






Investigating Software Usage in the Social Sciences: A Knowledge Graph Approach

David Schindler¹ , Benjamin Zapilko² , and Frank Krüger¹ 

¹ Institute of Communications Engineering, University of Rostock, Rostock, Germany
{david.schindler, frank.krueger}@uni-rostock.de
<https://www.int.uni-rostock.de/>

² GESIS - Leibniz Institute for the Social Sciences, Cologne, Germany
Benjamin.Zapilko@gesis.org
<https://www.gesis.org/>

Abstract. Knowledge about the software used in scientific investigations is necessary for different reasons, including provenance of the results, measuring software impact to attribute developers, and bibliometric software citation analysis in general. Additionally, providing information about whether and how the software and the source code are available allows an assessment about the state and role of open source software in science in general. While such analyses can be done manually, large scale analyses require the application of automated methods of information extraction and linking. In this paper, we present SoftwareKG—a knowledge graph that contains information about software mentions from more than 51,000 scientific articles from the social sciences. A silver standard corpus, created by a distant and weak supervision approach, and a gold standard corpus, created by manual annotation, were used to train an LSTM based neural network to identify software mentions in scientific articles. The model achieves a recognition rate of .82 F-score in exact matches. As a result, we identified more than 133,000 software mentions. For entity disambiguation, we used the public domain knowledge base DBpedia. Furthermore, we linked the entities of the knowledge graph to other knowledge bases such as the Microsoft Academic Knowledge Graph, the Software Ontology, and Wikidata. Finally, we illustrate, how SoftwareKG can be used to assess the role of software in the social sciences.

Keywords: Software in science · Scientific articles · Information extraction · Knowledge graph

1 Introduction

Software is used during the entire research life-cycle and thus has significant influence on the research and its results. Knowledge about the software that was used during a scientific investigation is of interest for various reasons [11],

for instance to track the impact of software to attribute its developers, to analyze citation patterns, or to assess provenance information with respect to the research workflow. This is of particular interest, as software might contain issues that can affect the scientific results, as for instance reported by [4, 30, 31]. Moreover, research often relies on closed source software which is often not fully validated [25], eventually creating uncertainty about the reliability of the results.

Recently, software citation standards [27] have gained increasing interest in the scientific community but are not consistently used, which hampers the identification of such information on a large scale. Moreover, researchers are surprisingly creative when it comes to spelling variations of the actual software name (see Table 6). Named entity recognition (NER) provides a convenient method to identify entities in textual documents and could thus be employed to extract software mentions from scientific articles. The objective is to identify all software with an assigned name while ignoring unspecific statements such as ‘custom script’ or ‘custom code’. We employed an LSTM based approach to NER, which was trained using transfer learning based on distant and weak supervision and a small corpus of manually annotated articles [26]. The resulting information was then structured, interlinked, and represented in a knowledge graph which enables structured queries about software mentions in and across scientific publications. By exploiting this information as a knowledge graph, we follow W3C recommendations and best practices [7].

In this paper, we present SoftwareKG, a knowledge graph that links 51,165 scientific articles from the social sciences to software that was mentioned within those articles. SoftwareKG is further curated with additional information about the software, such as the availability of the software, its source and links to other public domain knowledge graphs. Using this information and exploiting additional information via links to other knowledge graphs, SoftwareKG provides the means to assess the current state of software usage, free and open source software in particular, in the social sciences. Links to other knowledge bases play an important role since additional information about software and scientific articles can be accessed which is not available directly from the article. All software and data associated with SoftwareKG is publicly available [26].

The remainder of this paper is structured as follows: First we describe how the information from articles was extracted, curated and structured. Afterwards, we provide a brief description of SoftwareKG including entity and relation statistics. We then discuss potential error sources and illustrate how SoftwareKG could be employed for the analysis of software usage in the social sciences. Finally, we discuss related work, summarize, conclude and lay out potential further work.

2 Document Selection and Corpus Generation

2.1 Gold Standard Corpus Generation

The gold standard corpus (GSC) was created by randomly selecting 500 articles from PLoS¹ using the keyword “Social Science”. All articles were scanned

¹ <https://www.plos.org/>.

Table 1. Overview of the GSC.

GSC statistics		Most frequent software	
# Sentences	31,915	R	77
# Annotations	1380	SPSS	60
# Distinct	599	SAS	44
Train	847 (+1005)	Stata	41
Devel	276	MATLAB	38
Test	257	Matlab	28

for Methods & Materials (M&M) sections, based on the assumption that those sections contain most of the software usage statements [2]. From the initial set of 500 articles, M&M sections were found in and extracted from 480, which then served as a base for the GSC. The remaining articles did not contain a M&M or similar section and were thus omitted. Ground truth annotation was performed by seven annotators using the BRAT v1.3 [28] web based annotation tool. Annotators were instructed to label all software usage statements, excluding any version or company information (see Example 1). Inter-rater reliability was assessed on 10% of all sentences and showed almost perfect agreement [14] (Cohen’s $\kappa = .82$). The GSC was then split into training, development and test sets with relative amounts of 60%, 20%, and 20%, respectively. The training set was extended by 807 sentences with software names to increase the amount of positive samples. The set of positive samples was retrieved by selecting sentences that contain at least one of the 10 most common software names of a previous analysis [2]. An overview of the resulting GSC is provided in Table 1. The GSC is publicly available [26].

