Graph-TempCZ: A Graph Representation of Software Mentions for Predicting Software Usage in Scientific Publications

Congfeng Cao[1], Pengyu Zhang[2], Jelke Bloem[1]
[1]Institute for Logic, Language and Computation, University of Amsterdam
[2]INDE Lab, University of Amsterdam
{c.cao, p.zhang, j.bloem}@uva.nl

Abstract

Predicting how software is used, shared, and evolves across publications is essential to studying scientific progress. Existing methods for representing software usage in publications rely mainly on tabular or textual formats, which limit their structural expressiveness and consequently their ability to predict software usage. We address these gaps by representing software mentions and citations as a graph and formulating software usage prediction as a link prediction task. To support this study, we construct the first large-scale graph dataset of publication and software mentions, **Graph-TempCZ**, covering 1959-2022 with over six million mention relationships. Experiments using both traditional machine learning and Graph Neural Network (GNN) show that graph-based models substantially outperform feature-based baselines, achieving a 5.98% improvement in test accuracy. Temporal experiments further reveal that models trained on one year generalize effectively to nearby years but show gradual performance decay as the temporal gap increases. This work provides the first comprehensive foundation for analyzing software usage through a temporal graph representation. All code and data are publicly available to support further research on the project page of Graph-TempCZ

Keywords: Software Usage, Graph Learning, Temporal Link Prediction, Bibliometrics

1. Introduction

Software plays a vital role across the research life cycle. From data collection and analysis to visualization and publication, software underpins nearly every stage of modern scientific research (Goble, 2014; Zhang et al., 2024). Therefore, structured representation and prediction of how software is used, shared, and evolves across publications has become an essential part of studying scientific progress and technological innovation. For example, Mullick et al. (2022) employ a sentence-level classifier to extract entities such as software and code from materials science publications. However, current methods for representing and extracting software usage from publications lack structural and temporal expressiveness, thereby limiting their ability to predict software usage.

Lack of a graph representation. Existing representations of software usage mainly rely on tabular data, such as listing software that appears in publications as records (Yang et al., 2018; Zhao and Wei, 2017), or on text-level co-occurrence, which captures that software names are mentioned within sentences and text of publications (Pan et al., 2015; Tateisi et al., 2016). These methods derive from viewing publications and software as flat tabular data or as linguistic co-occurrences extracted through natural language processing methods. While informative, they overlook the underlying structure of how publications and software are interconnected. In practice, software usage naturally forms a network in which publications cite or mention software, similar to publication citation networks (Del Gratta et al., 2016; Singha Roy and Mercer, 2024). Without a graph representation, it is difficult to capture indirect relations, such as shared dependencies or joint usage across research domains, which limits our ability to predict software usage as an interconnected network.

Lack of a temporal representation. Software usage is inherently continuous and time dependent, as software is repeatedly mentioned in publications across years when it is used, updated, and cited in new research (Wang and Li, 2024; Pan et al., 2018). However, existing methods ignore this temporal dimension, treating software usage as static rather than evolving over time. Without modeling temporal patterns, it becomes impossible to analyze how software adoption changes or to predict future usage trends, which are essential for understanding the development of research software.

As shown in Figure 1, we address these challenges by representing software mentions and citations as graphs. In the graph, the link prediction task aims to build the relationships between nodes, that is, the citation and mention links between software and publications in publication-software citation and mention graphs. This is particularly relevant when the software is missing or improperly cited in publications. Therefore, we perform a link prediction task to predict software usage in publications and further apply temporal link prediction to predict software usage in publications over different

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 7791–7803
11–16 May 2026.

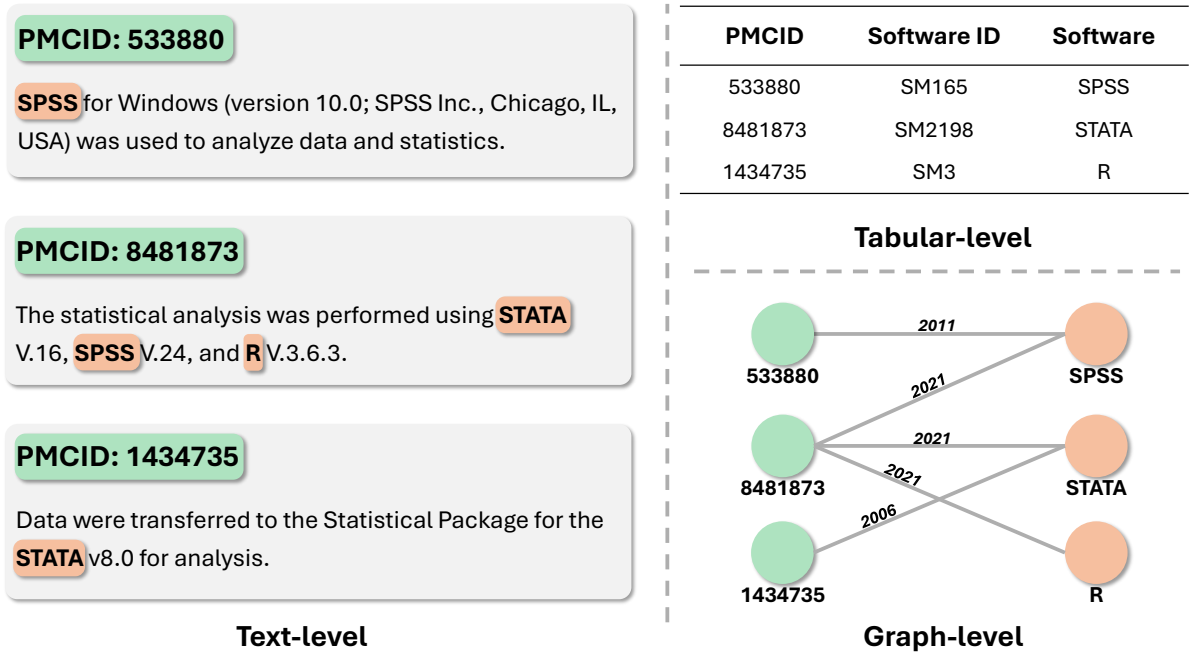

Figure 1: Overview of representation of software usage. **Text-level** represent where software names are mentioned within sentences of publications. The `PMCID` is a unique identifier assigned to each publication accepted into PubMed Central. The `Software ID` is a unique identifier assigned to each software. **Tabular-level** representation represents publications and software in a two-dimensional table, listing their respective attributes such as metadata and usage statistics. **Graph-level** representation represents the citation or mention relationships between publications and software, where nodes contain attribute information and edges encode citation or mention links.

time periods.

