Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Olaf Liadal

Explainable Research Paper Recommendation Using Scientific Knowledge Graphs

Master's thesis in Computer Science
Supervisor: Krisztian Balog
June 2021

NTNU
Norwegian University of
Science and Technology

Olaf Liadal

Explainable Research Paper Recommendation Using Scientific Knowledge Graphs

Master's thesis in Computer Science
Supervisor: Krisztian Balog
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Abstract

Researchers can find themselves lost searching for relevant scientific literature in the large amounts that are published on a daily basis. The arXivDigest service for scientific literature recommendation and similar services are there to keep that from happening. This thesis explores new methods for the recommendation of scientific literature, using arXivDigest as a laboratory.

We introduce methods for explainable research paper recommendation that exploit the rich semantic information that is stored in scientific knowledge graphs. To enable these methods to access the information that is available about the researchers they are producing recommendations for, we also introduce methods that can be used to link researchers to appropriate entries in a scientific knowledge graph. Our methods have all been deployed and are running live on arXivDigest, where users are able to provide feedback on the recommendations they receive, and discovered potential links between users and entries in a scientific knowledge graph surface in a suggestion feature.

A user study shows that our recommendation methods are not much better at finding the literature that is relevant for users than the arXivDigest baseline recommender system itself. One of our methods does, however, appear to be better than the baseline when it comes to explaining recommendations. Ultimately, our methods only scratch the surface of what is possible, and graph-based research paper recommendation does show promise.

Samandrag

Vitskapsfolk kan føle seg fortapte i søk etter relevant vitskapleg litteratur blant dei store mengdene som publiserast på dagleg basis. Målet til arXivDigest, som er ein teneste for anbefaling av vitskapleg litteratur, og andre liknande tenester er at dette ikkje skal skje. Denne masteroppgåva utforskar nye metodar for anbefaling av vitskapleg litteratur, med arXivDigest som laboratorium.

Vi introduserer metodar for anbefaling av vitskapleg litteratur som nyttar seg av den rike semantiske informasjonen som er lagra i vitskaplege kunnskapsgrafar. For å gjere det mogleg for desse metodane å hente ut den informasjon som er tilgjengeleg om vitskapsfolka dei produserer anbefalingar for, introduserer vi også fleire metodar som kan brukast til å kople vitskapsfolk til oppføringar i ein vitskapleg kunnskapsgraf. Alle metodane våre har blitt tatt i bruk og køyrer hos arXivDigest, der brukarar kan gi tilbakemelding på anbefalingane dei mottek, og moglege koplingar mellom brukarar og oppføringar i ein vitskapleg kunnskapsgraf presenterast gjennom ein forslagsfunksjon.

Eit brukarstudie viser at metodane våre ikkje fungerer stort betre enn "baseline"-systemet til arXivDigest når det kjem til å finne den litteraturen som er relevant for brukarar. Éi av metodane våre ser likevel ut til å vere betre enn "baseline"-systemet når det kjem til å forklare anbefalingar. Metodane våre utforskar berre nokre få av tallause moglegheiter, og grafbaserte metodar for anbefaling av vitskapleg litteratur viser potensiale.

Acknowledgments

I would like to thank my supervisor, Professor Krisztian Balog, who has been very helpful and provided great guidance throughout the entire process of writing this thesis — on several occasions, at odd times of the day.

Contents

Chapter 1

Introduction

Consumers often struggle to find their way in seas of available products. It does not matter whether the consumer is a movie enthusiast looking for their next movie to watch or a researcher looking to stay on top of new scientific publications; unless the consumer knows just what they are looking for, finding the products that interest them can be a tiresome process of trial and error. This is where recommender systems come in. They aim to reduce the cognitive overload among consumers by guiding them towards potential products of interest. This thesis focuses on research paper (or scientific literature) recommender systems, addressing the needs of the researchers. We set aside movie enthusiasts and other consumer groups for this time.