2.2 Silver Standard Corpus Generation

Named entity extraction methods, in particular those relying on neural networks, require large amounts of training data to achieve reasonable recognition results. Annotated training data, however, is often unavailable and expensive to produce. The application of silver standard corpora [22] and transfer learning [24] has been shown to increase the recognition rates in such cases [5]. Silver standard corpora (SSCs) are annotated corpora that are not annotated by a manual process but rather provide “suggestive labels” created by employing distant or weak supervision [1]. In this work, we utilize the Snorkel data programming framework [21] which allows the specification of rules and dictionaries to provide such labels. The labeling rules are developed based on open knowledge bases and existing literature and generalize to other scientific domains even so they were optimized towards the social science corpus. Given those rules, Snorkel trains an unsupervised model for annotation by weighted combination of labeling rules by analyzing the correlations between the matches of the different rules. Finally, Snorkel provides scores to rank text candidates that were previously extracted from the text by using n-grams with a maximum length which was set to six

Table 2. Aliases for the software ‘Statistical Package for the Social Sciences’.

Wikidata category	Label
label (English)	Statistical Package for the Social Sciences
alias (English)	SPSS
alias (German)	PASW statistics; PASW
alias (French)	SPSS Inc.; PASW

tokens. The maximum length was determined from the GSC. As we optimize the recall, we include all candidates that exceeded the default scoring threshold of .5. In the following, the distant and weak supervision approach is highlighted in more detail.

Distant Supervision. Distant supervision uses external knowledge bases [17] to retrieve information about candidates of interest. As Wikidata² is recommended for distant supervision [29], we queried the knowledge graph for software names. To cover spelling variations and aliases for the different software, we considered various subcategories of software and different software types, and included all aliases from Wikidata’s “Also known as” attribute in the languages English, German, Spanish and French. The different languages were included because abbreviations may differ based on the authors’ language background even if the articles were written in English. We chose these languages as they represent the major languages in Wikipedia. An overview of Wikidata’s variations for SPSS is provided in Table 2. Variations are considered as potential candidates if they do not appear in the regular English dictionary. Using the English dictionary for exclusion of potential candidates was successfully used by Duck et al. [2].

Weak Supervision with Context Rules. In addition to the use of external knowledge bases, we implemented labeling functions based on the context of the software usage statements. To this end, we distinguished between general and exact context rules. The first examines the context of a candidate for special words or phrases indicating a software mention based on head word rules [2], while the latter implements the set of rules resulting from training an iterative bootstrapping for software identification [19]. The general rules employ information about the presence of particular tokens in the context of the candidates, such as ‘software’, ‘tool’ or ‘package’ or the presence of version numbers such as ‘v0.3’, ‘version 2’ or ‘2.0.12’. Furthermore, a rule is used that scans for the presence of the developer’s name in the context of the candidate. The identification of the potential candidate’s context was done after stop word removal. Part of Speech tags were employed for selecting from overlapping n-grams. The exact context rules that determine the context based on a specific pattern are based

² <https://www.wikidata.org/>.

on the literature [19]. Examples for the top two rules are: `use <> software` and `perform use <>` where the software position is marked by `<>`. Example 1 illustrates both the positive application of general context rule (dashed line) and the exact context rule (dotted line). The exact context rules are applied on the lemmatized context on the training set as in the original rules. Of the top 10 exact rules, the top 8 were used because the others did not extract any true positives.

Example 1. *We used SPSS software version 23 (SPSS Inc., Chicago, USA) for non-image-based statistical analyses and to compare volumes of subcortical structures.*

SSC Retrieval and Tagging. Snorkel’s generative model was used in its default configuration to generate the suggestive annotations and not further fine-tuned. As a final rule the most common false positive n-grams in the training corpus with no true positives were negatively weighted. To assess the quality of the suggestive labels, the Snorkel generative model was applied on both the training and development corpus and evaluated against the gold standard annotated labels with Snorkel’s internal evaluation. We trained the Snorkel model on the training set of our corpus and tested if adding further unlabeled data improves the results, since Snorkel is able to learn unsupervised. However, it was observed that the quality did not improve when adding up to five times the size of the original dataset. Overall, the Snorkel model achieved a precision of .33 (.32), a recall of .69 (.64) and an F-score of .45 (.42) on the development (training) set.

