Mining Software Entities in Scientific Literature: Document-level NER for an Extremely Imbalance and Large-scale Task

Patrice Lopez
science-miner
France
patrice.lopez@science-miner.com

Caifan Du
Johanna Cohoon
University of Texas at Austin
USA

Karthik Ram
Berkeley Institute for Data Science
USA

James Howison
University of Texas at Austin
USA

ABSTRACT

We present a comprehensive information extraction system dedicated to software entities in scientific literature. This task combines the complexity of automatic reading of scientific documents (PDF processing, document structuring, styled/rich text, scaling) with challenges specific to mining software entities: high heterogeneity and extreme sparsity of mentions, document-level cross-references, disambiguation of noisy software mentions and poor portability of Machine Learning approaches between highly specialized domains. While NER is a key component to recognize new and unseen software, considering this task as a simple NER application fails to address most of these issues.

In this paper, we propose a multi-model Machine Learning approach where raw documents are ingested by a cascade of document structuring processes applied not to text, but to layout token elements. The cascading process further enriches the relevant structures of the document with a Deep Learning software mention recognizer adapted to the high sparsity of mentions. The Machine Learning cascade culminates with entity disambiguation to alleviate false positives and to provide software entity linking. A bibliographical reference resolution is integrated to the process for attaching references cited alongside the software mentions.

Based on the first gold-standard annotated dataset developed for software mentions, this work establishes a new reference end-to-end performance for this task. Experiments with the CORD-19 publications have further demonstrated that our system provides practically usable performance and is scalable to the whole scientific corpus, enabling novel applications for crediting research software and for better understanding the impact of software in science.

CCS CONCEPTS

• **Computing methodologies** → **Information extraction**; • **Applied computing** → *Document analysis.*

KEYWORDS

Software; Scientific Literature; Entity Recognition; Entity Disambiguation; Document Analysis

ACM Reference Format:
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, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3481936

1 INTRODUCTION

Software is increasingly used to support research activities. Many researchers include software with their results, as shown by services like rOpenSci [5] and Papers with Code[1]. Quality software facilitates scientific progress and research reproducibility. Software, however, is still relatively invisible in research, citation databases, and academic search engines. Because it is impossible to search for research software as easily as for articles, users miss relevant software and software is not easily found by potential users. In addition, because identifying and crediting contributions of software developers is difficult, researchers have less incentives to develop better and more reusable software. While some recent works have focused on improving software cataloging [11] and standards for software citation [15], mining software mentions in the scientific literature as currently published offers a factual approach, directly usable to increase research software visibility.

Named entity recognition (NER) in scholarly literature has been successfully applied across a range of entity types, including species names, chemical, and biomedical entities [6, 12, 13, 30]. Mining software entities has attracted a lot of interest in the recent years. A recent review article [16] offers a comprehensive analysis of prior works that extract software and data mentioned in research articles. From 48 reviewed studies, of which 15 cover software and packages, Krüger and Schindle categorize four extraction approaches: (1) term search, (2) manual extraction, (3) rule-based extraction, and (4) supervised learning.

Term search was employed in 12 studies, although only one study related to software extraction. Term search refers to searching bibliographical databases for known string identifiers, such as URL,

[1]https://paperswithcode.com

Table 1: Distribution of software citations in the Biomedical domain from [14]

types	examples	proportion
mention with reference	... was calculated using **biosys (Swofford & Selander 1981)**...	37%
mention	... were analuzed using **MapQTL (4.0)** software...	31%
device-like	... **GraphPad prism** software (San Diego, CA, USA) was used for data analyses	19%
URL	... freely available from **http://www.cibiv.at/software/pda/**...	5%
mention with website	... using **BrainSuite2 (http://brainsuite.usc.edu)**...	5%
user manual reference	... as analyzed by the **BIAevaluation software (Biacore, 1997)**...	2%
unnamed	... was carried out using software implemented in the Java programming language	1%

DOI, or particular software names known in advance, strongly limiting the capacity of recognition. Manual extraction, applied in 16 studies with six addressing software, is relevant to occasional review work, but is not able to scale.

Rule-based approaches rely on pre-defined heuristics to automatically extract mentions of software from text. 16 of the studies reviewed by Krüger and Schindle used rule-based approaches; six of these were related to software extraction. While rules can achieve good precision, they are usually limited in terms of coverage. They are time-consuming to create and to maintain. Furthermore, since this approach is often used because of a lack of annotated data, rule evaluation can be unreliable and their portability to new domains is uncertain. Supervised learning has been used in four studies, two only covering software recognition [9, 10]. These and [27] use a CRF approach, while [16] experiments with a BiLSTM-CRF model. However, they are all based on datasets with very limited number of documents that over-represent software mentions. While *strict* supervised learning provides the best performance for NER applications, they typically require large and realistic sets of annotated data, which are missing for software entities.

Our system relies on a gold-standard annotated corpus, the Softcite Dataset [8], presented in section 2 and available online[2] under CC-BY license. This new dataset, by its size and quality, makes it possible to experiment with fully supervised Machine Learning (ML) approaches and to provide more rigorous evaluations. The analysis of the Softcite Dataset helped us understand how software are mentioned and identify the main requirements and challenges of this task (section 3), which go far beyond the traditional NER used by prior research. In this paper we provide details on our system, the Softcite recognizer, available as open source[3] (including models and evaluation scripts) under the Apache 2 license.

To better capture the specificities of mining scholarly literature, our efforts are realized from an end-to-end perspective. In particular, we address the concrete challenge of applying state-of-the-art ML methods to dozens of millions of published PDFs across different scientific domains, where software mentions represent only a few relevant tokens out of several thousands in every document. Finally, to validate our approach, we mined the CORD-19 PDF publication set [32], with results available online[4].

[2]https://zenodo.org/record/4445202
[3]https://github.com/ourresearch/software-mentions
[4]https://zenodo.org/record/5140437

Table 2: Overview of the annotations in the gold-standard Softcite Dataset, with the proportion per 1000 of annotated tokens given the total number of tokens of the corpus (total of 46,052,050 tokens).

annot. types	# annot.	# tokens	‰tokens
software name	5,172	6,396	0.139
version	1,591	3,627	0.079
publisher	1,358	2,686	0.058
URL	215	2,571	0.056
total	8336	15,280	0.332

2 THE SOFTCITE DATASET

We developed the Softcite Dataset [8] as a on-going long term effort to understand software citations and to support experiments with supervised learning. This dataset includes the mentions of *software names* and related attributes *version* (version number or version date), *publisher*, and *URL* from 4,971 full-text research articles, a total of 8,336 annotations. Articles were randomly selected from the PubMed Central (PMC) Open Access Subset (2,521 articles), and from open access articles in Economic (2,450 articles) via Unpaywall [29]. The manual annotation was realized with a multi-round annotation process starting from the complete article content and involved 38 human annotators. Percentage inter-annotator agreement for the annotations is 75.5%. All annotations were then reconciled and checked for a final choice by two expert annotators, ensuring consistency in the labeling and a *gold-standard* quality.