ArXivDigest [19] is a scientific literature recommendation service. Researchers can register as users, specify the fields or topics they are the most interested in, and register links to their personal websites and profiles on other academic literature search and management services, such as DBLP, Google Scholar, and Semantic Scholar. Users of the service are allowed to register their own experimental recommender systems and submit recommendations through the arXivDigest API, through which it is also possible to retrieve information about the papers that are candidates for recommendation and access information about users, such as profile information, previous recommendations, and feedback on previous recommendations. All users consent to their information being "freely" available, and experimental recommender systems are free to use the information however they want (within reason). Existing arXivDigest recommender systems utilize the information that is available in many different ways, but there are significant chunks of information that have been left untouched: the information that is available about the users externally through personal websites and profiles at other service, and the information that is available about the papers that are candidates for recommendation through external sources. Of external services, Semantic Scholar, for which users can register links to their profiles, is of most interest to us. This service also happens to be a great source of additional information about candidate papers.

Table 1.1: Percentages of arXivDigest users who have registered links to their personal websites and profiles at external academic literature search and management services.

	DBLP	Google Scholar	Semantic Scholar	Personal website
Users (%)	31	39	21	35

Semantic Scholar is a project by the Allen Institute for AI which applies artificial intelligence to automate the task of extracting meaning from scientific literature [37]. The project has processed huge amounts of literature since its inception, and the extracted data about authors and papers, which is organized in a scientific knowledge graph, is available through the Semantic Scholar API. The graph includes basic information, such as author names, paper titles, abstracts, venues, and years of publication, as well as the relations that exist between authors and papers, such as authorship and citations. We have previously proposed a method for the extraction of publication metadata from academic homepages [30]. The ultimate goal was to build an experimental arXivDigest recommender system exploiting the information extracted from the personal websites of users. Developing an extraction solution that can extract information from an academic homepage that is comparable in quality and quantity to the information available about the owner of the page and their publications through Semantic Scholar is a difficult task. For the purpose of developing arXivDigest recommender systems, this task might even be an unnecessary distraction, as most of the users are researchers, and most researchers have profiles on Semantic Scholar.

Has Krisztian Balog published a paper at TREC in the last two years? How many publications has he authored together with Maarten de Rijke? How many times has he been cited by Donald Knuth? Do his publications tend to have large impacts on later publications by other researchers? Answers to these types of questions can be found with access to the Semantic Scholar profile of an author. They all reflect the interests of the author in some way and can, if the author is an arXivDigest user, be used to find publications that are likely to be relevant for them. Unfortunately for us, as seen in Table 1.1, only 21 % of arXivDigest users have registered their Semantic Scholar profiles.

1.1 Problem Definition

This thesis focuses on the development of experimental arXivDigest recommender systems that utilize scientific knowledge graphs as sources of information about users and candidate papers. The specific scientific knowledge graph used is the scientific literature graph of Semantic Scholar. For the systems to be able to produce recommendations for a user, the Semantic Scholar author ID of the user (which is contained in the link to the user's Semantic Scholar profile) must be known. We have seen that only 21 % of arXivDigest users have registered links to their Semantic Scholar profiles. This percentage is quite low. We are, therefore, going to try to find the correct Semantic Scholar author IDs for as many as possible of the remaining users ourselves.

We are left with two objectives: finding the missing Semantic Scholar author IDs of users and developing experimental recommender systems. More formally:

1. Given the information that is available about an arXivDigest user, produce a ranking of potential Semantic Scholar profile candidates for the user.
2. Given the information that is available about an arXivDigest user through both arXivDigest and the scientific literature graph of Semantic Scholar, produce a ranking of the papers that are candidates for recommendation at arXivDigest, including explanations for the rank of each paper.

1.2 Research Questions

This thesis will attempt to answer the following research questions:

RQ1 How can an arXivDigest user be linked to an appropriate entry (or author node) in a scientific knowledge graph?

RQ2 In what ways can the information that is available about an arXivDigest user through an external scientific knowledge graph be used to produce scientific literature recommendations for the user?

RQ3 In what ways can a scientific knowledge graph be used as an external source of information about papers that are candidates for recommendation at arXivDigest when producing scientific literature recommendations? Is it feasible to get hold of the necessary information in reasonable time?

The first question will be answered by implementing two methods that search through the Semantic Scholar Open Research Corpus (S2ORC) dataset [31] and look for Semantic Scholar author profiles that match an arXivDigest user in different ways. The simplest method searches for the user's name and ranks the authors present in the search results based on their frequencies of occurrence. The other method, which performed the best and was used to generate profile suggestions for users, searches through the dataset using both the user's name and topics of interest and ranks the authors present in the search results using the BM25 ranking model.