The articles for the SSC were obtained by retrieving all articles from PLoS for the keyword “Social Science” on 27th of August 2019. As for the GSC, the M&M sections were extracted and tagged by Snorkel’s model, resulting in a corpus of 51,165 labeled documents. In total 282,650 suggestive labels were generated for the entire corpus. Example 2 illustrates a correctly tagged new sample and one partially correct example where a redundant tag was inserted.

Example 2. *All statistical procedures were performed using IBM SPSS Statistics software version 22. Task accuracy and response times were analyzed using the SPSS software package (SPSS v17.0, Chicago, Illinois, USA).*

3 Extraction of Software Mentions

3.1 Model

For the extraction of software from scientific articles we used a bidirectional Long Short Term Memory Network [8] in combination with a Conditional Random Field Classifier [12] (bi-LSTM-CRF) derived from the description by Lample et al. [13]. This model achieves state of the art performance for Named Entity Recognition. We used a feature vector consisting of: 1) pretrained word-embeddings from scientific publications [20], and 2) bi-LSTM based character-embeddings. Word embeddings capture multi-level semantic similarities between

Table 3. Summary of the extraction model hyper-parameter settings.

Hyper-parameter	Setting
Word embedding size	200 (pre-trained, not trainable)
Character embedding size	50 (trainable)
Character LSTM size	25
Main LSTM size	100
Number of labels	3 ('O', 'B-software', 'I-software')

words [16] while character based features allow learning from the orthography of words directly. The input layer is followed by a bidirectional LSTM layer to consider the surrounding context of software mentions, a fully connected layer for classification, and a final CRF layer for the estimation of the most likely tagging sequence [13]. The model’s hyper-parameters are summarized in Table 3.

3.2 Training

As discussed above, training of the model was based on two different corpora, an SSC and a GSC. Sequential transfer learning [24] was employed to transfer information learned from the suggestive labels of the SSC to the GSC with the high quality labels to cope with the small amount of training data. For SSC training we selected all positive samples per epoch and used one negative sample per positive sample. We used different negative samples for each epoch until all were seen once. The number of pre-training epochs was optimized by training with up to 25 consecutive epochs on the SSC, after each of which we tested the performance to be expected by applying a standard, optimized re-training routine on the GSC training set. This procedure was selected due to the high computational requirements. For the SSC training, 2 epochs were found to provide the best basis. To optimize the SSC training, different learning and drop-out rates were considered. The best performing model was then selected for further optimization of the GSC training. All optimizations and evaluations were done with *rmsprop* to perform stochastic gradient descent.

During GSC training optimization, the following hyper-parameters were systematically considered: 1) *drop-out rate* in range of .3–.6, 2) *learning rate* in the range of .0001–.003, 3) *learning rate decay*, and 4) *sample weighting* adjusts the loss function label specific in a range of 0–.2 to increase the weight of positive samples. For GSC training we stop the training after 22 epochs. The final hyper-parameters for SSC and GSC training are provided in Table 4.

3.3 Evaluation and Extraction

To determine the expected quality of the final software mention identification, the selected model’s performance was evaluated in precision, recall and F-score on both, the development and the test set. We consider both precision and recall

Table 4. Hyper-parameter settings for the training process.

Hyper-parameter	SSC	GSC
Learning rate	.002	.0015
Learning decay	.0001 (linear)	.0007 (exponential)
Dropout rate	.5	.4
Sample weight	.1	.1
Epochs	2	22

Table 5. Overview of the recognition results of the software identification.

Evaluation	Training	Precision		Recall		F-score	
		test	dev	test	dev	test	dev
B-software	SSC	.21	(.21)	.72	(.70)	.32	(.32)
	GSC	.83	(.75)	.74	(.68)	.78	(.71)
	SSC→GSC	.86	(.83)	.85	(.78)	.86	(.80)
I-software	SSC	.36	(.31)	.68	(.35)	.47	(.33)
	GSC	.86	(.75)	.66	(.47)	.75	(.58)
	SSC→GSC	.76	(.77)	.82	(.61)	.79	(.68)
Partial match	SSC	.21	(.22)	.72	(.74)	.32	(.34)
	GSC	.85	(.75)	.76	(.69)	.80	(.72)
	SSC→GSC	.87	(.84)	.85	(.80)	.86	(.82)
Exact match	SSC	.20	(.20)	.68	(.64)	.30	(.30)
	GSC	.80	(.72)	.72	(.66)	.76	(.69)
	SSC→GSC	.83	(.81)	.82	(.78)	.82	(.79)

as highly important for the intended applications of the model. Precision allows, for instance, to give accurate impact measures while a high recall is beneficial for discovering rare, domain specific software. For testing on the development set the final model was trained on the training set alone while for testing on the test set the model was trained on both the training and development sets. This approach enables the estimation of the influence additional training data has. There are four relevant evaluation methods: 1) B-software: a match of the first token in a software name, 2) I-software: a match of all other words in a software name except for the first, 3) partial match: an overlap of the estimated and the true software name, and 4) exact match: the exact identification of the entire software name. To also assess the effect of the manually annotated data and the transfer learning, the following evaluations were performed: 1) SSC: the model resulting from SSC training only, 2) GSC: the model resulting from GSC training only, and 3) SSC→GSC: the final model resulting from transfer learning based on SSC and GSC training. The performance scores are summarized in Table 5.