To support this study, we construct the first large-scale graph dataset of publication-software mentions, namely **Graph-TempCZ**, which captures mention relationships between publications and software. The dataset covers publications from 1959 to 2022 and contains $6{,}569{,}773$ relationships, $1{,}460{,}616$ publication nodes, and $893{,}433$ software nodes. Building on **Graph-TempCZ**, we further construct a series of yearly subgraphs from 2010 to 2020 to facilitate temporal link prediction. Each yearly subgraph represents the publication-software mention relationships observed in that specific year, forming a continuous decade-long sequence of graphs. This design enables a systematic examination of how software usage evolves over time and how models trained on one year generalize to others.

Finally, we evaluate link prediction performance on **Graph-TempCZ** using traditional machine learning and Graph Neural Network (GNN) methods. Both traditional machine learning and GNN methods demonstrate predictive capabilities, with GNN-based models achieving the highest test accuracy of 92.88%, which represents a 5.98% improvement over traditional baselines. Furthermore, based on temporal link prediction, we show that software usage can be effectively predicted from historical data, revealing clear temporal effects.

Our main contributions are as follows:

- We introduce a graph-based representation for representing software mentions in publications, revealing latent structural relationships between publications and software that were overlooked in prior representations. Building on this representation, we are the first to formulate software usage prediction as a link prediction task and compare traditional and graph learning methods, finding that graph-based models, particularly Graph Neural Network (GNN) models, achieve a test accuracy of 92.88%, representing a 5.98% improvement over traditional baselines.
- We apply temporal link prediction to show that future software usage can be effectively predicted from historical data. Models trained on a given year achieve the best results when training and testing years coincide, and generalize well to neighboring years, with performance gradually decreasing as the temporal gap widens.
- We construct a large-scale temporal

publication-software mention graph dataset, **Graph-TempCZ**, with rich publication metadata, covering publications from 1959 to 2022 and comprising $6{,}569{,}773$ relationships, $1{,}460{,}616$ publication nodes, and $893{,}433$ software nodes. To the best of our knowledge, the stratified subgraphs from 2010 to 2020 constitute the first large-scale and long-range dataset dedicated to studying the temporal dynamics of software usage.

2. Related Work

2.1. Datasets for Software Usage in Publications.

Several efforts have produced datasets of software mentions in scientific publications. A prominent example is the SoftCite dataset, a gold-standard corpus derived from the manual annotation of 4,971 research articles in biomedicine and economics (Du et al., 2021). SoftCite provides thousands of labeled software mentions along with detailed attributes such as version and creator information.

SoMeSci (Software Mentions in Science) is another corpus that compiles 3,756 software mentions annotated in 1,367 PubMed Central articles as a knowledge graph (Schindler et al., 2021). SoMeSci enriches each mention with metadata (e.g., versions, URLs, developers) and categorizes the type of software and its context of use.

SoftwareKG is a knowledge graph containing information about software mentions extracted from more than 51,000 scientific articles in the social sciences, identifying over 133,000 software mentions (Schindler et al., 2020).

The CZ Software Mentions dataset extracts 1.12 million unique software mention strings from 2.4 million papers in the NIH PMC-OA Commercial subset, 481k unique mentions from the NIH PMC-OA Non-Commercial subset (both gathered in October 2021), and 934k unique mentions from 3 million papers in the Publishers' collection (Istrate et al., 2022).

Despite these advances, existing datasets have notable limitations that our work aims to address. Many corpora are restricted in scope or size, for example, SoftCite and SoMeSci focus on biomedicine and cover only a few thousand articles (Schindler et al., 2021). While their manual annotations are of high quality, they cannot easily scale to millions of publications. SoftwareKG is also limited by its scale, disambiguation quality, and lack of publication metadata. These limitations also hinder the training of effective mention prediction models that can generalize across scales and time.

Our work tackles these gaps by constructing a large-scale, cross-domain graph of publication-software mentions and by integrating temporal information. The resulting dataset, Graph-TempCZ, spans multiple fields and decades, enabling network-centric analyses of software usage that move beyond static, single-domain snapshots.

2.2. Analysis of Software Usage in Scientific Research.

A growing body of research analyzes how software is used and cited in scholarly work, revealing that the role of software in research is widespread but still underexplored. Howison and Bullard (2016) conducted one of the earliest qualitative studies and found that scientists frequently mention software informally in publications, yet proper scholarly citations are often missing. Only between 31% and 43% of mentions involve formal citations, while informal mentions are common even in high-impact journals and across different types of software. Park and Wolfram (2019) further showed that research software is rarely included in formal indexing systems such as Clarivate's Data Citation Index, indicating that software reuse is seldom documented or visible in citation databases.

Orduña-Malea and Costas (2021) proposed a link-based webometric approach to characterize online mentions of scientific software, using VOSviewer as a case study across different analytical frameworks. Schindler et al. (2022) analyzed the evolution of software usage and citation patterns across various fields, journal ranks, and publication impacts based on a software knowledge-graph dataset.

Existing studies of software usage remain largely retrospective and descriptive. Manual content analyses often cover limited samples, and most quantitative analyses rely on aggregated statistics such as counting mentions or co-occurrences, which overlook the underlying graph structure linking papers and software. Moreover, temporal aspects are seldom modeled: previous works typically describe historical growth trends but do not attempt to predict future software usage.

In contrast, our work adopts a graph-based representation. We treat publications and software as nodes in a network and learn the link structure between them. This approach captures indirect relationships such as shared dependencies and enables temporal link prediction, moving from static descriptive analyses toward a connected, time-aware understanding of software in science.

2.3. Link Prediction in Scientific Networks.

Link prediction is a fundamental problem in network science with broad applications to scholarly

data, such as predicting new collaborations or citation links. Classical link prediction methods include heuristic indices such as Common Neighbors (CN), Adamic-Adar (AA), and the Jaccard coefficient (Zhang and Chen, 2018). While simple and interpretable, these measures rely on fixed assumptions and often fail to generalize to heterogeneous or evolving graphs.