Table 2 presents an overview of the annotations in the dataset. The final published dataset includes only the paragraphs with at least one annotation. Before the Softcite Dataset, the largest similar dataset used for supervised learning experiments [9, 10, 16] was a set of 85 annotated documents (versus 4,971 in the Softcite Dataset) presented in [9], with average name mention frequency per document of 40.3 for the training set and 70.0 for the test set, whereas mention frequency is only 1.04 software name per document in the randomly selected Softcite Dataset.

3 TASK DESCRIPTION

3.1 Heterogeneity of software citations

In a previous study [14], summarized by Table 1, we observed that software citations in Biomedicine came in many different forms, dominated by informal mentions. These results show the necessity of identifying various types of information beyond a software name: associated bibliographical references, URL, software publishers and creators, and version numbers. Recognizing the associated bibliographical references takes several steps: identify the reference markers locally associated with a software mention, link these markers to their reference entries usually in the bibliography section, parse the references, and match them to registered bibliographical records such as a CrossRef DOI entries. URLs are frequently in footnotes, which necessitates cross-reference linking within the document.

3.2 Sparsity of mentions

As the count of software mentions in Table 2 shows, one of the key challenges in automatic software recognition is the low frequency of the mentions in documents. Only 1,441 out of 4,971 (29.0%) of the articles contain at least one software mention. This is even more significant at the level of tokens (including traditional delimiters but excluding space characters), which is the level of prediction in a sequence labeling process like NER. The 4,971 full-texts contain a total of around 46 million tokens, but only 15,280 tokens are relevant to a software mention; so around one token would be positively labeled for each 3,000 "negative" tokens, with a ratio as low as one token per 17,500 tokens for *publishers* and *URL* fields.

This issue is usually referred as the Class Imbalance Problem (CIP) in Machine Learning. CIP is determined by the Imbalance Ratio (IR), i.e., the ratio between the negative samples and the positives. An IR value above 500 is usually already considered to be extreme [20]. With the higher observed IR, an ML approach to finding new unseen software mentions is a major difficulty. We observed that very few studies investigated extreme CIP in the context of NER. These studies usually focused on chemical and biomedical entities, which, while highly imbalanced, do not present such extreme IR as we observe for software mentions [2, 31].

3.3 Document-level information

As mentioned previously, information related to cited software is often spread in different places within a document. In addition to this constraint, the Softcite Dataset shows that the distribution of software mentions in documents is not uniform and the same entity is frequently mentioned several times in one document [8]. Each context of citation can bring different information. A global view of the software mentions at document level is therefore a practical requirement for fully recognizing all mentions to software and further matching them to unique software entities.

3.4 PDF processing

Mining for specific entities is only relevant to certain textual structures of a scientific document, such as paragraphs, abstracts, figure captions, etc. In the Softcite corpus, we calculated that, on average, 28% of publication content should be filtered out. This includes metadata (author, affiliations), bibliographical sections, table and figure content, formulas, headnotes, page numbers, reference markers, or editorial annexes (like conflicts of interest).

Text mining work usually assumes the availability of clean and structured text enabling recognition only on relevant document parts, but this is not the condition of real world applications. The most widespread and easily available scientific publication format is raw PDF, a presentation-oriented format that destroys the semantics and the original structure of data, introducing noise in text encoding and text order stream. This format raises major issues for text mining applications, both in terms of significant source of errors and technical feasibility [33].

Publishers' structured XML including the text body, such as JATS, is text-mining friendly, but has limited availability. Decades of back files are only available in PDF. Even when available, XML full-texts are in a variety of different native publisher XML formats, often incomplete and inconsistent from one to another, difficult to use at scale. Moreover, XML full-texts are usually subject to numerous complex and time-consuming commercial agreements. Thus, supporting PDF by using a layout aware parsing and conversion tool appeared very early as a key requirement for our task.

3.5 Specialized domains

Technical and scientific documents aim at supporting specialist communication and are thus written in specialist language, 30-80% of which is composed of terminology according to [1]. Scientific texts follow thus a specialized language with specific vocabularies and nomenclatures highly depending on technical domains.

The usage of existing general-purpose NLP components on scientific text results most of the time in a significant loss of accuracy as compared to custom models. For example, ScispaCy improves F1-scores by 3.3 to 23.5 points on some tasks related to Biomedical literature as compared to the general domain spaCy [25]. SciBERT surpasses BERT-base model by 0.1 to 7.07 points in F1-score depending on the task [3]. We also observed in these works that the portability of ML models from one scientific field to another is uneven, even when customized. This directly challenges the recognition of cross-domain entities like software across fields.

4 DOCUMENT PROCESSING AS A CASCADE OF MACHINE LEARNING MODELS

Figure 1 presents the Softcite pipeline, which tries to address these challenges in an accurate, scalable, and generic manner. The different approaches and components are presented in details in the next sections.

4.1 Document structuring

We rely on GROBID[5] for parsing, extracting, and structuring the content of scientific articles in PDF, but also to drive further entity mining in relevant sections. GROBID is an open source library specialized for scholarly PDF implementing a cascade of ML models to mark up the structure of a document. The tool, created by the first author of this paper [22], is used for this purpose by many large scientific information service providers such as ResearchGate, Academia.edu, and Internet Archive, by large-scale citation services, for example scite.ai, which has extracted more than 800 million bibliographical citation contexts with this tool [26], or for creating machine-friendly datasets of research papers, like the recent PDF set of the CORD-19 dataset [32].

[5] https://github.com/kermitt2/grobid

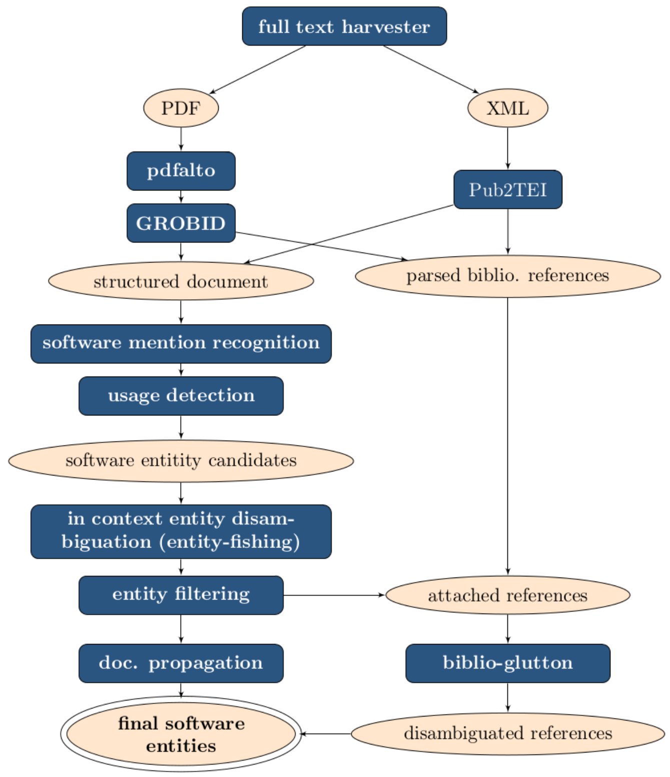


Figure 1: Overview of the Softcite software extraction pipeline. Blue boxes represent the main processing components, and ovals the different data results.