Table 6. Overview of the most frequent spelling variations for SPSS.

Software mention	Frequency
SPSS	7068
Statistical Package for the Social Sciences	944
IBM SPSS statistics	875
Statistical Package for Social Sciences	784
IBM SPSS	480

For comparison we also tested a CRF based on a set of standard features³ on the GSC which achieved an F-Score of .41 (.36) with a precision of .30 (.24) and a recall .66 (.70) and on test (devel) set for exact recognition.

As we found an increase in the recognition performance from increasing the amount of training data for all evaluation metrics, the model used for information extraction was trained on the full GSC. It was then applied to all 51,165 M&M section from PLoS to identify software usage statements, resulting in 133,651 software names, 25,900 of which are unique. This seems plausible, as it reflects a similar frequency of mentions per article (2.6) as in the GSC (2.9).

3.4 Entity Disambiguation and Additional Information

Software names in scientific literature contain many variations in spelling and level of detail. This ranges from including the manufacturer’s name to using version information and different interpretations of software name abbreviations. Table 6 gives an overview of the most common spelling variations of the statistical software SPSS. In total, 179 different spellings for SPSS were identified, most of which were not in the GSC. This means the model is able to generalize to previously unknown software names.

To disambiguate the different spelling variations, a three part entity linking was employed to reflect software mention specific variations: 1) analysis of simple spelling variations based on mentions, 2) abbreviation based linking, and 3) exploitation of information from the DBpedia knowledge graph. Software mentions were normalized by case folding and removal of special characters such as numbers and Greek letters. Furthermore, syllables such as “pro” were removed from the end of software names and the remaining words were stemmed to match common variations such as ‘statistical’ and ‘statistic’. The result of this step was a transformed version of the software mention. To match the different interpretations of abbreviations, stop-words were removed and abbreviations were created from the first letters of the remaining words. The transformed name and the abbreviations were then used to cluster the first software names in the hierarchical linking pipeline.

For further disambiguation all software from DBpedia including *label*, *name*, *wikiPageRedirects* and *wikiPageDisambiguates* was retrieved and used to link the

³ <https://sklearn-crfsuite.readthedocs.io>.

software names and select unique names as previously suggested [29]. Here, programming languages were included and software of type “video game” excluded. As for the Wikidata query, the languages English, German, French and Spanish were considered. If available the developer of a software was also included. The actual linking was performed as follows:

- 1) labels from DBpedia were used to further group spelling variations,
- 2) aliases from all languages were used, and
- 3) if neither of the previous provided a match, a combination of label and developer was employed.

The representative name used in the final data model was then created from the DBpedia label of the software, if exists, or the most frequent matched name, otherwise. DBpedia entries were retrieved for 66,899 (1160 unique) of the software names. As a result of the entity disambiguation, the set of unique software names was reduced from 25,900 to 20,227.

If available, the following additional information was collected in a manual process for the most frequent software: 1) the corresponding identifier in the Software ontology, Wikidata, and Wikipedia, 2) the URL of the software, 3) the manufacturer of the software, 4) whether the software is freely available, 5) whether the source code of the software is freely available, and 6) the license that was used for the publication of the source code. Additional information was retrieved for 133 software products covering 67,477 software mentions, representing half of all software mentions.

4 SoftwareKG: Data Model and Lifting to RDF

SoftwareKG was generated from the extracted and additionally collected information. In particular, it contains metadata about the publications, the authors and the software used in the publications. The graph contains 3,998,194 triples, 1,013,216 resources which are represented in 5 distinct types and 25 distinct properties. The graph holds 51,165 *Publication* resources, 20,227 *Software* resources, 334,944 *Author* resources, and 473,229 *Organization* resources. In total, the graph contains 133,651 software mentions. The most frequent software mentioned in the papers are *SPSS* (11,145 mentions), *R* (11,102 mentions), and *STATA* (5,783 mentions). Of the included software, 75 software applications are available for free, while 58 are not free. For the remaining software, this information is missing. Similarly, the source code is available of 52 software applications.

The SoftwareKG data model exploits terms from established vocabularies, mostly from schema.org⁴. For designing the data model, established vocabularies have been reused as it is seen as best practice [7]. The model can easily be extended by further properties, e.g. by terms of the CodeMeta Project⁵ for describing software metadata. For some very specific properties, we had to define our own properties which are denoted in the model by the namespace `skg`.

