Building an Explainable Graph-based Biomedical Paper Recommendation System

Hermann Kroll
krollh@acm.org
Institute for Information Systems, TU Braunschweig
Germany

Christin K. Kreutz
ckreutz@acm.org
TH Mittelhessen & Herder Institute
Germany

Bill Matthias Thang
m.thang@tu-bs.de
Institute for Information Systems, TU Braunschweig
Germany

Philipp Schaer
philipp.schaer@th-koeln.de
TH Köln (University of Applied Sciences)
Germany

Wolf-Tilo Balke
balke@ifis.cs.tu-bs.de
Institute for Information Systems, TU Braunschweig
Germany

Abstract

Digital libraries provide different access paths, allowing users to explore their collections. For instance, paper recommendation suggests literature similar to some selected paper. Their implementation is often cost-intensive, especially if neural methods are applied. Additionally, it is hard for users to understand or guess why a recommendation should be relevant for them. That is why we tackled the problem from a different perspective. We propose XGPRec, a graph-based and thus explainable method which we integrate into our existing graph-based biomedical discovery system. Moreover, we show that XGPRec (1) can, in terms of computational costs, manage a real digital library collection with 37M documents from the biomedical domain, (2) performs well on established test collections and concept-centric information needs, and (3) generates explanations that proved to be beneficial in a preliminary user study. We share our code so that user libraries can build upon XGPRec.

Keywords

Explainable Paper Recommendation, Biomedical Document Retrieval, System Design, User Interface, Digital Libraries

1 Introduction

This article is the short version of our technical report [8]. Digital libraries provide effective access paths for users to explore their underlying collections. This vast number of publications to look at combined with possibly under-specified information needs of users can lead to challenges when trying to find related work for a topic of interest as keyword-based options are insufficient [6]. As a remedy, paper recommendation systems suggest related literature based on some user's selected article(s) [3, 18]. We focus on the following task definition for paper recommendation: *Given an initial article, what are other relevant articles with regard to the input article?*

Current paper recommendation systems do not focus on limiting computational complexity or performing the task with fewer resources, even though the emissions produced by neural retrieval methods are several orders of magnitude higher than those of BM25 [16]. Many methods rely on cost-intensive deep learning [6]. In practice, the computational costs of neural retrieval methods and the collection of representative training data usually hinder their implementation in a digital library. Another understudied issue but desirable goal in paper recommendation is explainability [6]. Explanations are helpful to (1) justify individual recommendations, (2) understand how a system works, and (3) distinguish good from bad recommendations via users' feedback [1].

We tackled the two issues of high costs (in terms of retrieval and training data collection) and missing explainability by building upon graph-based document representations in cooperation with PubPharm, the specialized service for Pharmacy (some of this paper's authors are part of the PubPharm team). In addition to keyword-based retrieval, PubPharm offers a graph-based discovery system called the Narrative Service[1] as a more sophisticated access path for users. The Narrative Service allows users to formulate their searches as narrative query graphs, i.e., as short stories of interest involving relevant biomedical concepts and their interactions. The service comes with two central advantages [10]: precise literature retrieval and structured overviews of the literature when using variables, e.g., structuring the literature by possible treatment options in adults with diabetes. We expand our and PubPharm's existing Narrative Service and provide more functionality to our users. We design and implement an explainable paper recommendation system (a demo video is available at[2]), called **XGPRec**. Graph representations can visualize complex interconnections and enable users from the biomedical domain to immediately grasp if a paper fits what they mean to find [7]. In this work, we focus on justifying recommendations [1], which we achieve by displaying overlapping graph patterns between an initial and a candidate paper to explain recommendations for users. We demonstrate XGPRec at the showcase of a real digital library collection, namely the extensive MEDLINE document collection with about 37 million documents. Our research objective for this work is thus: *How can we design a fast, reliable and explainable paper recommendation system?* Our code is available at GitHub[3] and Software Heritage[4]. In this work, we focus on central findings of our research. A discussion about related work and details our system and evaluation can be found in our corresponding technical report [8].