The remaining questions will be answered by analyzing and implementing several recommendation methods that (to different extents) use information from Semantic Scholar about users and their published papers, collaborators of users (or co-authors) and their published papers, candidate papers, and authors of candidates papers and their published papers:

- One method ranks candidate papers based on their venues of publication. The more papers the user has published at the candidate paper's venue of publication, the higher the candidate paper is ranked.
- Two methods look at venue co-publishing between the authors of candidate papers and the user. These methods rank candidate papers whose authors publish at the same venues as the user the highest. The first method considers only venue co-publishing in its ranking process. The second one also takes into consideration the influence of the papers published by the candidate paper authors, where the influence of a paper is represented by its *influential citation count* — a metric that is derived from the paper's pure citation count and is supposed to better reflect the paper's influence on citing papers [40].
- Two methods look at citations. The first one ranks candidate papers based on the degree to which the user has previously cited their authors. The second one looks to the collaborators of the user and the degree to which they have previously cited the authors of candidate papers.
- The last method combines the approach of the first citation-based method with the approach of the baseline arXivDigest recommender system, which uses Elasticsearch to rank candidate papers using the topics of interest of the user.

1.3 Main Contributions

The main contributions of this thesis can be summed up in four points:

- Algorithms for linking users to entries in a scientific knowledge graph.
- Novel algorithms for research paper recommendation that exploit information stored in scientific knowledge graphs.
- Efficient implementations of these algorithms, deployed live on arXivDigest.
- Experimental methods for evaluation of explainable recommendations.

1.4 Outline

The remainder of this thesis is structured as follows:

Chapter 2 introduces the field of academic search and several academic search tasks, including research paper recommendation, and describes the arXivDigest service and platform, and scientific knowledge graphs.

Chapter 3 goes more in depth in research paper recommendation and explainable recommendation.

Chapter 4 describes methods that can be used to link users to entries in a scientific knowledge graph.

Chapter 5 explores ways to use the data that is stored in scientific knowledge graphs to produce scientific literature recommendations.

Chapter 6 concludes this thesis by answering the research questions and discussing future work.

Chapter 2

Background

This chapter starts with an introduction of the field of academic search, including descriptions of common academic search tasks and evaluation approaches. After this, we discuss the arXivDigest platform for personalized scientific literature recommendation, which will act as our laboratory in later chapters. Lastly follows a description of scientific knowledge graphs, including a description of Semantic Scholar and its scientific literature graph.

2.1 Academic Search

Academic search is a field within information retrieval that focuses on the retrieval of scientific data. This section starts by introducing five academic search tasks. A common aspect of tasks in information retrieval, in general, is the need to evaluate and compare different approaches. This need can be addressed in several ways; one way, which is becoming increasingly prevalent, is the use of centralized evaluation infrastructures. This section rounds off with descriptions of some of the evaluation infrastructures that have been deployed for use in the context of academic search tasks.

2.1.1 Academic Search Tasks

We now describe the research paper search, research paper recommendation, collaborator discovery, expert finding, and reviewer assignment academic search tasks. At the end of the section, we list other, similar tasks. The descriptions are based on the surveys conducted by Beel et al. [8] and Khan et al. [26] of the research paper recommendation field and scholarly data, respectively.

Research Paper Search

Research paper search, in the context of academic search engines, deals with the problem of calculating the relevance of research papers given a search query, and producing rankings of papers based on their relevance [8]. This type of search is often referred to as ad hoc search. Semantic Scholar, which will be further introduced in Section 2.3.1, and CiteSeerX [29] are examples of academic search engines. The TREC OpenSearch challenge [23] is an example of a research paper search task which allows participants to develop and test their own retrieval methods with real traffic provided by CiteSeerX and another search engine called SSOAR. The TREC OpenSearch experimental platform and evaluation infrastructure will be described in Section 2.1.2.