⁴ <https://schema.org/>.

⁵ <https://codemeta.github.io/>.

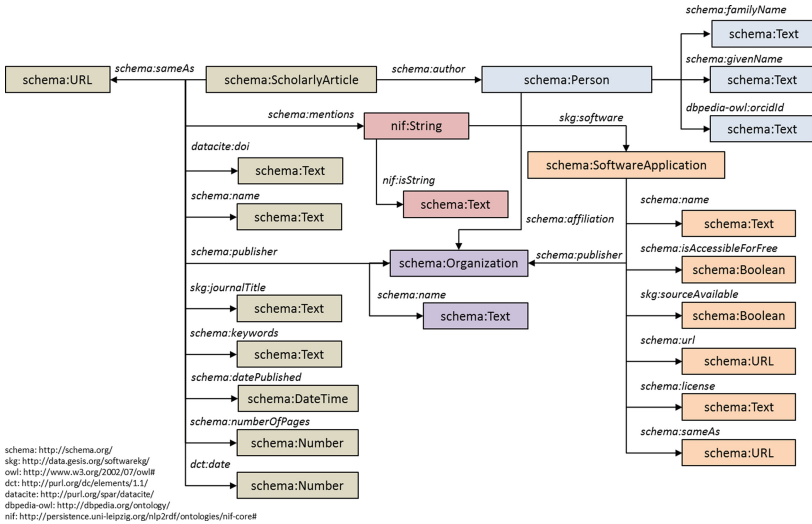


Fig. 1. Illustration of the data model

The core elements of the model are *Software*, *Publication*, *Author*, and *Organization* as shown in Fig. 1. The class *Software* represents a software application. It gathers properties for representing the name of the software application, the publisher, the homepage, the license under which the software has been published, whether it is available for free, and whether the source code is available or not. The property `schema:sameAs` gathers links to Wikipedia, Wikidata, the Software Ontology, and to DBpedia. The class *Publication* represents a scientific article. Here, we use properties to represent title, author(s), DOI, publisher, the publication date, and other metadata. We link to the same publication in the Microsoft Academic Graph by using `schema:sameAs`. The property `schema:mentions` captures the detected software mentions in a publication. Each mention is from the type `nif:String` which connects to the precise string in the paper (`nif:isString`) and to the meant *Software* (`skg:software`). The class *Author* represents each of the authors of a publication with his/her name, affiliation, and Orcid ID. Eventually, to the class *Organization* it is linked to by authors (as their affiliation), publications (as their publisher), and software (also as publisher).

All extracted and linked information is generated in JSON-LD following the data model. SoftwareKG is published under the Creative Commons BY 4.0 license. The KG can be accessed from a Virtuoso triple store with a SPARQL endpoint⁶ and is downloadable as a dump [26]. It can also be accessed through its official website⁷ which also contains statistics and a set of SPARQL queries.

⁶ <https://data.gesis.org/softwarekg/sparql>.

⁷ <https://data.gesis.org/softwarekg/site/>.

5 Use Cases and Exploitation

As initially stated, knowledge about the software employed in scientific investigations is important for several reasons. The previous sections describe how SoftwareKG, a knowledge graph about software mentions in scientific publication was created. This section illustrates the usage of SoftwareKG and how to leverage information from other knowledge graphs to perform analyses about software in science. The queries and the plots illustrating their results can also be found at the accompanying website.

The frequency of software mentions in scientific articles allow to assess the impact of individual software to science. Moreover, it allows to attribute the developers of such software. Figure 2 illustrates the frequencies of the 10 most common software per year in absolute numbers with respect to our corpus. The data was obtained by using the query in Listing 1.1. It can be seen that both, SPSS and R are predominantly used in the social sciences, reflecting their usage for statistical analyses. In general, statistical analysis software is the most frequently used type of software.

The availability of software used for the original analyses of scientific investigations plays a central role in its reproducibility. Moreover, the usage of open source software allows researchers to inspect the source code, reducing uncertainty about the reliability of the scientific analyses and the results [25]. Figure 3 illustrates the usage of free and open source software over time.

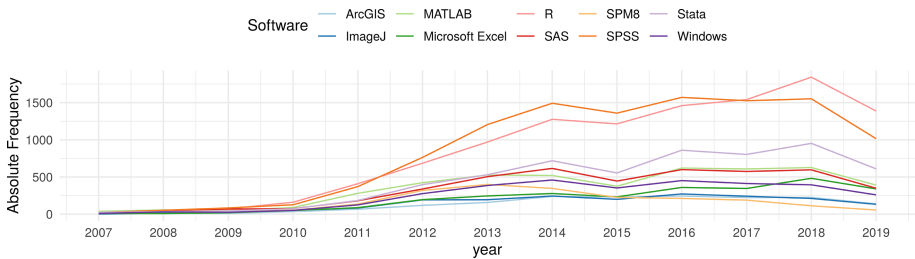


Fig. 2. Absolute amount of the 10 most common software per year.