Advances in Graph Neural Networks (GNN) have greatly enhanced link prediction. Models such as GraphSAGE (Hamilton et al., 2017b) and Graph Attention Networks (GAT) (Veličković et al., 2018) learn node embeddings through neighbor aggregation or attention mechanisms, capturing structural dependencies. However, several challenges remain when applying link prediction to scientific knowledge graphs, particularly those involving heterogeneous node types and temporal dynamics.

TGAT (da Xu et al., 2020) introduces time encoding mechanisms for dynamic graphs, while TGN (Rossi et al., 2020) incorporates memory modules to update node states over time. Zaporojets et al. (2022) investigate how temporal evolution affects performance in an entity linking task. They introduce TempEL, an entity linking dataset constructed from time-stratified English Wikipedia snapshots covering the years 2013 to 2022, and demonstrate that models experience noticeable temporal performance degradation over time.

Our work extends temporal link prediction to the domain of software usage by integrating GNN-based architectures with large-scale, time-sliced publication-software graphs. We systematically evaluate both static and temporal predictors across multiple year pairs, revealing temporal generalization patterns and providing insights into how graph-based learning can forecast evolving software adoption trends.

3. Dataset Construction

As shown in Figure 2, our data processing pipeline consists of three main stages: publication metadata augmentation, graph construction, and temporal graph construction.

3.1. The CZ Software Mentions Dataset

Our dataset is constructed based on the disambiguated part of the CZ Software Mentions dataset (Istrate et al., 2022). The CZ Software Mentions dataset is a large-scale collection of software mentions extracted from millions of biomedical papers using a SciBERT-based NER model, providing context, metadata, and links for each mention. The disambiguated part groups diverse name variations of the same software into unified entities using clustering, synonym retrieval, and string similarity, resulting in $14,770,209$ items.

For software properties, this dataset contains **`software name`** (extracted by NER), **`software ID`** (an assigned identifier), **`the text`** (the sentence mentioning the software), **`disambiguated software name`** (the disambiguated software name), **`version`** (the extracted software version), and the **`curation label`** (indicating whether the mention refers to software or not). For publication properties, this dataset contains several identifiers of publications, such as **`PMCID`** and **`DOI`**. An example of raw data can be seen in Appendix A.1.

3.2. Data Augmentation

The CZ Software Mentions dataset provides only basic publication identifiers, such as the **`PMCID`**, but lacks essential metadata that carry detailed semantic information, including titles and abstracts. Therefore, we augment the dataset by collecting additional publication-related information. Detailed information about our augmented dataset is provided in Table 1. We retrieve publication metadata from Europe PMC, which is developed by EMBL-EBI with support from the Europe PMC Funders' Group and provides comprehensive access to biomedical and life science literature.

Specifically, we collect the metadata fields of **`title`**, **`abstract`**, **`full text`**, **`publication date`**, and **`journal name`** from the XML-formatted records. A single publication may mention multiple software entities, possibly multiple times. Therefore, we collect $1{,}732{,}602$ unique PMCIDs from $14{,}770{,}209$ software mention records. Based on these unique PMCIDs, we successfully obtain $1{,}707{,}170$ publication metadata records.

3.3. Graph Construction

We construct the Graph-TempCZ based on the augmented dataset. The graph consists of two types of nodes: software nodes and publication nodes, and one type of relationship: the mention relationship. The software nodes contain attributes: **`software ID`**, **`software name`**, and **`disambiguated software name`**. As the software lacks textual information such as a description, we aggregate the sentences mentioning the software as the text information of software nodes.

The publication nodes contain attributes: **`PMCID`**, **`PubMed id`**, **`DOI`**, **`publication date`**, **`title`**, **`abstract`**, **`full text`**, and **`journal name`**. As for the relationships, a single publication may contain multiple mentions of the same software. We represent the number of such mentions as the weight of the relationship.

This graph includes disconnected subgraphs where a publication mentions only one software, and that software is mentioned solely by that paper.

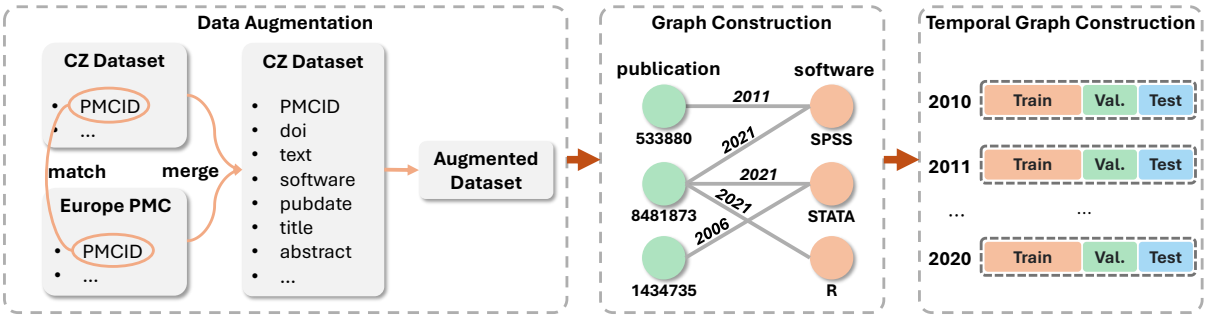

Figure 2: Overview of the data augmentation and graph construction process. In the data augmentation step, we use the **PMCID**s from the disambiguated CZ Software Mentions dataset (Istrate et al., 2022) to retrieve publication metadata from Europe PMC, including titles, abstracts, and journal information. After filtering out nodes without attributes and removing disconnected subgraphs, the augmented data are organized into a large-scale interconnected publication–software graph, **Graph-TempCZ**. Finally, a series of yearly subgraphs from 2010 to 2020 are constructed to support temporal link prediction experiments, and each yearly dataset is split into training, validation, and test sets with an $8:1:1$ ratio.