Each ML model in GROBID is a sequence labeling model. Sequence labeling is defined in an abstract manner and its concrete implementation can be selected by choosing among standard ML architectures, including a fast linear chain CRF and a variety of state-of-the-art Deep Learning (DL) models. Sequence labeling models are limited to the labeling of a linear sequence of text, therefore they associate a one-dimension structure to a stream of tokens. One way to create additional levels of embedded structures is to cascade several sequence labeling models, the output of a first model being piped to one or several models. This is the approach taken by GROBID, and Figure 2 shows the current model cascade. Each model can use a different sequence labeling algorithm, different features, and different tokenizers, depending on the labels used by the particular model, on the amount of available training data, on the runtime, memory, and accuracy constraints, etc.

In addition to layout recognition and recovery of the reading order, GROBID and its PDF parsing component *pdfalto* handle a variety of text cleaning processes: UTF-8 encoding and character composition, de-hyphenization, reconnecting paragraphs interrupted by a page break, a figure or a table, special font resolution, recognition of formula, etc. Complementary to the support of ALTO, a modern format for OCR output, *pdfalto* implements additional features relevant to scientific documents, in particular the recognition of superscript/subscript style and the robust recognition of line numbers for review manuscripts.

Bibliographical references and the corresponding reference callouts within the text body are also identified and parsed following a set of dedicated ML models, with a F1-score between 80 and 86%, depending on the evaluation approach, for a dataset of 1,943 PubMed Central articles. Finally a parsed reference with incomplete metadata is resolved against the current 105 million DOI records of CrossRef via *biblio-glutton*[6], a fast open source reference matching service showing an accuracy of 95.39 F1-score on a set of 17,015 raw bibliographical reference/DOI pairs extracted from PubMed Central PDF/JATS articles (see biblio-glutton repository).

Various evaluations for the different GROBID models and for the end-to-end process are available at the project repository[7]. These evaluations are continuously updated with new releases.

4.2 Layout tokens, not text

If processing PDF is very challenging from a text mining perspective, this format provides a lot of advantages to the human readers[8]. We think it is possible to exploit some of these advantages to better present and interact with text mining results. More precisely by calculating the bounding box coordinates of the extracted information in the PDF, we make possible *augmented* interactive PDF.

In the complete cascading of process, GROBID does not manipulate text but *Layout Tokens*, a structure containing the Unicode text token but also the associated available rich text information (font size and name, style attributes, etc.) and the location in the PDF expressed by bounding boxes. Layout Tokens are grouped following layout criteria (lines, blocks, columns) as a first result of the PDF layout analysis by *pdfalto*, and then further semantically grouped through the ML process, as a succession of labeled fields.

Operations on 2D bounding boxes are well known and straightforward to apply to Layout elements. By synchronizing the bounding boxes with the sequence labeling, we can render any text mining results on their original PDF source. Text mining is then not limited to populating a database, it allows user-friendly visualizations of semantically enriched documents and new interactions.

Layout information is also used to instantiate layout features, which can be exploited or not depending on the capacity of the ML models. Layout features are crucial for the reliable recognition of structures such as titles, abstracts, section titles, figures, tables, reference markers which are often only characterized by their relative position (vertical space, indentation, blocks, etc.) and font style (e.g. *superscript* for reference markers). While the layout features are not directly involved in the recognition of software names, they are used to identify and select relevant structures to be processed, to recover valid text order, encoding, and text tokens, and in the identification of reference markers attached to software mentions.

4.3 Identification of Candidate Mentions

Similarly as the models in GROBID, software mention recognition is implemented as a sequence labeling task, where the labels are

[6]https://github.com/kermitt2/biblio-glutton
[7]https://grobid.readthedocs.io/en/latest/Benchmarking/
[8]The main advantages for the users are a fixed layout, a preservation format, a universal support by document visualization and editing tools, the ease of storage, annotations and additions of web links and, finally, a professional printing typeset quality.

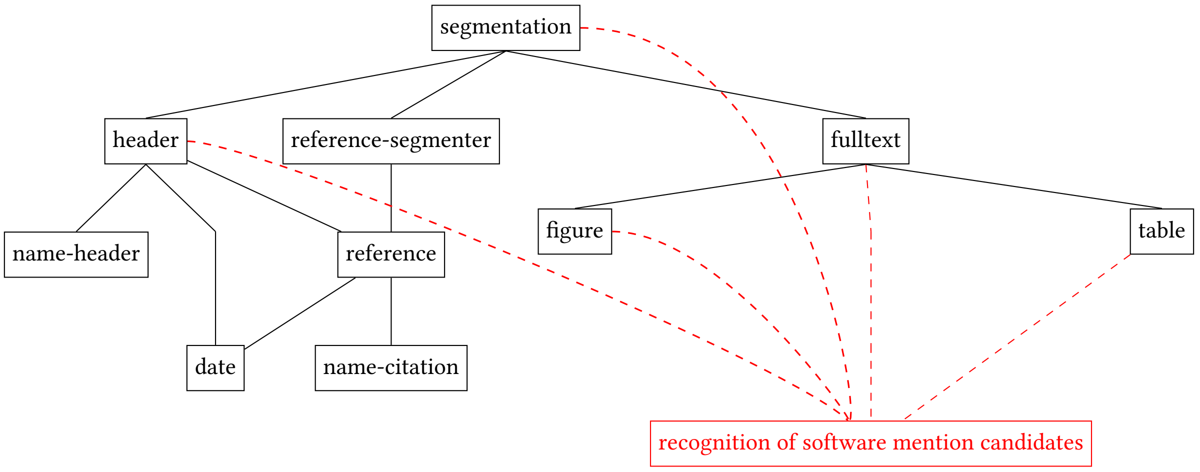


Figure 2: The GROBID cascade of sequence labeling models. The software mention model (in red) is added as a downstream model applied on certain outputs of the existing GROBID models. These GROBID models correspond to the relevant textual structures where a software mention can take place (title, abstract, and keywords from the article header; paragraphs and section titles from the fulltext body; captions from the figure and table parts; and finally footnotes from the main segmentation).

applied to sequences of Layout Tokens, identifying relevant software names and attributes. This sequence labeling model is added to the GROBID cascade, applied to a selection of relevant structures identified by upstream GROBID models. Figure 2 presents the cascading process and which structures are considered for software recognition. We call this approach *structure-aware document annotation*, in contrast to the large majority of text mining approaches for scientific literature which ignore the document structure and its related semantics. Tokens are propagated to downstream models with information about the structures where they appear and layout attributes, allowing the pipeline to exploit more contextual information than usual approaches.