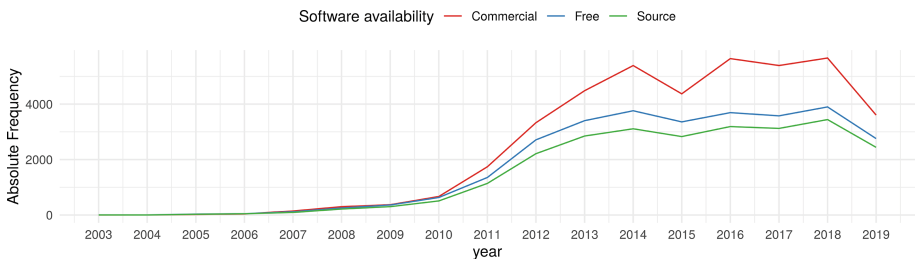


Fig. 3. Absolute frequency of the usages of commercial, free, and open source software.

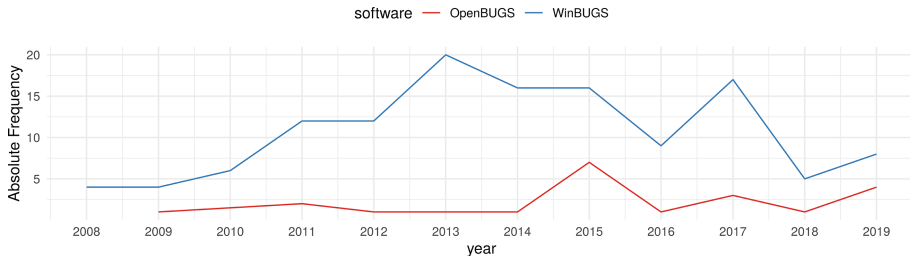


Fig. 4. Absolute frequency of the usages of OpenBUGS and WinBUGS.

```

select ?name ?year (count(?name) as ?count)
WHERE {
  ?s rdf:type <http://schema.org/SoftwareApplication> .
  ?s <http://schema.org/name> ?name .
  ?m <http://data.gesis.org/softwarekg/software> ?s .
  ?p <http://schema.org/mentions> ?m .
  ?p <http://purl.org/dc/elements/1.1/date> ?year .
}
GROUP BY ?name ?year
HAVING (count(?name) > 1)
ORDER BY DESC(?count)

```

Listing 1.1. SPARQL query to retrieve frequency of software mention per year

When the development of a software is discontinued, it is often replaced by another software. This is, for instance, the case with the free software WinBUGS, with the latest release from 2007, and the open source software OpenBUGS, where the development started in 2005. This relation can easily be retrieved via Wikidata using the *replaced-by*(P1366) relation. Through linking SoftwareKG with Wikidata we combined information about the replacement of software with knowledge about its usage in scientific articles which enables statements about when and how such transitions arrive in science. The query linking SoftwareKG and Wikidata is available on the SoftwareKG website and with the source code [26]. Figure 4 illustrates the frequencies of both the free software WinBUGS and the open source software OpenBUGS per year. From the plot, it can be seen that while the development of WinBUGS was discontinued more than 10 years ago, the successor did not replace it yet.

6 Related Work

Identifying software mentions in scientific publications is an open research problem and can serve several different purposes: 1) mapping of available software, 2) measuring the impact of software, and 3) analyzing how software is used, shared and cited in science. A more detailed overview of the problem and the

reasons can be found in [11]. Besides manual extraction, as for example done by Howison et al. [9] to investigate software citations and their completeness, automated extraction enables the analysis of software usage in a larger context. Greuel and Sperber [6] create a mapping of mathematical software through automated filtering of potential mentions and manual review. Duck et al. [2] use a fully automated, rule-based scoring system configured by hand and based on a dictionary of known software and databases which was later improved by applying machine learning onto the rule set to achieve .67 F-score in the domain of Bioinformatics [3]. Pan et al. [19] use iterative bootstrapping to learn common mentioning patterns and software names starting from an initial set of seed software entities achieving .58 F-score and use the extraction results to provide impact measures. While those investigations all deal with the extraction of software they are not concerned with entity linking and can therefore only argue about distinct software mentions.