Attribute	Description	Data Type
`ID`	Identity column, software mention ID	Text
`PMCID`	The PMCID of the paper	Numerical value
`pmid`	PubMed id of the paper	Numerical value
`DOI`	DOI of the paper	Text
`text`	The sentence, from which the software mention was extracted	Text
`software`	The extracted software mention	Text
`curation_label`	Curation result for the software mention by curation team	Text
`mapped_to_software`	Disambiguated software mention	Text
`pubdate`	Publication year of the paper	Date
`title`	title of the publication	Text
`abstract`	abstract of the paper	Text
`full_text`	full text of the paper	Text
`publication_date`	detailed date of the paper	Date
`journal_name`	journal name that the paper published	Text

Table 1: Overview of the augmented data. Using the **PMCID** as an identifier, we retrieve publication records in XML format from Europe PubMed Central and extract key metadata, including the **title**, **abstract**, **full text**, detailed **publication date** (day, month, and year), and **journal name**.

We remove these isolated subgraphs. Finally, we construct a large-scale graph with interconnected nodes, containing two types of nodes (publication and software) and one type of edge (mention).

3.4. Temporal Subgraph Construction

As mentioned in Section 1, we evaluate the prediction of software usage based on mentions in neighboring years. Therefore, we further construct a series of temporal subgraphs derived from Graph-TempCZ to support temporal link prediction.

We split the data by year and select subsets from 2010 to 2020 in Graph-TempCZ. To preserve the original graph structure of Graph-TempCZ, we use annual data instead of a finer-grained split. Furthermore, we retain all annual data instead of sampling an equal number of nodes or relationships, ensuring that each yearly graph faithfully reflects the actual scale and structure of the publication-software graph. We split the annual data into training, validation, and test sets in a $8:1:1$ ratio.

3.5. Data Statistics

For the Graph-TempCZ dataset, which contains $6{,}516{,}437$ edges, $1{,}460{,}616$ publication nodes, and $893{,}433$ software nodes, the mean degree of publication nodes is 4.5, and that of software nodes is 7.3. This indicates that, on average, each software is mentioned in 4.5 publications, while each publication mentions 7.3 pieces of software.

For the temporal subgraphs, detailed statistics are presented in Table 2. From 2010 to 2020, the numbers of publications and software entities increase annually from $26{,}703$ and $34{,}923$ to $239{,}179$ and $247{,}713$, respectively. Meanwhile, the number of mentions increases from $113{,}642$ to $1{,}128{,}364$.

Year	$\text{Node}_{\text{publication}}$	$\text{Node}_{\text{software}}$	Total Node	Edge/Mention	$\text{MD}_{\text{publication}}$	$\text{MD}_{\text{software}}$
2010	26,703	34,923	61,626	113,642	4.3	3.3
2011	36,586	46,156	82,742	153,735	4.2	3.3
2012	53,737	63,175	116,912	216,752	4.0	3.4
2013	71,419	81,299	152,718	287,977	4.0	3.5
2014	84,956	97,613	182,569	343,899	4.0	3.5
2015	99,516	113,379	212,895	412,058	4.1	3.6
2016	115,010	133,332	248,342	487,303	4.2	3.7
2017	133,041	147,642	280,683	572,530	4.3	3.9
2018	145,834	164,697	310,531	649,350	4.5	3.9
2019	176,110	195,059	371,169	811,597	4.6	4.2
2020	239,179	247,713	486,892	1,128,364	4.7	4.6

Table 2: Overview of the constructed temporal graph from 2010 to 2020. The numbers of nodes and edges consistently increase year by year, reflecting the growing volume of publications and software usage. $\textbf{MD}_{\textbf{publication}}$ denotes the mean degree of publication nodes ($\textbf{Node}_{\textbf{publication}}$), and $\textbf{MD}_{\textbf{software}}$ denotes the mean degree of software nodes ($\textbf{Node}_{\textbf{software}}$). In total, 79.45% ($5{,}177{,}207$ out of $6{,}516{,}437$) of mentions (**Edge/Mention**) are retained from the Graph-TempCZ dataset. The mean degree of publication nodes ranges from 4.0 to 4.7, which remains consistent with the overall graph. In contrast, the mean degree of software nodes increases steadily from 3.3 to 4.6 over the period from 2010 to 2020, indicating that an increasing number of software is being mentioned in publications.

In total, 79.45% of mentions are retained from the Graph-TempCZ dataset. The mean degree of publication nodes ranges from 4.0 to 4.7, which is consistent with the overall graph. However, the mean degree of software nodes increases steadily from 3.3 to 4.6 over the same period, suggesting that an increasing amount of software is being mentioned over time. This highlights the importance of analyzing software usage from a temporal perspective.

4. Experimental Setup

Our study aims to answer two research questions:

RQ1: Can link prediction be used to reconstruct software mentions in a publication-software graph? As mentioned in Section 1, publications that mention software inherently form a mention/citation graph. Therefore, these mention relationships can be represented as links in the graph. We further explore whether link prediction can be used to reconstruct software mentions in this publication-software graph.

RQ2: Can software usage be predicted from data in neighboring years, framed as a temporal link prediction task? Trends in software usage are crucial for predicting future usage and development of software. Therefore, we further investigate the effectiveness of training a temporal link prediction model on data from a specific year to predict software usage from 2010 to 2020.

4.1. Models

For link prediction on Graph-TempCZ, we use both traditional and graph learning methods.

For the traditional methods, we extract three types of features and input them into the XGBoost model (Chen and Guestrin, 2016). XGBoost is a widely adopted and highly competitive model among traditional feature-based methods (Rajkomar et al., 2018). These features include (1) node-attribute features, (2) graph-topological features, and (3) a combination of the two.

For the graph learning methods, we employ a two-layer GraphSAGE (Hamilton et al., 2017a) as an end-to-end model. GraphSAGE is an inductive and scalable framework that can efficiently generate embeddings for previously unseen data. Therefore, GraphSAGE is more suitable for predicting software usage in our large-scale graph compared with its counterparts, such as the Graph Convolutional Network (Kipf and Welling, 2017) and the Graph Isomorphism Network (Xu et al., 2019).

For temporal link prediction, we then adopt a two-layer GraphSAGE model as the base model. We train a separate model on the training set of each year from 2010 to 2020 and evaluate each model on the test sets of all years, resulting in 11 trained models and 121 corresponding test evaluations.