[1] https://narrative.pubpharm.de
[2] https://youtu.be/oLZFCtVuQWU
[3] https://github.com/HermannKroll/NarrativeRecommender/
[4] Software Heritage ID:swh:1:dir:eaeaac5c6a9ccb00542431398e43dec34d910faf

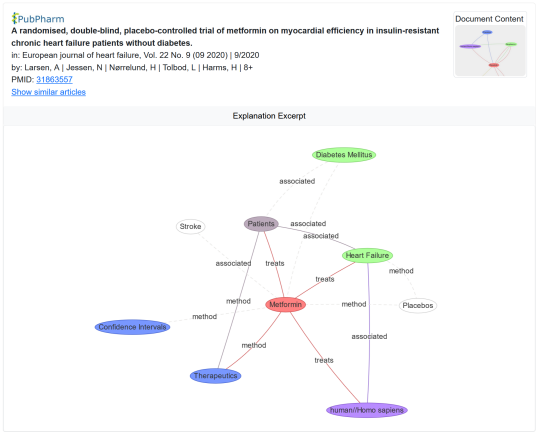

Figure 1: Screenshot of our prototypical system: The generated explanation why the candidate document should be relevant to the input document is shown. Shared information (nodes and edges) are visualized with colors whereas information that is added by the candidate document is visualized as dashed lines and not colored nodes.

2 Narrative Service – Graph-based Discovery

PubPharm's graph-based retrieval service, called the Narrative Service [10], allows users to express their information need as a narrative query graph, i.e., graph patterns with triple-like statements (concept, interaction, concept). A query is then answered by documents that contain the search graph pattern. For the query processing step, they transformed texts into graphs by detecting relevant concepts and extracting their interactions. Concepts were identified by deriving annotations from the PubTator service [19, 20] and performing a dictionary-based concept linking through vocabularies derived from ChEBML [14], Wikidata [17] and the Medical Subject Headings. Statements (concept interactions) were extracted by using PathIE and a sentence-based extraction method that extracts general *association* statements if two concepts were mentioned within the same sentence. Briefly, PathIE extracted a statement between two concepts if two detected concepts were connected on the dependency parse of a sentence (basically the grammatical structure) derived with the Stanford CoreNLP toolkit [13]. Details about the extraction methods have already been published in [9] and [10]. The methods' and system's code is freely available[5],[6]. The discovery system currently features around 37 million publications from the MEDLINE collection and 70k COVID-19 pre-prints from ZB MED's preVIEW service [11].

3 Graph-based Recommendation and Explanation

The discovery system contains a set of documents $\mathcal{D}$ [10]. A document $d \in \mathcal{D}$ is represented by its text d_{text} (title and abstract), a list of concept annotations $d_{concepts}$ and a list of extracted statements $d_{statements}$. A concept annotation maps a specific text span of d to a pre-known concept of the concept vocabulary C (the set of all known concepts by the system). Concepts are identified by precise IDs and a type, e.g., *drug* or *disease*. An extracted statement is composed of a subject-predicate-object triple, e.g., (*Metformin*, *treats*, *Diabetes*), the sentence it was extracted from, and a confidence value (how good the extraction was based on some NLP method). Sect. 2 contains details about applied methods and details. The **document graph** of d is the set of subject-predicate-object triples extracted from the document. A document graph is given by $graph(d) = (V, E)$ where V is the set of nodes (detected concepts) and E is a set of concept interactions (triples). This paper aims to reuse our existing document graph representation [10] to perform a reliable and explainable paper recommendation; see Figure 1.

Task Definition (Explainable Paper Recommendation): *Given an input document d, compute a ranked list of documents $\subseteq \mathcal{D}$. For each document d_i in that list provide an explanation e_i of why d_i is related to the input d.*

3.1 Scoring Document Graph Components

First, we score the nodes and edges of a document graph to know which parts are the most important by using three distinct features.