Automatically generating knowledge graphs from information about scientific publications allows the analysis of scientific workflows and enable large scale meta-analyses. The Open Academic Graph⁸, for instance, captures metadata of scientific publications, while the Scholarlydata project [18] captures linked data about scientific conferences and workshops. Jaradeh et al. [10] create the Open Research Knowledge Graph which captures semantic information from scientific publications. They apply deep learning methods to capture information about process, method, material and data in publications and use DBpedia for entity linking. However, they do not provide a quantitative evaluation of their text mining approach. Recuperio et al. [23] employ different existing classifiers to extract knowledge from scientific articles, perform disambiguation between extracted targets and create a knowledge graph based on the gathered information. Luan et al. [15] create a knowledge graph from tasks, methods, metrics, materials, and other entities and their relations from scientific publications.

The work presented here is the first that creates a knowledge graph particularly tailored for analyses of software in science based on recognition performance that outperforms previous approaches and the first to include entity disambiguation for software mentions.

7 Limitations and Potential Sources of Uncertainty

Most of the results presented in this work are based on methods of machine learning and automatic analyses which mainly rely on the quality of the provided labeled corpus. For the entire pipeline we identified several sources of uncertainty, which might accumulate and may result in a bias for further analyses:

- The corpus was retrieved from PLoS by using the keyword “Social Science” potentially resulting in the following two issues: The employed keyword did not only result in articles from the social sciences, but also from the bio medicine and related research domains. On the other hand this aspect facilitates the transfer of the model to the domain of life sciences. Additionally, the

⁸ <https://www.openacademic.ai/oag/>.

PLoS corpus itself contains a bias towards the open access general purpose journal, which might not reflect the preferred publication target of researchers from the target domain.

- While the GSC annotation is of high quality in terms of inter-rater reliability the annotation task proved to be a complex task for the annotators, for instance when differentiating between algorithms and software with the same name. The absence of version information makes this decision even harder.
- The SSC is constructed with suggestive labels, with false positives that may be carried over to the final model. Indeed, we found examples, such as *Section* and *ELISA*, which both are software names, but also commonly appear in scientific publications without a connection to the software.
- The employed model achieves a high recognition performance. As suggested by the improvement of the test set evaluation, there is still potential to increase the performance by using a larger GSC.
- Reliably estimating the error of the entity disambiguation is difficult due to the absence of a ground truth. However, the method benefits from just working on extracted names which strongly restricts the chance of errors. We found some cases in which a linking to DBpedia was not possible because of multiple matches with DBpedia entries for which we could not automatically determine which match is the correct one.

In summary, SoftwareKG provides a reasonable quality in terms of knowledge identification, but automatic analyses should be carried out carefully.

8 Conclusion and Future Work

In this work we introduce SoftwareKG and a method for creating a large scale knowledge graph capturing information about software usage in the social science. It was generated by employing a bi-LSTM for the automatic identification of software mentions in the plain text of publications. The proposed method achieves a high recognition score of .82 F-score in terms of exact match, which is a strong improvement over the state of the art. By using transfer learning based on data programming with distant and weak supervision rules, the performance could be significantly improved. The proposed approach is the first to integrate entity linking for the disambiguation of software names and the first to construct a knowledge graph to facilitate reasoning by allowing running queries against the constructed graph. Additionally, the available information in our graph was enhanced by manual annotation to support further analyses regarding free and open source software where otherwise no analysis would be possible. To create a large scale basis for reasoning and illustrate how such a basis can be constructed we applied our method to construct a knowledge graph over all articles published by PLoS tagged with “Social Science”. Finally, we employed SoftwareKG to illustrate potential analyses about software usage in science.

Future work includes the automatic collection of additional information about the software such as the version or the source code repositories which implies an extension of the data model, e.g. by including properties of CodeMeta.

This enables the identification of the particular implementation by employing software preservation services such as SoftwareHeritage⁹.

Acknowledgements. This work was partially carried out at GESIS - Leibniz Institute for the Social Sciences and was financially supported by the GESIS research grant GG-2019-015 and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - SFB 1270/1 - 299150580.

