articles, such as those published in journals, conference proceedings, book chapters and technical reports. Other types of documents (e.g. books, theses) are not supported.

The documents have to be published with an explicit license, compatible with unconstrained re-distribution and derivation. In practice, the document files have to be published under one of the following licenses: Public Domain, CC-0 or CC-BY. Documents under other licenses or unknown Open Access licenses have to be excluded from the annotation corpus, because it would limit our ability to publish the full training data openly and thus would hinder its reuse by other users.

1.2 Harvesting documents

The selection of the documents is typically specified by a list of criteria corresponding to the scientific/technical domains, publication years, document types, languages and venues. Bibliographical databases such as CORE could be used to select a set of document candidates in the form of a list of DOIs. Attention must be given to produced unbiased candidate lists, which might involve stratified sampling according to different metadata facets.

From a list of DOIs, *biblio-glutton-harvester* (https://github.com/kermitt2/biblio_glutton_harvester) can harvest metadata automatically and full texts (PDF, XML or latex sources) available in Open Access, with indication of the license of the full text files. License information is crucial for selecting documents for annotations. The tool has a high success rate of Open Access full text download, thanks to various web scraping techniques.

See in particular Harvesting from a list of DOI in the *biblio-glutton-harvester* documentation.

1.3 Selecting documents

The tool *biblio-glutton-harvester* produces a JSON catalog file (*map.json*) indicating the different resources in Open Access that have been successfully harvested. This file includes the Open Access license associated with the successfully downloaded resource, when such license information has been explicitly retrieved from Unpaywall, in combination with metadata from PubMed Central or arXiv.

Although the harvester can access XML and latex source versions of the publications, we suggest to focus only on PDF documents as the full text reference version for annotations. PDF is the general format available for Open Access papers (in particular for pre-prints), while XML or LaTeX versions are by far less frequent. Using PDF documents will make the training data more robust for machine learning models, also making them relevant for any kind of input, including XML and clean texts. Conversely, using XML as the only input format will make the processing of noisy PDF documents less reliable.

We can define additional selection criteria to ensure the usability of a given document for the annotation process. The following criteria shall be considered to select valid documents:

- successfully downloaded Open Access full text,
- full text file available under Public Domain, CC-0 or CC-BY license by filtering using the catalog JSON file,
- PDF version is harvested,
- TEI can be successfully produced from the PDF using Grobid (this ensures that the PDF contains usable text layers),
- the language of the publication is valid (typically, the ISO 639 language code is provided by Grobid), note that the currently available training data is limited to English,
- selected papers should be of reasonable length, for example between 3 and 15 pages, to facilitate human annotations.

Actions:

- The Softcite dataset consists of 4,971 full-texts in English (available in Open Access under CC-BY license), half in Life Sciences and half in Economics, for a total of around 46 million tokens. For SoFAIR, in order to improve performance across domains, the corpus would need a few hundred documents to be included in the corpus, covering domains such as Humanities, Mathematics, Physics, Chemistry. In addition, the new documents should be recent, because the Softcite dataset contains mainly articles published more than 10 years ago.
- Who : Open University
- What : Create corpus of pdf with licence CC-BY and metadata for Softcite
- When : beginning of april

2 Preparation of pre-annotated documents

2.1 Pre-annotated software mentions

From the selected full text PDF documents, the next step is to generate XML documents in the expected training data format, with pre-annotated software mentions following the current models.

This step can be realized via a utility command line of the Softcite Software Mention Recognizer, available under https://github.com/softcite/software-mentions. See the documentation for the installation of the tool and using this command line:

```
> ./gradlew create_training -Pin=/documents/in/ -Pout=/documents/out/
```

The pre-annotated documents will be available in XML TEI from the input PDF documents. The PDFs are parsed and automatically structured by Grobid. The existing Softcore models are then used on the relevant text structures to pre-annotate software mentions, which should help annotators when examining the documents.

2.2 Pre-classified mention context characterization

A complementary annotation task is to define the functional role of the software in the research work described in a publication, based on the software mention context. Given an XML corpus of software mention annotations, it is possible to extract the software mention contexts automatically with pre-classification information, for further manual annotation relative to the context characterization.

The utility command to perform this data preparation is available in the repository https://github.com/softcite/software-mentions, more precisely via this script. The script will extract content from annotated TEI XML documents, segment paragraph contexts into sentence contexts, and then pre-classify the context using the current Softcite classifier. The result is a JSON file with the mentioned contexts and a pre-classification.

Alternatively, it is also possible to simply start from a CSV file with sentences to be pre-classified. The same JSON file will be produced.