Research Paper Recommendation

Research paper recommendation is similar to research paper search, but instead of calculating the relevance of papers given an explicit search query, relevance is calculated based on context, using traditional (or less so) recommendation methods, such as stereotyping, collaborative filtering and content-based filtering [8]. *Citation recommendation* is another similar task. Both research paper and citation recommendation deal with producing recommendations based on information available in a certain context. In research paper recommendation, the context is the user of the system (as in [19]); in citation recommendation, the context is, e.g., a paper draft [10] or a context of words [16].

Beel et al. [8] found, in their survey of the research paper recommendation field in 2015, that each of the recommendation approaches described in the existing literature were based on one of several different methods: stereotyping, content-based filtering, collaborative filtering, co-occurrence, graph-based recommendation, global relevance, and hybrid methods. Section 3.1 will elaborate on these methods.

Collaborator Recommendation

Collaborator recommendation (or *similar researcher search*) concerns the recommendation of potential collaborators for researchers [26]. CollabSeer [13] is a collaborator recommendation service which finds potential collaborators for researchers based on collaborator networks and research interests. ScholarSearch [21] is another collaborator recommendation system, which acts as a search engine. Given the name of a researcher as a query, it finds potential collaborators for that researcher by ranking the expertise profiles of other researchers against their expertise profile. The expertise profiles are based on data extracted from publications and academic homepages.

Expertise Retrieval

Expert finding deals with finding people with knowledge about a given topic [6]. This task is quite similar to collaborator recommendation, but instead of ranking researchers based on their similarity to another researcher, researchers are ranked based on their estimated level of expertise in a query topic. The "inverse" task of expert finding is *expert profiling*, which deals with the problem of identifying the topics of expertise of researchers.

Reviewer Assignment

The *reviewer assignment* problem deals with the automatic assignment of reviewers to papers submitted to conferences [15]. The techniques used to solve this problem are very similar (and often identical) to the techniques used in research paper recommendation [8]. In research paper recommendation, small selections of papers are to be picked from a large corpus and recommended to a large collection of users; in the reviewer assignment problem, all the papers in a relatively small corpus are to be picked and assigned to a small collection of reviewers.

Other Tasks

Other academic search tasks include book recommendation [32], academic news feed recommendation [14], academic event recommendation [27], venue recommendation [42], and academic dataset recommendation [38]. The field of scientometrics, which deals with analyzing the impact of researchers, research papers, and the links between them [8], is also highly relevant to academic search.

2.1.2 Evaluation Infrastructures for Academic Search Tasks

There are three main approaches to information retrieval system evaluation: user studies [25], online evaluations [22], and offline (or test collection based) evaluations [35]. User studies measure user satisfaction through explicit ratings provided by users. They are considered the optimal evaluation approach, and should generally have at least two dozen participants in order for the results to have any significance [8]. Online evaluations measure acceptance rates of retrieval results in real-world systems using explicit measures, such as click-through rate (CTR). Offline evaluations use metrics such as average precision and reciprocal rank to evaluate retrieval results against some ground truth. They are much more convenient to conduct than user studies and online evaluations, but are also much less useful, as there is a disconnect between user satisfaction and system accuracy — often, there is little to no correlation between the results from user studies and online evaluations and offline evaluations.

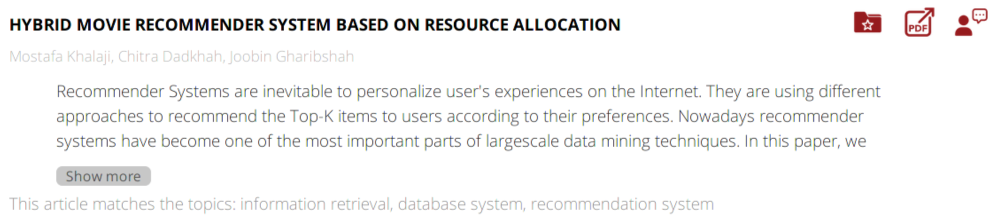
In their survey of the research paper recommendation field in 2015, Beel et al. [8] found several shortcomings. A highlighted shortcoming was a neglect of user modeling and user satisfaction, and instead a large focus on offline evaluations. In later years, online evaluations and evaluation infrastructures have gained traction. Evaluation infrastructures are (typically) cloud-based systems that allow external actors to evaluate their own retrieval algorithms [36]. The practice is commonly referred to as Evaluation-as-a-Service (EaaS). In their review of state-of-the-art evaluation infrastructures for academic shared tasks, Schaible et al. [36] list three important requirements for evaluation infrastructures: (1) the possibility to perform both online and offline evaluations, (2) domain specificity in evaluations (users of academic retrieval systems are experts, and behavioral patterns depend on the field), and (3) reproducibility.