4.3.1 Undersampling, holdout set, and training set. In section 3.2, we stressed the very challenging nature of the Imbalance Ratio for this NER application. In ML, the CIP is usually addressed by sampling strategies used either for removing data from the majority class (undersampling) or adding data to the minority class via artificially generated examples or Active Learning (oversampling). With these sampling techniques, the training data is viewed as a working resource which is crafted to maximize the accuracy of the trained model. An n-fold cross evaluation or any random splitting on this working set is here misleading because the distribution of the classes is artificially modified to boost the accuracy of the less frequent classes. For evaluating a model, in particular against the CIP effects, we need to create a stable holdout set as close as possible to the observed class distribution and data diversity.

Holdout set. The Holdout set is defined at document-level in order to allow document-level evaluation. We selected 20% of the Softcite Dataset full-texts (994 articles), reproducing the overall distribution of documents with annotation (29.0%), the distribution between Biomedicine and Economics fields, and we used a stratified sampling to reproduce the overall distribution of mentions per document.

Training set. The remaining 80% of documents (3,977 articles) was then divided at paragraph-level into positive (1,886 paragraphs with at least one manual annotation) and negative (612,597 paragraphs without manual annotations). Negative examples here are viewed as a pool of possible negative samples to be added to the positives examples, depending on the undersampling strategies.

Undersampling methods. In this work, we focused on adapting undersampling approaches to NER where undersampling being the most commonly used approach for addressing CIP in classification problems [21]. We experimented with two different approaches to reducing the weight of the negative majority class:

- **Random negative sampling**. Different ratio of random negative paragraph examples are used in combination with all the positive paragraph examples to create training sets. We considered ratio from 1 to 50 and selected the best ratio using a validation set. We identified the best ratio at 15, with overall accuracy decreasing beyond 20, the positive samples becoming too diluted in comparison with the negative ones.
- **Active negative sampling**. A model trained only with positive examples was first created. This model was then applied to all the paragraphs without manual annotations, and we selected those where a mention is wrongly predicted, complemented with random sampling to reach the experimentally defined ratio. With this approach, we suppose that negative examples where such errors occur can better contribute to the improvement of the models for controlling false positives.

As baseline (noted *none*), we also ran NER models trained only on positive paragraphs, which corresponds to the Softcite Dataset, without any sampling.

Sequence labeling algorithms. The labeling is done paragraph by paragraph, as extracted by the GROBID upstream models. The following sequence labeling algorithms have been benchmarked:

Table 3: Summary of scores (P: Precision, R: Recall, F: F1-score) at span level (exact match) against the holdout set (994 complete articles). *no sampling* **refers to a training with only paragraphs containing at least one annotation from the 80% remaining articles. Paragraphs without annotations (negative sampling) are then added to the training data via** *random sampling* **or** *active sample***. Bold indicates the best scores for a given field. Reported scores for DL models are averaged over 5 training/runs.**

model	under-	software name			publisher			version			URL			F1 micro
	sampling	P	R	F	P	R	F	P	R	F	P	R	F	average
CRF (custom features)	none	29.2	58.5	38.9	41.5	76.6	53.8	51.9	84.9	64.4	18.2	68.6	28.7	45.8
	random	66.9	53.7	59.6	70.4	75.1	72.7	79.8	83.6	81.6	34.8	45.7	39.5	66.3
	active	69.0	52.8	59.8	70.3	73.7	72.0	80.9	82.7	81.8	32.6	42.9	37.0	66.2
BiLSTM-CRF	none	21.9	68.5	33.2	45.3	82.8	58.5	53.6	90.5	67.3	16.7	57.1	25.8	41.9
	random	57.1	71.9	63.7	67.4	85.2	75.3	73.0	88.7	80.1	51.0	74.3	60.5	69.0
	active	62.7	68.5	65.5	69.0	85.2	76.2	63.5	**92.6**	75.4	**63.2**	68.6	65.8	69.8
BiLSTM-CRF +features	none	20.9	74.5	32.7	45.7	**85.7**	59.6	58.4	91.8	71.4	14.5	48.6	22.4	41.4
	random	54.1	73.6	62.4	68.5	84.2	75.5	72.2	92.2	81.0	50.0	65.7	56.8	68.3
	active	54.5	73.3	62.5	68.2	85.2	75.7	79.5	92.2	85.4	47.5	**80.0**	59.6	69.3
BiLSTM-CRF +ELMO	none	35.6	74.9	48.2	71.6	79.4	75.3	72.9	88.3	79.8	11.6	**80.0**	20.3	54.5
	random	67.4	63.0	65.1	63.9	83.7	72.5	**83.1**	84.9	**83.9**	54.8	48.6	51.5	70.2
	active	61.9	70.4	65.9	74.1	84.7	**79.0**	77.7	90.5	83.6	48.0	68.6	**56.5**	71.6
BERT-base -CRF	none	15.1	74.2	25.1	40.2	79.4	53.4	42.1	87.9	56.9	04.5	71.4	08.5	30.4
	random	52.8	67.8	59.3	61.6	79.0	69.2	65.9	85.3	74.3	15.0	54.3	23.5	61.9
	active	56.9	67.9	61.9	66.1	78.5	71.8	73.5	85.3	79.0	19.0	54.3	28.2	65.3
SCIBERT-CRF	none	25.7	**80.4**	39.0	44.1	84.7	58.0	71.7	92.2	80.7	27.8	71.4	40.0	47.6
	random	60.5	77.0	67.8	68.1	82.8	74.7	75.4	91.3	82.6	40.3	71.4	51.6	71.4
	active	**69.3**	72.8	**71.0**	**75.6**	82.8	**79.0**	80.2	87.9	**83.9**	45.3	68.6	54.6	**74.6**

- **linear chain CRF**: Conditional Random Fields with custom feature engineering [17],
- **BiLSTM-CRF**: Bidirectional LSTM-CRF with Gloves static embeddings [18],
- **BiLSTM-CRF + features**: Bidirectional LSTM-CRF with Glove static embeddings including a feature channel,
- **BiLSTM-CRF+ELMO**: Bidirectional LSTM-CRF with Gloves static embeddings and ELMO dynamic embeddings [28]
- **BERT-base-CRF**: fine-tuned BERT base model [7] with CRF activation layer, pre-trained on general English text
- **SCIBERT-CRF**: fine-tuned BERT base model pre-trained on scientific text [3] with CRF activation layer,

The CRF implementation is based on a custom optimized fork of *Wapiti*[9] [19]. The other algorithms rely on the Deep Learning library *DeLFT*[10] built on top of TensorFlow and Keras. All are natively integrated in GROBID JVM to optimize runtime. Transformers use CRF as final activation layer, which provides a stable improvement of 0.3 to 0.5 points in F1-score as compared to a softmax layer.

All these models take exactly the same input sequence, associated with the same features, and use strictly the same evaluation method, avoiding some common experimental biases related to preprocessing and evaluation. Hyperparameters of the Deep Learning models have been set experimentally on a validation set part of the training data and an early stop is used.

The different implemented sequence labeling algorithms give a comprehensive picture of the best named entity recognition approaches available today. In order to select the most adapted to our application, we evaluated them against three criteria: accuracy, runtime, and domain portability.