Tf-idf. Some graph nodes or edges might carry more information (are more relevant) than very general ones. In information retrieval, term-frequency tf and inverse-document-frequency idf are two paradigms used to determine a term's relevance concerning a document d. We follow that paradigm and design a tf-idf score for nodes and edges. We define tf for a concept c within a document d as $tf(c,d) = \frac{\#(c,d)}{\#C}$ with $\#(c,d)$ being the number of occurrences of concept c within d and $\#C$ being the number of all annotated concepts within d (for normalization). Next, we define idf for a concept c as $idf(c) = \log \frac{|\mathcal{D}|}{|\{d \in \mathcal{D} \wedge c \in d\}|}$. $|\mathcal{D}|$ is the number of documents in our collection, and the denominator counts documents that include the concept c. With that, we can score nodes with tf-idf as follows:

$$n\text{-}tf\text{-}idf(n,d) = tf(n,d) \cdot idf(n). \tag{1}$$

For edges, we faced issues when maintaining an idf index: First, the index can get quite large (quadratic growth with regard to the size of the concept vocabulary). Second, our statement extraction methods are restricted to sentence levels and might be error-prone [10]. Many connections might be lost during that step, affecting the idf score. That is why we decided to approximate the tf-idf-score for an edge by combining the tf-idf scores of its subject and object plus multiplying it with a predicate specificity (basically a score determining how specific a predicate is: *treats* is more specific than a general *association*). We define the tf-idf score for an edge $e = (s, p, o)$ concerning a document d as:

$$e\text{-}tf\text{-}idf(e,d) = (n\text{-}tf\text{-}idf(s,d) + n\text{-}tf\text{-}idf(o,d)) \cdot specificity(p) \tag{2}$$

Coverage. The discovery system is designed for biomedical abstract retrieval. Each abstract typically starts with some background information in the corresponding field. Concept mentions within that background part might be less important than concepts mentioned across the whole abstract. We therefore define *coverage* of a node (concept) n within a document d as:

[5]https://github.com/HermannKroll/NarrativeIntelligence
[6]Software Heritage ID:swh:1:dir:9e2435bb03d544039cc96fa1b17537050faec6e3

$$n\text{-}coverage(n,d) = \frac{last\_position(n,d) - first\_position(n,d)}{text\_length(d)} \quad (3)$$

The method calculates the difference between the concept's last mention and the first mention within the document and normalizes it by its text length. Coverage approximates whether the concept is used from the beginning to the end or briefly mentioned somewhere as a side note. The higher the coverage is, the more relevant a concept c should be. We then define the coverage of a document graph's edge $e = (s, p, o)$ as:

$$e\text{-}coverage(e,d) = \min(\{n\text{-}coverage(s), n\text{-}coverage(o)\}). \quad (4)$$

Confidence. As mentioned, our extraction methods come with a confidence score, i.e., a score of how sure the tool is about the corresponding extraction. Please note that a document graph's edge could be extracted from different sentences within d. The **confidence** for an edge e is defined as the maximum confidence value of the statement extractions within $d_{statements}$ that support e. We do not have confidence values for concept annotations because the detection methods are dictionary-based linking methods that perform a binary decision.

Scoring. Finally, we can define the scores of nodes and edges. Coverage and tf-idf are combined to compute the score for each node: $n\text{-}score(n,d) = n\text{-}coverage(n,d) \cdot n\text{-}tf\text{-}idf(n,d)$

For edges, we combine confidence, coverage, and tf-idf:

$$e\text{-}score(e,d) = confidence(e,d) \cdot e\text{-}coverage(e,d) \cdot e\text{-}tf\text{-}idf(e,d) \quad (5)$$

Graph Cores. With our previous scoring functions, we can now determine the relevance of different components of each document graph. For instance, the statement extraction step might yield multiple edges (with different predicates) between two nodes, e.g., a general *association* and a specific *treats* predicate. For our recommendation step, we filter the graphs by only keeping the most relevant (best-scored) edge between two concepts and by only keeping edges between different concept types, e.g., drug-disease treatments. A **graph core** is a scored document graph (i.e., the scoring functions have been applied to each node and edge) that only keeps the best-scored edge between two nodes. The function $\textbf{graph-core}(d) = d_{core}$ takes a document and returns its core.