References

1. Boland, K., Krüger, F.: Distant supervision for silver label generation of software mentions in social scientific publications. In: Proceedings of the BIRNDL 2019, Paris, France, July 2019. <http://ceur-ws.org/Vol-2414/paper3.pdf>
2. Duck, G., Nenadic, G., Brass, A., Robertson, D.L., Stevens, R.: bioNerDS: exploring bioinformatics' database and software use through literature mining. *BMC Bioinformatics* **14**(1), 194 (2013)
3. Duck, G., Nenadic, G., Filannino, M., Brass, A., Robertson, D.L., Stevens, R.: A survey of bioinformatics database and software usage through mining the literature. *PLoS ONE* **11**(6), e0157989 (2016)
4. Eklund, A., Nichols, T.E., Knutsson, H.: Cluster failure: why fMRI inferences for spatial extent have inflated false-positive rates. In: Proceedings of the National Academy of Sciences, p. 201602413 (2016)
5. Giorgi, J.M., Bader, G.D.: Transfer learning for biomedical named entity recognition with neural networks. *Bioinformatics* **34**(23), 4087–4094 (2018). <https://doi.org/10.1093/bioinformatics/bty449>
6. Greuel, G.-M., Sperber, W.: swMATH – an information service for mathematical software. In: Hong, H., Yap, C. (eds.) ICMS 2014. LNCS, vol. 8592, pp. 691–701. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44199-2_103
7. Heath, T., Bizer, C.: Linked data: evolving the web into a global data space. *Synth. Lect. Semant. Web Theory Technol.* **1**(1), 1–136 (2011)
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997)
9. Howison, J., Bullard, J.: Software in the scientific literature: problems with seeing, finding, and using software mentioned in the biology literature. *J. Assoc. Inf. Sci. Technol.* **67**(9), 2137–2155 (2016)
10. Jaradeh, M.Y., et al.: Open research knowledge graph: next generation infrastructure for semantic scholarly knowledge. In: Proceedings of the K-Cap, pp. 243–246. ACM (2019)
11. Krüger, F., Schindler, D.: A literature review on methods for the extraction of usage statements of software and data. *IEEE Comput. Sci. Eng.* (2019). <https://doi.org/10.1109/MCSE.2019.2943847>
12. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning, ICML 2001, pp. 282–289. Morgan Kaufmann Publishers Inc., San Francisco (2001)
13. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. arXiv preprint [arXiv:1603.01360](https://arxiv.org/abs/1603.01360) (2016)
14. Landis, J.R., Koch, G.G.: The measurement of observer agreement for categorical data. *Biometrics* **33**(1), 159–174 (1977)

⁹ <https://www.softwareheritage.org/>.

15. Luan, Y., He, L., Ostendorf, M., Hajishirzi, H.: Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In: Proceedings of the EMNLP (2018)
16. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
17. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2, pp. 1003–1011. Association for Computational Linguistics (2009)
18. Nuzzolese, A.G., Gentile, A.L., Presutti, V., Gangemi, A.: Conference linked data: the scholarlydata project. In: Groth, P., et al. (eds.) ISWC 2016. LNCS, vol. 9982, pp. 150–158. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46547-0_16
19. Pan, X., Yan, E., Wang, Q., Hua, W.: Assessing the impact of software on science: a bootstrapped learning of software entities in full-text papers. *J. Informetr.* **9**(4), 860–871 (2015)
20. Pyysalo, S., Ginter, F., Moen, H., Salakoski, T., Ananiadou, S.: Distributional semantics resources for biomedical text processing. In: Proceedings of LBM 2013 (2013)
21. Ratner, A.J., Bach, S.H., Ehrenberg, H.R., Ré, C.: Snorkel: fast training set generation for information extraction. In: Proceedings of the 2017 ACM International Conference on Management of Data, pp. 1683–1686. ACM (2017)
22. Rebholz-Schumann, D., et al.: CALBC silver standard corpus. *J. Bioinform. Comput. Biol.* **08**(01), 163–179 (2010). <https://doi.org/10.1142/s0219720010004562>
23. Buscaldi, D., Dessì, D., Motta, E., Osborne, F., Reforgiato Recupero, D.: Mining scholarly publications for scientific knowledge graph construction. In: Hitzler, P., et al. (eds.) ESWC 2019. LNCS, vol. 11762, pp. 8–12. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32327-1_2
24. Ruder, S.: Neural transfer learning for natural language processing. Ph.D. thesis, National University of Ireland, Galway (2019)
25. Russo, D., Voigt, C.C.: The use of automated identification of bat echolocation calls in acoustic monitoring: a cautionary note for a sound analysis. *Ecol. Ind.* **66**, 598–602 (2016). <https://doi.org/10.1016/j.ecolind.2016.02.036>
26. Schindler, D., Zapolko, B., Krüger, F.: SoftwareKG (1.0), March 2020. <https://doi.org/10.5281/zenodo.3715147>
27. Smith, A.M., Katz, D.S., Niemeyer, K.E.: Software citation principles. *PeerJ Comput. Sci.* **2**, e86 (2016). <https://doi.org/10.7717/peerj-cs.86>
28. Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S., Tsujii, J.: 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 ACL, pp. 102–107. ACL (2012)
29. Weichselbraun, A., Kuntschik, P., Brasoveanu, A.M.: Name variants for improving entity discovery and linking. In: Proceedings of the LDK 2019. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2019)
30. Zeeberg, B.R., et al.: Mistaken identifiers: gene name errors can be introduced inadvertently when using excel in bioinformatics. *BMC Bioinformatics* **5**(1), 80 (2004). <https://doi.org/10.1186/1471-2105-5-80>
31. Ziemann, M., Eren, Y., El-Osta, A.: Gene name errors are widespread in the scientific literature. *Genome Biol.* **17**(1) (2016). <https://doi.org/10.1186/s13059-016-1044-7>