[9]https://github.com/kermitt2/wapiti
[10]https://github.com/kermitt2/delft

4.3.2 Accuracy. We observe with Table 3 that CIP is indeed a major concern for software mention recognition. Training models without consideration of CIP can lead to models two times less accurate. SCIBERT-CRF, which is pretrained on scientific texts, performs better than the general-purpose BERT-base or BiLSTM-CRF with Gloves embeddings. BiLSTM-CRF+ELMO, although pre-trained on general language, provides solid F1-score but benefits less from the negative sampling than SCIBERT-CRF. In general, we observe that DL models benefit significantly more from the undersampling than CRF only. We further observe that additional custom features has no significant impact when used by the DL models.

4.3.3 Runtime. Given the scaling constraint of scientific text mining, the accuracy of the models must be considered in balance with their runtime. Table 5 shows that the runtimes can differ extremely from one model to another one. CRF scales particularly well on commodity hardware, running more than three times faster than the fastest DL approach with GPU, BiLSTM-CRF. Although accurate, BiLSTM-CRF+ELMO is more than 250 times slower than CRF, which makes impossible a usage for mining scientific literature at scale. SCIBERT-CRF combines a strong accuracy with a more acceptable runtime, 14 times faster than BiLSTM-CRF+ELMO.

4.3.4 Domain portability. One aspect often neglected in the evaluation of information extraction models is the capacity of a model to perform well on a domain different from the one it has been trained on. As mentioned in section 3.5, this is particularly relevant in science, a mosaic of different highly specialized languages.

Table 4: Evaluation of domain portability. The models have been trained on the PMC sub-collection and are evaluated on the Economics domain. The numbers are F1-scores, averaged over 5 training for the indicated DL models.

Trained on Biomedicine	Evaluated on Economics				
models	software	publisher	version	URL	micro-average
CRF (custom features)	37.9	13.7	48.6	16.0	35.9
BiLSTM-CRF	51.0	22.0	57.8	**58.3**	49.1
BiLSTM-CRF+ELMO	53.4	19.1	57.1	53.3	51.2
BERT-base-CRF	45.6	17.0	66.7	17.0	42.6
SCIBERT-CRF	**58.6**	**34.8**	**80.7**	46.2	**57.9**

Table 5: Average runtimes of different sequence labeling models. The runtimes were obtained on a Ubuntu 18.04 server Intel i7-4790 (4 CPU), 4.00 GHz with 16 GB memory. The runtimes for the Deep Learning architectures are based on the same machine with an Nvidia GPU GeForce 1080Ti (11 GB). Runtime can be reproduced with a python script in the project GitHub repository.

model	best modality	layout tokens/s
CRF	CPU threads: 8	100,879
BiLSTM-CRF	GPU batch size: 200	30,520
BiLSTM-CRF+ELMO	GPU batch size: 7	365
SCIBERT-CRF	GPU batch size: 6	5,060

For evaluating domain portability, we have trained the models on the PubMed Central collection (1,109 documents with at least one software mention for a total of 6,096 annotations), covering the biomedical domain, and performed an evaluation on articles in Economics, a notably different domain (119 articles with at least one software mention for a total of 538 annotations). Results are summarized by Table 4.

While we see that covering a new domain is challenging, Deep Learning models all show a much stronger average recognition accuracy than CRF on all the fields, in particular for SCIBERT-CRF, the difference of accuracy being significantly amplified in the unseen domain. The capacity to recognize new, unseen software information in a variety of domains is critical for the creation of knowledge bases. SCIBERT-CRF shows the best portability, while combining best accuracy and a manageable scalability. This model appears today as the best choice for large-scale text mining and entity discovery in scientific literature.

4.4 Attachment of software attributes and reference markers

We use simple proximity rules to attach the software attributes (*version*, *publisher*, *URL*) to their corresponding *software name* mention and to attach a bibliographical reference marker to a software mention. These rules were experimentally developed and evaluated against the Softcite Dataset, where the relations between attributes and software names are manually encoded.

4.4.1 Attachment of software attributes. Software attributes can be attached to a software name (viewed as *head* of the full software component) only if they occur in the same paragraph. We did not observe attachment beyond a paragraph in the Softcite Dataset.

Attachment of a software attribute with character offsets a_start and a_end is as follow:

(1) attribute is discarded if no software name is present in the paragraph,
(2) attribute a is attached to the software name n at position offsets (n_start, n_end) minimizing the following distance function d:

$$d(a,n) = \begin{cases} (\texttt{n\_start} - \texttt{a\_end}) \times 2, & \text{if } \texttt{a\_end} \leq \texttt{n\_start} \\ \texttt{a\_start} - \texttt{n\_end}, & \text{otherwise} \end{cases}$$

4.4.2 Attachment of bibliographical reference markers. Given a software component (composed of a software name plus zero or several software attributes), we note *n_end* the end character offset of the software name and *g_end* the end character offset of the complete component (maximum offsets of all the software attributes and name positions of the software component, so if no attribute present n_end = g_end). A bibliographical reference marker is attached to a software mention group if it is overlapping the chunk $[\texttt{n\_end}, \texttt{g\_end} + k]$, where k is set experimentally. Based on the Softcite Dataset, we use $k = 5$.

4.4.3 Evaluation of the attachment rules. Table 6 presents the accuracy of attaching software attributes to the correct software names in the Softcite Dataset. Given that these simple rules perform accurately in the case of software mentions, we did not investigate more sophisticated approaches.

Table 6: Evaluation of attribute attachment to the correct software name, for a total of 2,537 expected attachments.

attribute fields	precision	recall	F1-score
version	99.6	98.5	99.1
publisher	99.5	98.3	98.9
URL	98.7	95.4	97.0
biblio. reference	97.5	100	98.7
all (micro-avg)	99.4	98.2	98.8

4.5 Software usage detection

Whether or not a software mentioned in an article was in fact used in the research is important to understand why a software is cited and to better credit research developers for the actual impact of

Table 7: Evaluation of the usage prediction for mentioned software. Annotated examples are from the Softcite Dataset and the scores (P: Precision, R: Recall, F: F1-score) are micro-average average over 10-folds. Implementations are realized with DeLFT, a library based on Keras/TensorFlow.

software usage	annot. count	BidGRU × 10			SciBERT		
		P	R	F	P	R	F
used	3736	96.5	99.2	**97.9**	95.6	99.5	97.5
not used	357	86.4	57.6	**69.1**	88.2	45.4	60.0

their contributions. To explore the feasibility of an ML approach to this task, complementary binary Deep Learning classifiers have been trained to predict if the mentioned software is used or not in the research work described in a publication. Usage information are encoded in the Softcite Dataset with a silver-level quality (annotation by possibly several annotators with majority vote, but no final reconciliation by a an expert). We hypothesize here that the wording used to introduce and describe a software mention can characterize its possible usage. The sentences containing the mentions are used as classifier input, without additional features.