Action: Inria applies Software mention tool on corpus

3 Producing the annotations

3.1 Workflow

The workflow used for annotating the documents as the gold standard for the Softcite dataset (https://zenodo.org/records/7995565, 4971 full articles) is as follows:

- Double parallel annotation of an article by 2 different annotators
- Identification of disagreements
- Reconciliation by a third “expert” annotator

This process ensures that all the content of the documents and all the annotations have been seen and validated by at least 2 persons. This workflow was justified by the sparsity of software mentions in scholar articles. As it is very easy to overlook a mention, a double-blind annotation

appears as a robust solution, maintaining at the same time the quality of the positive examples (sentences and paragraphs with at least one annotation) and the negative ones useful for learning what is not software.

Preliminary training of the annotators using the existing annotated Softcite corpus is strongly recommended. Completion of the training period could be simply based on a certain volume of annotated texts with examination of disagreement against the existing annotated corpus, or when the annotations produced by the new annotators have reached a satisfactory accuracy against the existing annotated corpus.

3.2 Annotation guidelines

Annotation guidelines are the central part of the annotation process. They capture at the same time the scope of the annotations and the decision criteria to produce the annotations. Annotators must read the guidelines before annotating, and come back regularly to them as ambiguities and questions appear. When a new ambiguous case appears, annotators should extend the guidelines to cover this new case, thus capitalizing annotation experience and methods.

Annotation guidelines are also important to define the expected behavior of the Machine Learning models trained on the annotated corpus, so to specify the annotation task. ML models aim at reproducing the decisions defined in the guidelines, and the annotation guidelines could be seen as the annotation service description/contract.

Current annotation guidelines for software mentions are available at https://github.com/softcite/softcite_dataset_v2/blob/master/annotation_guidelines_tei_xml.md

Current annotation guidelines for the characterization of software mention contexts are available at https://github.com/kermitt2/kish/blob/master/resources/data/markdown/guidelines-softcite-context-classification.md

Following the guidelines is necessary to ensure that the produced annotations will be usable as additional training data. As the volume of existing training data is already very large and corresponds to several years of work, additional training data have to align with the existing ones to avoid degrading the machine learning models by inconsistent labeling.

More generally, the most common annotator mistakes are related to misalignment and annotation scheme clarity. These annotation guidelines address the two issues by supporting alignment of annotators with rules and example cases consolidated by the annotation of the Softcite corpus, which took place over several years.

With respect to the problem of deviating from the existing guidelines, we raise the following points:

1) If a particular attribute is not annotated in the new training data, the model will learn false negative cases conflicting with the older training data, making the whole attribute unreliable.
2) If a particular additional attribute not present in the original guidelines is annotated, it will be necessary to re-annotate this attribute in the existing 4971 documents to maintain some consistency, or to train a fully separate model just to this attribute. Introducing a new model supposes a significant change of the software mention recognizer and could only be considered if the development resources and skills are available.

3.3 Format of the annotations

Annotations are currently produced in XML following the TEI (Text Encoding Initiative) guidelines. This format can then be directly used to train the existing machine learning models. See the annotation guidelines and the existing Softcite corpus for illustrations.

Annotations can be produced using text or XML editors. It is usually considered that XML editors such as Oxygen could help the productivity of annotation. Annotators have to be familiar with XML, which might require additional preliminary learning and training.
We call `software mention` the reference to a software in a text, including the software names and every related component appearing together with the software name (e.g. software version, publisher, etc.). Software names and its software components are identified with TEI inline mark-up `<rs>`, for referencing string.

Extra XML attributes on mark-up are used to further refine the types of the identified entities. Relations between entities are encoded with attributes `@xml:id` and attribute `@corresp` as pointer;

Example of a Software mention with name, version and URL:

```
<p><rs type="software" xml:id="b308e79ccf-software-1">CRISPRFinder</rs>
version <rs corresp="#b308e79ccf-software-1"
type="version">1.0</rs> is freely accessible at <rs
corresp="#b308e79ccf-software-1"
type="url">http://crispr.u-psud.fr/Server/ CRISPRfinder.php</rs>.</p>
```

3.4 Document validation

Annotated documents can be checked using standard XML tools. XML well-formed should be supported by the user editor or can be easily tested using command lines such as *xmllint*.
This part is related to task 4.2

4 Consistency check for annotations

The XML annotated documents can normally be directly used for training models. In practice, these documents are supposed to be used in combination with the existing training data. Such a combination depends however on the strict respect of the guidelines and format. Deviating from the guidelines might degrade the existing models, either because non-consistent annotations have been produced (the model having contradictory decisions to be made, thus not able to learn stable inferences) or because some information will be skipped due to unexpected format.