3.2 Candidate Retrieval (First Stage)

The discovery system contains about 37 million documents (as of 04/2024). Given some input document d, comparing its core to that of every other document is obviously too expensive. That is why we headed for a two-step approach: A cheap first stage for initial candidate retrieval and a more expensive second stage that utilizes our graph cores and re-scores the candidate documents. Our three first stages rely on the input document d and work as follows: 1) FSCore searches for documents that include the same edges as the core of d. 2) FSNode searches for documents that include the same nodes as the core of d. 3) FSConcept searches for documents that the same concepts as annotated in document d. The more edges, nodes or concepts a candidate document share, the better it is scored.

We restrict our first stages by a fixed cutoff value, k, so we return only the best-scored k documents. If the score is equal, we sort documents by their IDs in descending order as our system maintains PubMed IDs and higher IDs usually mean newer publications. While BM25 computes nearly continuously distributed scores because it also considers the tf-idf scores of terms within candidate documents, our first stages come with a step function (either a component of the input is contained or not). For instance, the documents between rank 900 and 1200 could have the same score, as they contain the same overlap to the input. For that, we propose a flexible cutoff which considers the score at position k and then cuts the list at the next position the score drops again. We use a hard cutoff at $2 \cdot k$ in any case, to have a maximum boundary.

3.3 Recommendation (Second Stage)

The first stage returns a list of candidate documents $D_{candidates}$. For our second stage, we compute each candidate document's score by comparing their cores to our input document. In brief, if the input document does not have a core, we cannot compute scores and consider all input documents as equally relevant. If the input document has a core, we compare it to every core of the candidate documents. The score for the candidate document is defined as the sum of all edges shared between the input document core and the candidate document's core. An edge is considered as *shared* if it connects the same concepts. This strategy, however, relies on the existence of cores and the expression of relevant information in these cores. This might not always be the case: First, extraction methods are error-prone, i.e., relevant information might be lost. Second, our concept vocabulary might not contain *all* relevant concepts of that domain [10]. That is why we also integrate text-based scoring to consider not-as-graph-expressed information. Here, we use BM25 scoring by considering titles and abstracts. Let d_i be the input document and d_c be some candidate document. We compute the final score, as a weighted sum of the graph overlap and BM25:

$$\textbf{XGPRec}(d_i,d_c) = w_{graph} \cdot core\text{-}overlap(d_i,d_c) + w_{text} \cdot BM25(d_i,d_c) \quad (6)$$

The *core-overlap* returns the score of the core overlap based on our previous algorithm, and BM25 returns the BM25 score when comparing the text (title plus abstract) of d_i and d_c. Note that we normalize *core-overlap* and BM25 scores with regard to a candidate document list $D_{candidate}$ so documents receive scores between [0, 1] which makes both scores comparable and combineable.

3.4 Explanation Generation

Our algorithm takes an input document, some candidate document, and a parameter l as its input. The parameter l determines the length of the explanation to generate. Our idea is to take l edges shared between the input documents and the candidate and mark them as shared (later visualized in colors). In addition, we take up to $l \cdot 2$ edges of the candidate document, mark them as not shared (later visualized as dashed lines and not-colored nodes), and add them to our explanation. More precisely, we only consider the edges of the candidate core connected to one node shared between the input and candidate core. These additional edges should help the user to understand what the candidate documents add as new information to the shared pattern. This way users simultaneously see what is shared and what can be expected as new information in the candidate document.

Table 1: Detailed evaluation of our recommendation approach XGPRec.