Results are shown in Table 7. We found that a BidGRU architecture with Glove embeddings in a 10-classifiers ensemble (combination of 10 classifiers trained on 10 different partitions of the training data), the best RNN model we have experimented, surpasses SciBert significantly. This result is relatively unexpected because SciBert provides the best performance in many classification tasks related to scientific text [3]. However, the accuracy of the minority class *not used* is low for a practical usage. We will address the performance of the models by developing more training data and by increasing the data labeling quality to gold-standard.

4.6 Software Candidates and Entity disambiguation

Entity disambiguation, or entity linking, is the task of matching a raw mention to the concept it references. We consider here Wikidata as the reference set of entities. Wikidata currently contains around 13K entities corresponding to software (excluding video games), but also several million other scientific entities that can be used to detect false positives.

We use *entity-fishing*[11] [23] as the entity disambiguation library for the following reasons: the library is able to match in-context mentions (exploiting wording around the mention) against the complete Wikidata, is very fast in comparison to all other alternatives (full linking up to 4800 tokens/s on a 4-core server), it does not require a GPU, is highly memory-efficient as compared to other existing systems, and is designed to disambiguate any terms (not just Named Entities). It provides an accuracy close to the state of the art. For instance, in the overall unnormalized accuracy scenario for AIDA-CONLL-testb, it performs at 76.5 F1-score, compared to 80.27 for the recent BLINK system [34], a fine-tuned Bert considerably more resource-hungry and slower, limited to the English Wikipedia—and surpasses this system for the AQUAINT (89.1 vs. 85.88) and MSNBC (86.7 vs. 85.09) evaluation sets.

[11] https://github.com/kermitt2/entity-fishing

Table 8: Comparison between paragraph-level (default, noted ¶) and document-level (noted *doc.*) processing with and without entity disambiguation filtering (F1-scores on holdout set, average of 5 train/runs).

fields	SciBERT-CRF		SciBERT-CRF + entity disambiguation	
	¶	doc.	¶	doc.
software	71.0	74.1 (+3.1)	74.3 (+3.3)	**76.7** (+5.7)
publisher	79.0	76.2 (−2.8)	**81.9** (+2,9)	78.3 (−0.7)
version	83.9	84.7 (+0.8)	87.1 (+3.2)	**88.3** (+4.4)
URL	54.6	64.9 (+10.3)	55.4 (+0.8)	**66.7** (+12,1)
micro-avg	74.6	76.4 (+1.8)	77.7 (+3.1)	**79.1** (+4.5)

The disambiguation is realized with an ensemble approach (Gradient Tree Boosting) using a variety of features, including semantic vector similarity [4] and graph-based relatedness measure [24]. The context to be disambiguated is defined as the paragraph where a candidate mention occurs. The amount of text to be disambiguated is thus limited on average to a few paragraphs per document and remains scalable.

The first usage of the entity disambiguation is to filter out false positives. If a candidate software mention is likely a known scientific entity other than software in its context of usage, we discard the candidate. Table 8 shows the corresponding improvement of software mention recognition with a disambiguation score threshold of 0.4 (any non-software entities predicted above this threshold is considered as false positive software mention). We think that these results are encouraging and could be further improved by additional custom training of the disambiguation, the selection of improved contexts and extending Wikidata to cover more research software entities from other curated sources.

The second usage is more traditional entity linking. The mentions successfully disambiguated are enriched with Wikidata and English Wikipedia page identifiers. These identifiers and related information can help to further deduplicate mentions corresponding to the same software across different publications. Although entity-fishing is comprehensively benchmarked for general-domain texts, we do not currently have data manually annotated at entity level to evaluate the particular case of software entity linking.

4.7 Document-level processing

When processing a complete document, it is frequent that only some instances of mentions of the same software are identified, while others remained unlabeled. The wording around some mentions and the presence of other software attributes impact the performance of recognition. To address this, we apply a propagation of the matched software names to other unlabeled occurrences of the same term in the same document.

To avoid propagating a mention to spurious common and short strings, we apply a TF-IDF threshold to the software names to be propagated, keeping only those which are significant given the background full Softcite corpus. Table 8 presents an evaluation of this step with a TF-IDF threshold experimentally set at 0.001 using a validation set of the training data.

Table 9: Results of a re-harvested version of CORD-19 using the metadata file dated 2021-03-22.

	total count
total Open Access full texts	211,213
- with at least one annot.	76,448
software names	295,609
- with linked Wikidata ID	117,193
publishers	61,804
versions	104,199
URL	27,916
- with biblio. references	49,184
references with DOI	15,931
references with PMID	10,611

5 APPLICATION TO CORD-19

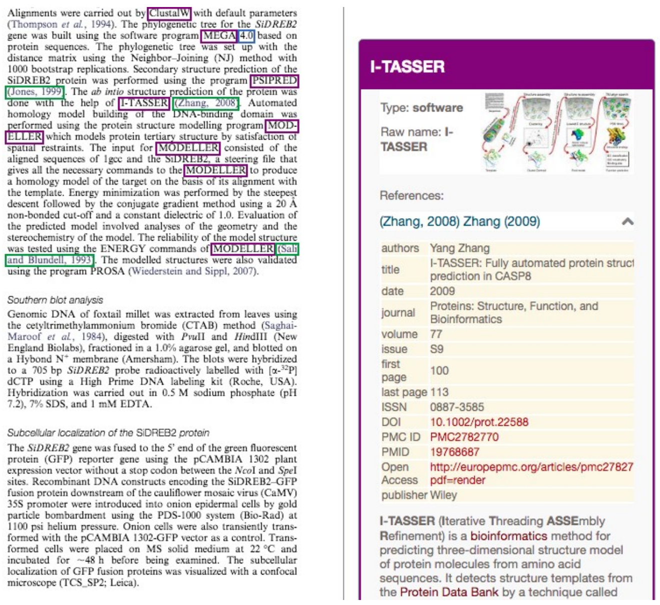
We applied the Softcite software recognizer with a SciBert-CRF model trained on the complete Softcite Dataset, to the CORD-19 publications [32], see Table 9. The CORD-19 corpus contains partial full-texts as JSON files. However, we started with the CORD-19 metadata file only and carried out a new and complete harvesting of the Open Access documents, in order to be able to use the PDF versions and more complete structured full-texts. Our full-text harvester is available online[12] and relies on the Unpaywall dataset [29] to identify URL of full-text papers based on metadata and DOI.

Our harvesting retrieved around 41K full-texts more than the official CORD-19 document set, allowing us to exploit more PDFs for extracting annotations coordinates and more accurate text structures. As indicated in Figure 1, PDF are processed via *pdfalto* and GROBID components while XML JATS files are processed by *Pub2TEI*[13], a collection of style sheets developed over 12 years able to transform a variety of publisher XML format to the same TEI XML format as produced by GROBID. This common format, which supersedes a dozen of publisher formats and many of their variants, centralizes further processing across PDF and heterogeneous XML sources.

Figure 3 shows how annotations can be visualized on the harvested PDF. When successful, entity linking allows users to access Wikidata and Wikipedia descriptions of the software. Similarly, when a bibliographical reference has been associated with a software in the document, users can visualize the metadata information of the publication, and, when available, directly access its Open Access full-text via Unpaywall.