In addition, the sparsity of software mentions makes overlooking mentions common, even with careful readers, and fatigue is unavoidable.

A tool is available at https://github.com/softcite/software-mentions to analyze the consistency of annotations over a complete corpus. See here for using this consistency script.

This tool analyzes the existing annotations and checks in the non-annotated contexts for the occurrence of seen mention without labeling or under a different label type. It allows one to identify automatically suspicious annotation contexts, which are then to be reviewed, because of inconsistencies between different documents or inside the same document. The tools appeared to be very useful to spot missing annotations and inconsistent annotation decisions for similar software mentions.

5 Re-training and evaluating models

Softcite Software Mention Recognizer (https://github.com/softcite/software-mentions) includes command lines to train and evaluate ML models, following a variety of ML architectures. It is possible to retrain models from scratch or incrementally. Evaluations can be done based on a fixed set (holdout set approach), random evaluation set, or via n-fold cross evaluation.

Training and Evaluation

Details on training and evaluating ML models are available here. See the configuration documentation to select the machine learning architecture to be used and its training parameters.

We identified that a very important aspect of the recognition of software mentions is negative sampling, i.e. the selection of negative contexts (text without any software mention annotation) to be used with the annotated contexts to help the models to learn what is *not* software. The reason is the very high sparsity of software mentions in scholar articles: using too many positive annotations would result in a model that sees software everywhere. On the contrary, using too many negative contexts will result in a model performing with a very low recall.

A pool of positive and negative examples is normally provided to the trainer as two TEI files derived from the full training data, see here. Different strategies can then be exploited to mix the positive and negative examples in an optimal way, maximizing the F1 score on a holdout set of full articles with a real distribution of mentions.

Depending on the outcome of the evaluation, it is often necessary to review the newly produced annotations or to produce more annotations. We recommend to evaluate new manual annotations by re-training and re-evaluating ML models very regularly, using frequent iterations. Given that manual annotation is very costly, this will make it possible to adjust the annotation work, quality and effort early, and limit the risk of producing low quality annotations for ML applications.

Action : from month 4 to 6 IBL-PAN give first result of the annotation progress to Inria to test the corresponding results in the model

6 SoFAIR Github

We propose to create in SoFAIR Github (https://github.com/SoFairOA) an environment dedicated to WP4 where each partner can put and find data and results of WP4.
We propose to have six folders like:

```
documents/metadata
documents/pdf
documents/tei-pre-annotated
documents/tei-validated
guidelines/
benchmarks/
```

In Github we can use issues to discuss annotation issues and pull requests for proposal solutions on tei-validated.

Action: create folders in GitHub repository for SoFAIR

7 KISH application

KISH application aims at facilitating annotation for people who are not comfortable with XML and TEI. https://github.com/kermitt2/kish
The result is in JSON and may be transformed in XML TEI
Currently KISH is not easy to set up and deploy, because the loading/export administration functions are done by Python command line. Using KISH requires a person with Python skills. In addition, the server to run the application has to be deployed and KISH installed.

Références

[1] Patrice Lopez, Caifan Du, Johanna Cohoon, Karthik Ram, and James Howison. 2021. Mining Software Entities in Scientific Literature: Document-level NER for an Extremely Imbalance and Large-scale Task. In Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, QLD, Australia. https://doi.org/10.1145/3459637.3481936

[2] Aricia Bassinet, Laetitia Bracco, Anne L'Hôte, Eric Jeangirard, Patrice Lopez, et Laurent Romary. 2023. Large-scale Machine-Learning analysis of scientific PDF for monitoring the production and the openness of research data and software in France. 2023. https://hal.science/hal-04121339

[3] Du C, Cohoon J, Lopez P, Howison J. Softcite dataset: A dataset of software mentions in biomedical and economic research publications. J Assoc Inf Sci Technol. 2021; 72: 870–884. https://doi.org/10.1002/asi.24454

[4] Du, C., Cohoon, J., Lopez, P., & Howison, J. 2022. Understanding progress in software citation: A study of software citation in the CORD-19 corpus. PeerJ Computer Science, 8, e1022. https://doi.org/10.7717/peerj-cs.1022

[5] David Schindler, Felix Bensmann, Stefan Dietze, and Frank Krüger. 2021. SoMeSci- A 5 Star Open Data Gold Standard Knowledge Graph of Software Mentions in Scientific Articles. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM '21). Association for Computing Machinery, New York, NY, USA, 4574–4583. https://doi.org/10.1145/3459637.3482017