Dataset	Strategy	T_{doc}	Recall	nDCG@10	nDCG@20	P@10	P@20	bpref
PM2020	**XGPRec**	$0.45 \pm 0.38s$	**0.61**	0.30	0.30	0.33	0.30	0.30
	- BM25	$0.44 \pm 0.38s$	**0.61**	0.25	0.25	0.28	0.26	0.29
	- CoreOverlap	$\leq 0.1 \pm 0.0s$	**0.61**	**0.32**	**0.31**	**0.34**	**0.30**	**0.31**
	BM25 Title	1.1s	0.50	0.23	0.23	0.25	0.23	0.25
	BM25 Title + Abstract	9.3s	0.57	0.29	0.28	0.31	0.28	0.28
	PubMed Rec.	-	0.29	0.30	0.30	0.33	0.29	0.17

4 Implementation and Evaluation

We implement our recommendation algorithm by building upon our [10] discovery system's code base. The discovery system already maintains indexes that can be used to estimate the idf scores for nodes and edges and to perform the retrieval in the first stage through an inverted concept and an inverted edge index. For instance, an index (1M concepts, 40MB space) maps a concept to the number of documents in which the concept has been detected. Suppose a user enters a document ID through manual input or via a link to our system. In that case, we retrieve the document's data from the database, perform first-stage retrieval, select the k best-scored documents, and then retrieve the actual candidate document data for the recommendation. Thus, we load complete document data only if required. For the BM25 computation, we created a new BM25 index by utilizing PyTerrier [12] (a Python Wrapper around the well-known Terrier toolkit). For the final XGPRec score, we slightly prefer graph scores over text scores, i.e., we set $w_{graph} = 0.6$ and $w_{text} = 0.4$. We set predicate specificity scores (see tf-idf score for edges) based on each predicate's hierarchical level in the three-level predicate taxonomy (most-specific predicates received a score of 1.0, one level higher 0.5, and the highest level (only associated) 0.25) defined by the discovery system. We set $k = 1000$.

User Interface. Figure 1 shows a screenshot of our prototype. In our user interface the user can enter a document ID as an input. Then, a list of candidate documents is retrieved and ranked via our recommendation strategy. For each entry of that list, we generate an explanation and visualize it as shown in Figure 1. Our graph patterns should help users quickly determine the information scent of the recommendation list – a feature not available in other systems. For the explanation visualization, we tested different l values (no. of shown edges when generating an explanation) and found a maximum number of twelve suitable. So, we generate graph patterns that fit into the user interface concerning the available space.

4.1 Evaluation

The full evaluation of our system, including three benchmarks (**TREC Precision Medicine 2020** (PM2020) [15], **TREC Genomics 2005** (Genomics) [4] **RELISH** [2]), a detailed first stage evaluation, differences between XGPRec and PubMed, a user study is available in our technical report [8]. In this paper, we limit our reports to PM2020: (31 topics/1192 input documents) is a biomedical document retrieval test collection that asks for treatment options (drug), cancer forms (disease), and a gene variant (gene/target). We follow Zhang et al. [21] in selecting relevant articles (judged as 2 - relevant) per topic as input documents while considering all other articles belonging to the same topic as candidate documents with their judgments (2 - relevant/1 - partially relevant/0 - not relevant).

4.2 Recommender Evaluation

We decided to use FSConcept with a flexible cutoff as the first stage for our recommendation approach because (1) we do not need an additional index (as the node graph index for FSNode) and (2) FSConcept is less restrictive than FSNode (it just requires concepts to be annotated in documents and not that these concepts need to appear on the document graph). Table 1 shows the results of our recommendation approach (XGPRec) compared to the PubMed Recommender. In general, our recommendation strategies took less than 1s per document for the computation (see T_{doc} in Table 1). On PM2020, XGPRec achieved comparable nDCG and precision to the PubMed Recommender but nearly doubled recall. Another observation was that XGPRec without the CoreOverlap component achieved the highest scores, i.e., by just using the BM25 scores. In comparison, using only BM25 on titles or titles and abstracts of input documents instead of employing a recommender system achieves good results. The variant using abstracts unsurprisingly produces higher recall, nDCG, precision and bpref than the one using titles only. The comparably high execution time makes this strategy uneligible in a real system. Results produced by BM25 title are comparable to XGPRec except for the RELISH case.