A single 4-core server with one GPU (GeForce 1080Ti) and 16 GB RAM can process an average of 0.53 PDFs per second. The processing includes the full PDF processing, document structuring, the software recognition with the best SciBert-CRF model, entity disambiguation of the candidates software, additional document-level propagation and finally the resolution against 105M CrossRef records of possible bibliographical references attached to the software mentions. This runtime makes it possible to process 20 million articles (an estimate of the total available Open Access research articles) in one month with around 15 similar servers, a reasonable setting with today's scientific computing capacities.

[12] https://github.com/kermitt2/article-dataset-builder
[13] https://github.com/kermitt2/Pub2TEI



Alignments were carried out by ClustalW with default parameters (Thompson *et al.*, 1994). The phylogenetic tree for the *SiDREB2* gene was built using the software program MEGA 4.0 based on protein sequences. The phylogenetic tree was set up with the distance matrix using the Neighbor–Joining (NJ) method with 1000 bootstrap replications. Secondary structure prediction of the SiDREB2 protein was performed using the program PSIPRED (Jones, 1999). The *ab intio* structure prediction of the protein was done with the help of I-TASSER (Zhang, 2008). Automated homology model building of the DNA-binding domain was performed using the protein structure modelling program MODELLER which models protein tertiary structure by satisfaction of spatial restraints. The input for MODELLER consisted of the aligned sequences of 1gcc and the SiDREB2, a steering file that gives all the necessary commands to the MODELLER to produce a homology model of the target on the basis of its alignment with the template. Energy minimization was performed by the steepest descent followed by the conjugate gradient method using a 20 Å non-bonded cut-off and a constant dielectric of 1.0. Evaluation of the predicted model involved analyses of the geometry and the stereochemistry of the model. The reliability of the model structure was tested using the ENERGY commands of MODELLER (Sali and Blundell, 1993). The modelled structures were also validated using the program PROSA (Wiederstein and Sippl, 2007).

Southern blot analysis

Genomic DNA of foxtail millet was extracted from leaves using the cetyltrimethylammonium bromide (CTAB) method (Saghai-Maroof *et al.*, 1984), digested with *Pvu*II and *Hin*dIII (New England Biolabs), fractioned in a 1.0% agarose gel, and blotted on a Hybond N$^+$ membrane (Amersham). The blots were hybridized to a 705 bp *SiDREB2* probe radioactively labelled with [α-^{32}P] dCTP using a High Prime DNA labeling kit (Roche, USA). Hybridization was carried out in 0.5 M sodium phosphate (pH 7.2), 7% SDS, and 1 mM EDTA.

Subcellular localization of the SiDREB2 protein

The *SiDREB2* gene was fused to the 5' end of the green fluorescent protein (GFP) reporter gene using the pCAMBIA 1302 plant expression vector without a stop codon between the *Nco*I and *Spe*I sites. Recombinant DNA constructs encoding the SiDREB2–GFP fusion protein downstream of the cauliflower mosaic virus (CaMV) 35S promoter were introduced into onion epidermal cells by gold particle bombardment using the PDS-1000 system (Bio-Rad) at 1100 psi helium pressure. Onion cells were also transiently transformed with the pCAMBIA 1302-GFP vector as a control. Transformed cells were placed on MS solid medium at 22 °C and incubated for ~48 h before being examined. The subcellular localization of GFP fusion proteins was visualized with a confocal microscope (TCS_SP2; Leica).

I-TASSER

Type: **software**

Raw name: **I-TASSER**

References:

(Zhang, 2008) Zhang (2009)

authors	Yang Zhang
title	I-TASSER: Fully automated protein struct prediction in CASP8
date	2009
journal	Proteins: Structure, Function, and Bioinformatics
volume	77
issue	S9
first page	100
last page	113
ISSN	0887-3585
DOI	10.1002/prot.22588
PMC ID	PMC2782770
PMID	19768687
Open Access	http://europepmc.org/articles/pmc27827 pdf=render
publisher	Wiley

I-TASSER (**I**terative **T**hreading **ASSE**mbly **R**efinement) is a bioinformatics method for predicting three-dimensional structure model of protein molecules from amino acid sequences. It detects structure templates from the Protein Data Bank by a technique called

Figure 3: Augmented PDF using the Softcite text mining tool: mentions of software and their attributes are display on top of the PDF as HTML dynamic layout, via the standard PDF.js library (left part). The user can interact directly *in situ* with the annotations, opening info boxes with Wikidata disambiguated information and local consolidated bibliographical reference relevant to the software.

6 CONCLUSION

In this work, we addressed several practical challenges often neglected when applying state-of-the-art NLP techniques to scientific literature: processing of raw PDF documents, rich text, specialized vocabularies, extremely imbalance NER, and scalability requirements. In addition, we developed a "layout-aware" text mining approach for scientific documents, where layout information can be propagated throughout the entire pipeline, making possible for users to directly interact with annotations on PDF and access entity-level information. We think that this approach can enable novel applications to value research software as first-class scientific contributions and more generally better credit research software.

We are currently applying the Softcite software mention recognizer on a set of 15 million Open Access articles to create a Knowledge Base of research software in combination with curated resources[14]. We hope that this Knowledge Base and our data-driven approach will help to better understand the impact of software in scientific research, support reuse and collaboration around open source development, and ultimately benefit the dissemination of science and technologies across society.

ACKNOWLEDGMENTS

We would like to acknowledge the support of the Alfred P. Sloan Foundation, Grant/Award Number: 2016-7209, and of the Gordon and Betty Moore Foundation, Grant Number: 8622.

[14] https://github.com/softcite/softcite_kb

REFERENCES

[1] K. Ahmad and S. Collingham. 1996. *POINTER Project Final Report.* Technical Report. University of Surrey. http://www.computing.surrey.ac.uk/ai/pointer/report.

[2] Abbas Akkasi, Ekrem Varoğlu, and Nazife Dimililer. 2018. Balanced undersampling: a novel sentence-based undersampling method to improve recognition of named entities in chemical and biomedical text. *Applied Intelligence* 48, 8 (01 Aug 2018), 1965–1978. https://doi.org/10.1007/s10489-017-0920-5

[3] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A Pretrained Language Model for Scientific Text. arXiv:1903.10676 [cs.CL]

[4] Roi Blanco, Giuseppe Ottaviano, and Edgar Meij. 2015. Fast and Space-Efficient Entity Linking in Queries. In *Proceedings of the Eight ACM International Conference on Web Search and Data Mining* (Shanghai, China) *(WSDM 15)*. ACM, 10 pages.