4.2. Baseline

For link prediction on Graph-TempCZ, the XGBoost model using combination features is considered the baseline. For the temporal link prediction task, the GraphSAGE model trained and tested on the 2010 training and test set serves as the baseline.

4.3. Training Setting

All XGBoost models employ $1,000$ trees, a learning rate of 0.1, and a maximum depth of 8 under a binary logistic objective with $1,000$ rounds. The dataset uses random negative sampling with a positive-to-negative sample ratio of $2{:}1$.

The dataset contains only positive edges indicating that a publication uses a software. To construct a supervised link prediction task, we generate negative instances via random negative sampling. For each positive publication and software pair, we randomly sample software nodes that are not used in the publication in the graph and treat these pairs as negatives.

For textual representation, we use paper abstracts to represent publications and aggregated software texts to represent software nodes. Abstracts summarize the core contributions and research focus of a publication, making them a concise representation of paper content. In contrast, software aggregated texts are typically extracted from analysis or experimental sections, where the purpose and usage details of the software are described. This design allows the textual features of both node types to capture their respective semantics while reducing the risk of information leakage, such as directly mentioning the software in the abstract.

We apply the following configurations for model training:

XGBoost with node-attribute features. We compute cosine similarities between paper abstracts and software aggregated texts using Sentence-BERT (SBERT) (Reimers and Gurevych, 2019) with the `all-MiniLM-L6-v2` model, yielding 384-dimensional embeddings.

XGBoost with topological model features. We use a comprehensive set of structural descriptors derived from the publication-software graph, including local indices (Jaccard similarity, cosine similarity, common neighbors (Liben-Nowell and Kleinberg, 2003), preferential attachment (Barabási and Albert, 1999), Adamic-Adar (Adamic and Adar, 2003)) and global indices (Katz (Katz, 1953), SimRank (Jeh and Widom, 2002), Random Walk with Restart (Tong et al., 2006), Stochastic Block Model entropy (Hoff et al., 2002), HITS (Kleinberg, 1999), PageRank (Page et al., 1999)).

XGBoost with combination features. This model integrates both node-attribute and topological features described above.

Graph Neural Network (GNN) model. We use the same SBERT model to obtain 384-dimensional sentence embeddings for paper abstracts and software aggregated texts, which are projected to a hidden size of 128. The GNN architecture is a two-layer GraphSAGE network with sum aggregation. Training is conducted with a learning rate of 1×10^{-4}, a batch size of 256, and a negative sampling ratio of 2.0 with 50 epochs.

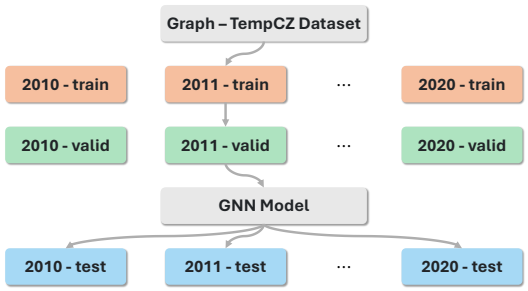

Figure 3: Training pipeline on the temporal graph. The Graph Neural Network (GNN) model is trained and validated on data from a specific year and tested across all years from 2010 to 2020.

Temporal GNN model. We adopt the same training setting as the above GNN model. However, as shown in Figure 3, the training pipeline differs in that the model is trained and validated on data from a specific year (e.g., 2011) and evaluated across all yearly test sets from 2010 to 2020.

4.4. Evaluation Metrics

We use three metrics to evaluate the models' link prediction performance on the test sets: accuracy (ACC), F1-score (F1), and recall. Accuracy measures the proportion of correctly predicted links. Recall evaluates the model's ability to identify true positive links. F1 balances precision and recall to handle prediction imbalance.

5. Experiment Result and Analysis

5.1. RQ1: Link Prediction on Graph-TempCZ

As shown in Table 3, the Graph Neural Network (GNN) model consistently outperforms all other methods across all metrics, achieving an accuracy of 0.9288 and a Recall of 0.9164. This indicates that the GNN effectively captures both semantic information from text and structural dependencies in the mention graph. The combination model, which integrates node-attribute and topological features, also exhibits strong performance (accuracy: 0.8690, F1: 0.8680), outperforming either feature type alone.

Among the traditional methods, the topological-feature model surpasses the node-attribute model,

	Accuracy	F1	Recall
Traditional Method			
Node Attributes	0.7845	0.7660	0.7713
Topological Features	0.8137	0.7893	0.7916
Combination	0.8690	0.8680	0.7957
GNN Method			
GraphSAGE	**0.9288**	**0.8905**	**0.9164**

Table 3: Link prediction performance comparison on Graph-TempCZ.

suggesting that structural relationships among papers and software play a more decisive role than textual similarity alone. However, the improvement achieved by the GNN model highlights the advantage of jointly learning feature representations and graph topology in an end-to-end manner.

Overall, these results confirm that using graph structures through GNN enhances link prediction performance in the publication-software mention network, validating the effectiveness of our graph representation for predicting software usage.

5.2. RQ2: Temporal Link Prediction

We further evaluate temporal link prediction by training the GNN on each individual year from 2010 to 2020 and testing it across all yearly test sets.

As shown in Table 4, models trained on data from any given year can be effectively transferred to predict software usage in other years. For instance, the model trained on 2018 achieves an accuracy of 0.9136 on the 2018 test set; on the other test sets, accuracy ranges from 0.8981 to 0.9076. F1 and recall show the same pattern. These results demonstrate that the temporal dynamics of publication-software relationships remain sufficiently stable for cross-year transfer.

Models achieve the highest performance across all metrics when training and testing are conducted on the same year, while performance gradually declines as the temporal gap widens when predicting future data. For example, the model trained on the 2010 data achieves its best results with an accuracy of 0.8645 and an F1 score of 0.7996. When evaluated on test sets from 2011 to 2020, its performance decreases from 0.8557 to 0.8246 in accuracy, from 0.7838 to 0.7304 in F1 score, and from 0.7847 to 0.7125 in recall. This decay pattern highlights the importance of temporal prediction quality, which remains higher when training and testing years are temporally close.

For forecasting future software usage, training on the most recent data is preferable. In particular, the models exhibit a roughly two-year effective transfer window, within which performance degradation remains limited. For instance, the model trained on 2010 data loses only 0.0170 in accuracy when applied to 2012, but the drop increases to 0.0334 after a five-year gap (2015). A similar trend is observed for F1 and recall.