5 Conclusion

This work extended our graph-based discovery system by an explainable paper recommendation component for a real-world digital library document collection. In contrast to many other works, our method (1) is unsupervised, i.e., we do not require training data and a library must thus not collect training data to implement a similar algorithm, and (2) it works on a real, large-scale collection with 37M documents. We argue that precise, concept-centric information needs are common in the biomedical domain, as seen in a PubMed query log analysis [5], PM2020 [15], our previous user study [10], or our discovery system's query log analysis [7]. In brief, our recommendation strategy XGPRec is fast in handling an extensive collection, offers a comparable performance to PubMed's real digital library recommendation system on concept-centric benchmarks, and provides users with suitable graph explanations. This research demonstrates how graph-based document representations allow beneficial exploration in digital libraries.

Acknowledgments

Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation): PubPharm – the Specialized Information Service for Pharmacy (Gepris 267140244).

References

[1] Krisztian Balog, Filip Radlinski, and Shushan Arakelyan. 2019. Transparent, Scrutable and Explainable User Models for Personalized Recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*. ACM, 265–274. https://doi.org/10.1145/3331184.3331211

[2] Peter Brown, RELISH Consortium, and Yaoqi Zhou. 2019. Large expert-curated database for benchmarking document similarity detection in biomedical literature search. *Database J. Biol. Databases Curation* 2019 (2019), baz085. https://doi.org/10.1093/DATABASE/BAZ085

[3] Andrew Collins and Jöran Beel. 2019. Document Embeddings vs. Keyphrases vs. Terms for Recommender Systems: A Large-Scale Online Evaluation. In *19th ACM/IEEE Joint Conference on Digital Libraries, JCDL 2019, Champaign, IL, USA, June 2-6, 2019*. IEEE, 130–133. https://doi.org/10.1109/JCDL.2019.00027

[4] William R. Hersh, Aaron M. Cohen, Jianji Yang, Ravi Teja Bhupatiraju, Phoebe M. Roberts, and Marti A. Hearst. 2005. TREC 2005 Genomics Track Overview. In *Proceedings of the Fourteenth Text REtrieval Conference, TREC 2005, Gaithersburg, Maryland, USA, November 15-18, 2005 (NIST Special Publication, Vol. 500-266)*. National Institute of Standards and Technology (NIST). http://trec.nist.gov/pubs/trec14/papers/GEO.OVERVIEW.pdf

[5] Jorge R. Herskovic, Len Y. Tanaka, William Hersh, and Elmer V. Bernstam. 2007. A Day in the Life of PubMed: Analysis of a Typical Day's Query Log. *Journal of the American Medical Informatics Association* 14, 2 (03 2007), 212–220. https://doi.org/10.1197/jamia.M2191

[6] Christin Katharina Kreutz and Ralf Schenkel. 2022. Scientific paper recommendation systems: a literature review of recent publications. *Int. J. Digit. Libr.* 23, 4 (2022), 335–369. https://doi.org/10.1007/s00799-022-00339-w

[7] Hermann Kroll, Christin Katharina Kreutz, Pascal Sackhoff, and Wolf-Tilo Balke. 2023. Enriching Simple Keyword Queries for Domain-Aware Narrative Retrieval. In *ACM/IEEE Joint Conference on Digital Libraries, JCDL 2023, Santa Fe, NM, USA, June 26-30, 2023*. IEEE, 143–154. https://doi.org/10.1109/JCDL57899.2023.00029

[8] Hermann Kroll, Christin K. Kreutz, Bill Matthias Thang, Philipp Schaer, and Wolf-Tilo Balke. 2024. Building an Explainable Graph-based Biomedical Paper Recommendation System (Technical Report). arXiv:identifier tba, PDF available at [cs.DL] https://github.com/HermannKroll/NarrativeRecommender/

[9] Hermann Kroll, Jan Pirklbauer, and Wolf-Tilo Balke. 2021. A Toolbox for the Nearly-Unsupervised Construction of Digital Library Knowledge Graphs. In *ACM/IEEE Joint Conference on Digital Libraries, JCDL 2021*. IEEE, 21–30. https://doi.org/10.1109/JCDL52503.2021.00014