[5] C Boettiger, S Chamberlain, E Hart, and K Ram. 2015. Building Software, Building Community: Lessons from the rOpenSci Project. *Journal of Open Research Software* 3, 1 (2015), e8. https://doi.org/10.5334/jors.bu

[6] Franck Dernoncourt, Ji Young Lee, and Peter Szolovits. [n.d.]. NeuroNER: an easy-to-use program for named-entity recognition based on neural networks. ([n. d.]). arXiv:1705.05487 http://arxiv.org/abs/1705.05487

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]

[8] Caifan Du, Johanna Cohoon, Patrice Lopez, and James Howison. 2021. Softcite dataset: A dataset of software mentions in biomedical and economic research publications. *Journal of the Association for Information Science and Technology* (2021). https://doi.org/10.1002/asi.24454

[9] Geraint Duck, Aleksandar Kovacevic, David L Robertson, Robert Stevens, and Goran Nenadic. 2015. Ambiguity and variability of database and software names in bioinformatics. *Journal of biomedical semantics* 6, 1 (2015), 29.

[10] Geraint Duck, Goran Nenadic, Michele Filannino, Andy Brass, David L Robertson, and Robert Stevens. 2016. A survey of bioinformatics database and software usage through mining the literature. *PloS one* 11, 6 (2016).

[11] Daniel Garijo, Maximiliano Osorio, Deborah Khider, Varun Ratnakar, and Yolanda Gil. 2019. OKG-Soft: An Open Knowledge Graph with Machine Readable Scientific Software Metadata. In *2019 15th International Conference on eScience (eScience)*. IEEE, San Diego, CA, USA, 349–358. https://doi.org/10.1109/eScience.2019.00046

[12] Martin Gerner, Goran Nenadic, and Casey M. Bergman. [n.d.]. LINNAEUS: A species name identification system for biomedical literature. 11, 1 ([n. d.]), 85. https://doi.org/10.1186/1471-2105-11-85

[13] Maryam Habibi, Leon Weber, Mariana Neves, David Luis Wiegandt, and Ulf Leser. [n.d.]. Deep learning with word embeddings improves biomedical named entity recognition. 33, 14 ([n. d.]), i37–i48. https://doi.org/10.1093/bioinformatics/btx228

[14] James Howison and Julia Bullard. 2016. Software in the Scientific Literature: Problems with Seeing, Finding, and Using Software Mentioned in the Biology Literature. *Journal of the Association for Information Science and Technology* 67, 9 (2016), 2137–2155. https://doi.org/10.1002/asi.23538

[15] Daniel S. Katz, Daina Bouquin, Neil P. Chue Hong, Jessica Hausman, Catherine Jones, Daniel Chivvis, Tim Clark, Mercè Crosas, Stephan Druskat, Martin Fenner, Tom Gillespie, Alejandra Gonzalez-Beltran, Morane Gruenpeter, Ted Habermann, Robert Haines, Melissa Harrison, Edwin Henneken, Lorraine Hwang, Matthew B. Jones, Alastair A. Kelly, David N. Kennedy, Katrin Leinweber, Fernando Rios, Carly B. Robinson, Ilian Todorov, Mingfang Wu, and Qian Zhang. 2019. Software Citation Implementation Challenges. *arXiv:1905.08674 [cs]* (May 2019). http://arxiv.org/abs/1905.08674 arXiv: 1905.08674.

[16] Frank Krüger and David Schindler. 2020. A Literature Review on Methods for the Extraction of Usage Statements of Software and Data. *Computing in Science Engineering* 22, 1 (Jan. 2020), 26–38. https://doi.org/10.1109/MCSE.2019.2943847

[17] John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. (2001).

[18] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. arXiv:1603.01360 [cs.CL]

[19] Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical Very Large Scale CRFs. In *Proceedings the 48th Annual Meeting of the Association for Computational Linguistics (ACL)* (Uppsala, Sweden). Association for Computational Linguistics, 504–513. http://www.aclweb.org/anthology/P10-1052

[20] Jaebeen Lee and Lea Deleris. 2020. Sequencing, Combining and Sampling Classifiers to Help Find Needles in Haystacks. In *24th European Conference on Artificial Intelligence, ECAI* (Santiago de Compostela, Spain).

[21] Joffrey L. Leevy, Taghi M. Khoshgoftaar, Richard A. Bauder, and Naeem Seliya. 2018. A survey on addressing high-class imbalance in big data. *Journal of Big Data* 5, 1 (01 Nov 2018), 42. https://doi.org/10.1186/s40537-018-0151-6

[22] Patrice Lopez. 2009. GROBID: Combining automatic bibliographic data recognition and term extraction for scholarship publications. In *International conference on theory and practice of digital libraries*. Springer, 473–474.

[23] Patrice Lopez. 2017. entity-fishing. In *WikiDataCon*. Berlin, Germany. https://upload.wikimedia.org/wikipedia/commons/5/50/Entity-fishing.pdf

[24] David Milne and Ian H. Witten. 2013. An open-source toolkit for mining Wikipedia. *Artificial Intelligence* 194 (2013), 222–239. https://doi.org/10.1016/j.artint.2012.06.007 Artificial Intelligence, Wikipedia and Semi-Structured Resources.

[25] Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing. In *Proceedings of the 18th BioNLP Workshop and Shared Task*. Association for Computational Linguistics, Florence, Italy, 319–327. https://doi.org/10.18653/v1/W19-5034

[26] J.M. Nicholson, M. Mordaunt, P. Lopez, A. Uppala, D. Rosati, N.P. Rodrigues, P. Grabitz, and S.C. Rife. 2021. scite: a smart citation index that displays the context of citations and classifies their intent using deep learning. *bioRxiv* (2021). https://doi.org/10.1101/2021.03.15.435418

[27] IB Ozyurt, JS Grethe, ME Martone, and AE Bandrowski. 2016. Resource Disambiguator for the Web: Extracting Biomedical Resources and their Citations from the Scientific Literature. *PLoS ONE* 11, 1 (2016), e0146300. https://doi.org/10.1371/journal.pone.0146300

[28] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*. 2227–2237.

[29] Heather Piwowar, Jason Priem, and Richard Orr. 2019. The Future of OA: A large-scale analysis projecting Open Access publication and readership. *bioRxiv* (2019). https://doi.org/10.1101/795310 arXiv:https://www.biorxiv.org/content/early/2019/10/09/795310.full.pdf

[30] Tim Rocktäschel, Michael Weidlich, and Ulf Leser. [n.d.]. ChemSpot: a hybrid system for chemical named entity recognition. 28, 12 ([n. d.]), 1633–1640. https://doi.org/10.1093/bioinformatics/bts183

[31] Katrin Tomanek and Udo Hahn. 2009. Reducing class imbalance during active learning for named entity annotation. In *Proceedings of the fifth international conference on Knowledge capture*. 105–112.

[32] Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Darrin Eide, Kathryn Funk, Rodney Michael Kinney, Ziyang Liu, William. Merrill, Paul Mooney, Dewey A. Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex D. Wade, Kuansan Wang, Christopher Wilhelm, Boya Xie, Douglas M. Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. 2020. CORD-19: The Covid-19 Open Research Dataset. *ArXiv* (2020).

[33] David Westergaard, Hans-Henrik Stærfeldt, Christian Tønsberg, Lars Juhl Jensen, and Søren Brunak. 2017. Text mining of 15 million full-text scientific articles. (jul 2017). https://doi.org/10.1101/162099

[34] Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Zero-shot Entity Linking with Dense Entity Retrieval. In *EMNLP*.