In addition, we also analyze the performance vertically by fixing the test year and comparing models trained on different years. The results reveal a clear upward trend: models trained on later years consistently achieve relatively higher scores on the same test set. For example, on the 2015 test set, accuracy improves from 0.8311 when trained on 2010 data to 0.8992 when trained on 2020 data, and F1 rises from 0.7439 to 0.8517. This pattern shows that newer training data, which includes larger graph scales, denser publication-software links, and more diverse software mentions, enables the model to capture richer structural and semantic information, thereby improving temporal generalization.

Overall, temporal link prediction on the selected temporal part of the Graph-TempCZ dataset demonstrates three key characteristics: (i) cross-year generalization, as models trained on any given year can predict links in other years with reasonably stable performance; (ii) strong near-diagonal dominance, where models achieve the best results when training and testing years are close, and performance gradually declines as the temporal gap widens; and (iii) consistent vertical improvement, indicating that models trained on later, larger, and denser graphs generalize better across all test years. Together, these findings suggest that richer graph structures and more comprehensive software coverage in recent years substantially enhance temporal robustness, while temporal distance remains the primary factor affecting transfer accuracy.

6. Discussion

We introduced **Graph-TempCZ**, a large temporal graph that captures publication-software mention relationships. Predicting software mentions in a link prediction task enables structured analysis of how publications and software are connected.

Our experiments show that the graph-based model substantially outperforms feature-based baselines in predicting publication-software links, highlighting the value of graph representations. Temporal experiments show strong generalization across adjacent years but a gradual decline as the temporal gap widens, indicating evolving yet locally stable usage patterns. These findings demonstrate that graph representations provide an effective framework for representing software usage and understanding its evolution over time.

In this work, we represent software usage in publications as a publication-software graph, which is highly similar to publication citation. Enriching node and edge attributes (e.g., software descriptions and

Accuracy

Train \ Test	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
2010	0.8645	0.8557	0.8475	0.8367	0.8358	0.8311	0.8325	0.8295	0.8294	0.8273	0.8246
2011	0.8687	0.8656	0.8515	0.8450	0.8405	0.8370	0.8374	0.8365	0.8365	0.8334	0.8339
2012	0.8799	0.8742	0.8761	0.8661	0.8602	0.8594	0.8585	0.8583	0.8577	0.8542	0.8540
2013	0.8933	0.8925	0.8860	0.8883	0.8712	0.8737	0.8716	0.8706	0.8692	0.8673	0.8651
2014	0.8906	0.8941	0.8878	0.8896	0.8931	0.8777	0.8775	0.8765	0.8759	0.8750	0.8733
2015	0.8914	0.8913	0.8901	0.8890	0.8890	0.8965	0.8797	0.8798	0.8788	0.8793	0.8776
2016	0.8849	0.8882	0.8876	0.8900	0.8912	0.8929	0.9038	0.8845	0.8840	0.8826	0.8817
2017	0.8885	0.8869	0.8899	0.8917	0.8941	0.8944	0.8945	0.9068	0.8867	0.8865	0.8857
2018	0.8989	0.8981	0.8997	0.9016	0.9022	0.9030	0.9051	0.9076	0.9136	0.8997	0.8976
2019	0.8861	0.8877	0.8868	0.8895	0.8893	0.8917	0.8934	0.8952	0.8945	0.9109	0.8826
2020	0.8888	0.8907	0.8955	0.8960	0.8975	0.8992	0.9018	0.9045	0.9061	0.9068	0.9200

F1

Train \ Test	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
2010	0.7996	0.7838	0.7703	0.7543	0.7510	0.7439	0.7459	0.7413	0.7394	0.7359	0.7304
2011	0.8146	0.8091	0.7842	0.7741	0.7672	0.7623	0.7629	0.7612	0.7601	0.7549	0.7545
2012	0.8263	0.8202	0.8204	0.8057	0.7980	0.7964	0.7949	0.7937	0.7925	0.7867	0.7854
2013	0.8451	0.8441	0.8357	0.8376	0.8153	0.8184	0.8160	0.8140	0.8116	0.8083	0.8050
2014	0.8423	0.8472	0.8386	0.8411	0.8460	0.8246	0.8244	0.8225	0.8216	0.8203	0.8173
2015	0.8423	0.8425	0.8411	0.8403	0.8408	0.8504	0.8289	0.8289	0.8274	0.8279	0.8255
2016	0.8339	0.8380	0.8379	0.8416	0.8430	0.8462	0.8606	0.8353	0.8345	0.8326	0.8308
2017	0.8393	0.8375	0.8412	0.8436	0.8474	0.8483	0.8489	0.8652	0.8380	0.8378	0.8365
2018	0.8512	0.8509	0.8531	0.8562	0.8571	0.8588	0.8620	0.8659	0.8745	0.8545	0.8513
2019	0.8346	0.8373	0.8364	0.8404	0.8403	0.8440	0.8465	0.8492	0.8485	0.8714	0.8339
2020	0.8371	0.8399	0.8460	0.8468	0.8494	0.8517	0.8560	0.8602	0.8625	0.8637	0.8846

Recall

Train \ Test	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
2010	0.8111	0.7847	0.7672	0.7522	0.7429	0.7357	0.7374	0.7328	0.7260	0.7216	0.7125
2011	0.8648	0.8542	0.8097	0.7967	0.7883	0.7839	0.7849	0.7818	0.7769	0.7696	0.7656
2012	0.8571	0.8606	0.8489	0.8331	0.8284	0.8251	0.8227	0.8173	0.8156	0.8067	0.8013
2013	0.8731	0.8728	0.8702	0.8645	0.8529	0.8536	0.8539	0.8492	0.8452	0.8394	0.8349
2014	0.8763	0.8803	0.8742	0.8769	0.8803	0.8625	0.8629	0.8586	0.8576	0.8560	0.8504
2015	0.8706	0.8719	0.8720	0.8760	0.8790	0.8824	0.8738	0.8730	0.8716	0.8708	0.8687
2016	0.8666	0.8677	0.8715	0.8769	0.8762	0.8843	0.8908	0.8783	0.8774	0.8761	0.8712
2017	0.8734	0.8745	0.8747	0.8759	0.8825	0.8852	0.8891	0.8978	0.8790	0.8793	0.8767
2018	0.8681	0.8729	0.8738	0.8785	0.8801	0.8846	0.8890	0.8949	0.9036	0.8833	0.8798
2019	0.8623	0.8664	0.8681	0.8728	0.8732	0.8788	0.8813	0.8852	0.8866	0.9050	0.8839
2020	0.8568	0.8607	0.8614	0.8625	0.8672	0.8682	0.8758	0.8819	0.8830	0.8864	0.9205