[10] Hermann Kroll, Jan Pirklbauer, Jan-Christoph Kalo, Morris Kunz, Johannes Ruthmann, and Wolf-Tilo Balke. 2024. A discovery system for narrative query graphs: entity-interaction-aware document retrieval. *Int. J. Digit. Libr.* 25, 1 (2024), 3–24. https://doi.org/10.1007/S00799-023-00356-3

[11] Lisa Langnickel, Roman Baum, Johannes Darms, Sumit Madan, and Juliane Fluck. 2021. COVID-19 preVIEW: Semantic Search to Explore COVID-19 Research Preprints. In *Public Health and Informatics*. IOS Press, Amsterdam, the Netherlands, 78–82. https://doi.org/10.3233/SHTI210124

[12] Craig Macdonald and Nicola Tonellotto. 2020. Declarative Experimentation in Information Retrieval using PyTerrier. In *ICTIR '20: The 2020 ACM SIGIR International Conference on the Theory of Information Retrieval, 2020*. ACM, 161–168. https://doi.org/10.1145/3409256.3409829

[13] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, System Demonstrations*. The Association for Computer Linguistics, 55–60. https://doi.org/10.3115/v1/p14-5010

[14] David Mendez, Anna Gaulton, A Patrícia Bento, Jon Chambers, Marleen De Veij, Eloy Félix, María Paula Magariños, Juan F Mosquera, Prudence Mutowo, Michał Nowotka, María Gordillo-Marañón, Fiona Hunter, Laura Junco, Grace Mugumbate, Milagros Rodriguez-Lopez, Francis Atkinson, Nicolas Bosc, Chris J Radoux, Aldo Segura-Cabrera, Anne Hersey, and Andrew R Leach. 2018. ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Research* 47, D1 (11 2018), D930–D940. https://doi.org/10.1093/nar/gky1075

[15] Kirk Roberts, Dina Demner-Fushman, Ellen M. Voorhees, Steven Bedrick, and William R. Hersh. 2020. Overview of the TREC 2020 Precision Medicine Track. In *Proceedings of the Twenty-Ninth Text REtrieval Conference, TREC 2020, Virtual Event [Gaithersburg, Maryland, USA], November 16-20, 2020 (NIST Special Publication, Vol. 1266)*. National Institute of Standards and Technology (NIST). https://trec.nist.gov/pubs/trec29/papers/OVERVIEW.PM.pdf

[16] Harrisen Scells, Shengyao Zhuang, and Guido Zuccon. 2022. Reduce, Reuse, Recycle: Green Information Retrieval Research. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*. ACM, 2825–2837. https://doi.org/10.1145/3477495.3531766

[17] Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85. https://doi.org/10.1145/2629489

[18] Bangchao Wang, Ziyang Weng, and Yanping Wang. 2021. A Novel Paper Recommendation Method Empowered by Knowledge Graph: for Research Beginners. *CoRR* abs/2103.08819 (2021). arXiv:2103.08819 https://arxiv.org/abs/2103.08819

[19] Chih-Hsuan Wei, Alexis Allot, Robert Leaman, and Zhiyong Lu. 2019. PubTator central: automated concept annotation for biomedical full text articles. *Nucleic Acids Research* 47, W1 (05 2019), W587–W593. https://doi.org/10.1093/nar/gkz389

[20] Chih-Hsuan Wei, Hung-Yu Kao, and Zhiyong Lu. 2013. PubTator: a web-based text mining tool for assisting biocuration. *Nucleic Acids Research* 41, W1 (05 2013), W518–W522. https://doi.org/10.1093/nar/gkt441

[21] Li Zhang, Wei Lu, Haihua Chen, Yong Huang, and Qikai Cheng. 2022. A comparative evaluation of biomedical similar article recommendation. *J. Biomed. Informatics* 131 (2022), 104106. https://doi.org/10.1016/J.JBI.2022.104106