Table 4: Temporal link prediction results on selected data from 2010 to 2020. Rows indicate the training year and columns indicate the test year. The three panels report accuracy, F1, and recall, respectively; darker shading denotes higher scores. Diagonal cells correspond to in-year evaluation.

version information) to obtain more powerful representations, exploring temporal graph learning methods to better handle long-range temporal shifts, and representing software from publication nodes, as in Zevallos et al. (2024) for citation graphs, are promising directions for future work.

In addition, our large-scale graph can be used by metascience researchers to perform case studies that explore changes in software usage in publications from 2010 to 2020 based on our temporal subgraphs at yearly or daily granularity. This can supplement existing survey-based methods (e.g. Joppa et al., 2013), bibliometric analyses (e.g. Howison and Bullard, 2016) and webometric analysis (e.g. Orduña-Malea and Costas, 2021). An example of such a case study would be the replacement of SPSS by R for statistical analysis in the biomedical field, shown in a traditional bibliometric analysis by Masuadi et al. (2021), as well as case studies on trends that have been observed in citation graph-based metascience research (Mohammad, 2020; Turki et al., 2024; Mariani et al., 2018).

7. Limitations

Our dataset covers over six million mention links, but it is limited to the biomedical domain, as inherited from the CZ Software Mentions dataset (Istrate et al., 2022). Extending the graph to additional disciplines and integrating other metadata (e.g., author networks, funding, or repository information) would enhance generalization and enable richer analyses of software usage across the scientific ecosystem.

To preserve the original graph structure of Graph-TempCZ, our temporal graphs are constructed at the granularity of one year, which may overlook finer temporal dynamics such as monthly or quarterly trends in software usage.

We employ a Graph Neural Network (GNN) model to explore generalization on the temporal graph dataset, whereas Temporal Graph Neural Networks (e.g., TGN (Rossi et al., 2020), TGAT (da Xu et al., 2020)) can explicitly model evolving node states and continuous-time interactions.

The CZ Software Mentions dataset uses named entity recognition methods to extract software names, while the accuracy of these methods is limited, which can be a factor affecting the practical application of our methods. We aggregate software mention text as the software textual description, which is only an approximation of the software's real functional description. With the rise of large language models with strong zero-shot information retrieval ability (Lewis et al., 2020), extracting software names and retrieving software descriptions could be promising directions for future work. The further advancement of these applications would further enhance our methods for software usage prediction.

8. Ethics Statements

Data sources and licensing. This study builds on two publicly available resources: the disambiguated CZ Software Mentions dataset (Istrate et al., 2022), and metadata retrieved from Europe PMC. We use these sources strictly under their respective terms of use. All derived releases (code, configuration, and graph indices) will comply with the original licenses and redistribution constraints.

Potential risks. Our models estimate links/usage between publications and software and can be used to analyze or forecast software usage trends. While we do not foresee direct harmful applications, trend forecasting could be misused (e.g., for gaming metrics or overstating impact).

9. Bibliographical References

Lada A Adamic and Eytan Adar. 2003. Friends and neighbors on the web. *Social Networks*, 25(3):211–230.

Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *Science*, 286(5439):509–512.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA. Association for Computing Machinery.

da Xu, chuanwei ruan, evren korpeoglu, sushant kumar, and kannan achan. 2020. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations*.

Riccardo Del Gratta, Francesca Frontini, Monica Monachini, Gabriella Pardelli, Irene Russo, Roberto Bartolini, Fahad Khan, Claudia Soria, and Nicoletta Calzolari. 2016. LREC as a graph: People and resources in a network. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2529–2532, Portorož, Slovenia. European Language Resources Association (ELRA).

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*, 72(7):870–884.

Carole Goble. 2014. Better software, better research. *IEEE Internet Computing*, 18(5):4–8.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017a. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2017b. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1025–1035, Red Hook, NY, USA. Curran Associates Inc.

Peter D Hoff, Adrian E Raftery, and Mark S Handcock. 2002. Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97(460):1090–1098.

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):2137–2155.

Ana-Maria Istrate, Donghui Li, Dario Taraborelli, Michaela Torkar, Boris Veytsman, and Ivana Williams. 2022. A large dataset of software mentions in the biomedical literature. *arXiv preprint arXiv:2209.00693*.

Glen Jeh and Jennifer Widom. 2002. Simrank: a measure of structural-context similarity. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 538–543, New York, NY, USA. Association for Computing Machinery.

Lucas N Joppa, Greg McInerny, Richard Harper, Lara Salido, Kenji Takeda, Kenton O'Hara, David Gavaghan, and Stephen Emmott. 2013. Troubling trends in scientific software use. *Science*, 340(6134):814–815.

Leo Katz. 1953. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

David Liben-Nowell and Jon Kleinberg. 2003. The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, page 556–559, New York, NY, USA. Association for Computing Machinery.

Joseph Mariani, Gil Francopoulo, and Patrick Paroubek. 2018. Measuring innovation in speech and language processing publications. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Emad Masuadi, Mohamud Mohamud, Muhannad Almutairi, Abdulaziz Alsunaidi, Abdulmohsen K Alswayed, Omar F Aldhafeeri, and Muhannad B Almutairi. 2021. Trends in the usage of statistical software and their associated study designs in health sciences research: a bibliometric analysis. *Cureus*, 13(1).

Saif M. Mohammad. 2020. NLP scholar: A dataset for examining the state of NLP research. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 868–877, Marseille, France. European Language Resources Association.

Ankan Mullick, Shubhraneel Pal, Tapas Nayak, Seung-Cheol Lee, Satadeep Bhattacharjee, and Pawan Goyal. 2022. Using sentence-level classification helps entity extraction from material science literature. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 4540–4545, Marseille, France. European Language Resources Association.

Enrique Orduña-Malea and Rodrigo Costas. 2021. Link-based approach to study scientific software usage: The case of vosviewer. *Scientometrics*, 126(9):8153–8186.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford infolab.

Xuelian Pan, Erjia Yan, Ming Cui, and Weina Hua. 2018. Examining the usage, citation, and diffusion patterns of bibliometric mapping software: A comparative study of three tools. *Journal of Informetrics*, 12(2):481–493.

Xuelian Pan, Erjia Yan, Qianqian Wang, and Weina Hua. 2015. Assessing the impact of software on science: A bootstrapped learning of software entities in full-text papers. *Journal of Informetrics*, 9(4):860–871.

Hyoungjoo Park and Dietmar Wolfram. 2019. Research software citation in the data citation index: Current practices and implications for research software sharing and reuse. *Journal of Informetrics*, 13(2):574–582.

Alvin Rajkomar, Eyal Oren, Kai Chen, Andrew M Dai, Nissan Hajaj, Michaela Hardt, Peter J Liu, Xiaobing Liu, Jake Marcus, Mimi Sun, et al. 2018. Scalable and accurate deep learning with electronic health records. *NPJ digital medicine*, 1(1):18.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael M. Bronstein. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. *arXiv preprint arXiv:2006.10637*.

David Schindler, Felix Bensmann, Stefan Dietze, and Frank Krüger. 2021. Somesci- a 5 star open data gold standard knowledge graph of software mentions in scientific articles. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 4574–4583, New York, NY, USA. Association for Computing Machinery.

David Schindler, Felix Bensmann, Stefan Dietze, and Frank Krüger. 2022. The role of software in science: a knowledge graph-based analysis of software mentions in pubmed central. *PeerJ Computer Science*, 8:e835.

David Schindler, Benjamin Zapilko, and Frank Krüger. 2020. Investigating software usage in the social sciences: A knowledge graph approach. In *European Semantic Web Conference*, pages 271–286. Springer.

Sudipta Singha Roy and Robert E. Mercer. 2024. Enhancing scientific document summarization with research community perspective and background knowledge. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 6048–6058, Torino, Italia. ELRA and ICCL.

Yuka Tateisi, Tomoko Ohta, Sampo Pyysalo, Yusuke Miyao, and Akiko Aizawa. 2016. Typed entity and relation annotation on computer science papers. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3836–3843, Portorož, Slovenia. European Language Resources Association (ELRA).

Hanghang Tong, Christos Faloutsos, and Jia-yu Pan. 2006. Fast random walk with restart and its applications. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 613–622.

Houcemeddine Turki, Abraham Toluwase Owodunni, Mohamed Ali Hadj Taieb, René Fabrice Bile, and Mohamed Ben Aouicha. 2024. A decade of scholarly research on open knowledge graphs. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 438–448, Torino, Italia. ELRA and ICCL.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Yuzhuo Wang and Kai Li. 2024. How do official software citation formats evolve over time? a longitudinal analysis of r programming language packages. *Scientometrics*, 129(7):3997–4019.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? In *International Conference on Learning Representations*.

Bo Yang, Ronald Rousseau, Xue Wang, and Shuiqing Huang. 2018. How important is scientific software in bioinformatics research? a comparative study between international and chinese research communities. *Journal of the Association for Information Science and Technology*, 69(9):1122–1133.

Klim Zaporojets, Lucie-Aimée Kaffee, Johannes Deleu, Thomas Demeester, Chris Develder, and Isabelle Augenstein. 2022. Tempel: Linking dynamically evolving and newly emerging entities. In *Advances in Neural Information Processing Systems*, volume 35, pages 1850–1866. Curran Associates, Inc.

Rodolfo Joel Zevallos, John E. Ortega, and Benjamin Irving. 2024. Related work is all you need. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 13874–13878, Torino, Italia. ELRA and ICCL.

Haoling Zhang, Alberto Maillo, Sumeer Ahmad Khan, Xabier Martínez de Morentin, Robert Lehmann, David Gomez-Cabrero, and Jesper Tegnér. 2024. Reviewability and supportability: New complementary principles to empower research software practices. *Computational and Structural Biotechnology Journal*, 23:3989–3998.

Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*,

NIPS'18, page 5171–5181, Red Hook, NY, USA. Curran Associates Inc.

Rongying Zhao and Mingkun Wei. 2017. Impact evaluation of open source software: an altmetrics perspective. *Scientometrics*, 110(2):1017–1033.

A. Example of Data

A.1. Example of Raw Data

This raw data contains three data sources of the software SPSS in Table 5.

	One example of each raw dataset		
	PMC-OA Non-Commercial	PMC-OA Commercial	Publishers collection
ID	SM165	SM165	SM165
curation_label	software	software	software
DOI	10.1007/ s13205-021-02985-4	10.1186/ s41205-021-00115-7	10.1016/ j.ihj.2016.03.007
license	non_comm	comm	NA
location	non_comm/3_Biotech/ PMC8446481.nxml	comm/3D_Print_Med/ PMC8411549.nxml	NA
number	30	13	29
PMCID	8446481	8411549	NA
pmid	34549015.0	34471999.0	NA
pubdate	2021	2021	2016
software	SPSS	SPSS	SPSS
source	Statistical analysis	Materials and methods	Methods
text	The statistical calculation was determined with SPSS windows version 15.0 (Cary, NC)	All statistical analyses were performed using SPSS 19.0 (SPSS Inc, Chicago, IL)	All data were analyzed using the SPSS 19 package program (SPSS, Chicago, IL, USA)
version	NA	19.0	NA

Table 5: Example of raw data. This example of raw data contains three data instances from three data resources. For software properties, this dataset contains **`software`**, **`ID`**, **`text`**, **`curation label`**, and **`version`** **`software`** denotes the software name. **`ID`** is an assigned identifier. **`text`** denotes the sentence mentioning the software. **`version`** denotes the extracted software version. **`curation label`** indicates whether the mention refers to software or not. For publication properties, this dataset contains **`PMCID`**, **`pmid`**, **`DOI`**, **`pubdate`** (publication date), and **`source`** (journal name).