Many shared tasks in academic search utilize evaluation infrastructures in order to simplify participation and standardize evaluations. In the TREC OpenSearch challenge [23], participants are given access to an existing search engine and are able to replace components of it with their own implementations. The participants produce rankings of candidate documents for a set of queries that are expected to be issued frequently by the search engine's users. These rankings are interleaved with the search engine's own rankings, and performance is measured in terms of impressions and CTR. Another evaluation infrastructure used for academic search tasks is STELLA [11]. The arXivDigest platform [19] for scientific literature recommendation, although not used in any shared tasks, also works as an evaluation infrastructure.

2.2 ArXivDigest

ArXiv[1] is an open-access archive for scientific articles within many fields. The service offers access to millions of articles but no simple way to explore them. Several services exist that try to make it easier for the user to browse arXiv and find relevant articles. One example is Arxiv Sanity Preserver[2], which offers article suggestions and recommendations in addition to revamped and slightly more user-friendly versions of many of the features arXiv itself offers, such as overviews of recent and new articles and search. Another example is arXivDigest [19].

[1]https://arxiv.org/
[2]http://www.arxiv-sanity.com/


HYBRID MOVIE RECOMMENDER SYSTEM BASED ON RESOURCE ALLOCATION

Mostafa Khalaji, Chitra Dadkhah, Joobin Gharibshah

Recommender Systems are inevitable to personalize user's experiences on the Internet. They are using different approaches to recommend the Top-K items to users according to their preferences. Nowadays recommender systems have become one of the most important parts of largescale data mining techniques. In this paper, we

This article matches the topics: information retrieval, database system, recommendation system

Figure 2.1: Recommendation shown on the arXivDigest website.

ArXivDigest [19] is a living lab for explainable personalized scientific literature recommendation. The platform allows users to register and submit recommendations with their own experimental recommender systems, and offers the owners of systems access to evaluation metrics and feedback from users. The recommendations that are displayed to a user on a particular day is an interleaving of the recommendations submitted by all registered systems, produced by selecting the top-k recommended papers (that have been published during the last week) from each system. Users can access their recommendations in two ways. They have the option to subscribe to daily or weekly digest emails, which contain summaries of the papers they have been recommended in the last day or week, or they can view all their recommendations on the arXivDigest website. A recommendation displayed in the web interface of arXivDigest is shown in Fig. 2.1.

2.2.1 The arXivDigest API

Experimental recommender systems submit their own recommendations through the arXivDigest API. The API requires a system's unique API key to be present in the HTTP headers of all requests made by the system. This API key is obtained by registering the system on the arXivDigest website. For registered systems, the recommendation process looks like this:

1. Retrieve API settings, such as user batch size, from `/`.
2. Retrieve the arXiv IDs of the papers that are candidates for recommendation from the `/articles` endpoint. Additional information about each paper can be retrieved from the `/article_data?article_id=[Paper ID]` endpoint.
3. Retrieve the IDs for a batch of users from the `/users?from=[Start ID]` endpoint. The `from` query parameter can be incremented by the user batch size in order to get the next batch of users.
4. Retrieve additional information about the users in the batch from the `/user_info?ids=[User IDs]` endpoint.

5. Retrieve the arXiv IDs of the papers that have already been recommended and displayed for each user in the batch from the `/user_feedback/articles?user_id=[User IDs]` endpoint.
6. Assign a score to each candidate paper together with an explanation for the score, filter out the papers that have already been recommended, and submit the top papers for each user to the `/recommendations/articles` endpoint.
7. Repeat steps 3 to 6 until recommendations have been submitted for all users.

ArXivDigest scrapes arXiv for new papers around midnight each weekday, and the API accepts recommendations from experimental systems between 00:30 and 03:00 on the same days. The IDs that are exposed by the `/articles` endpoint are those of the papers that have been scraped from arXiv during the last week.

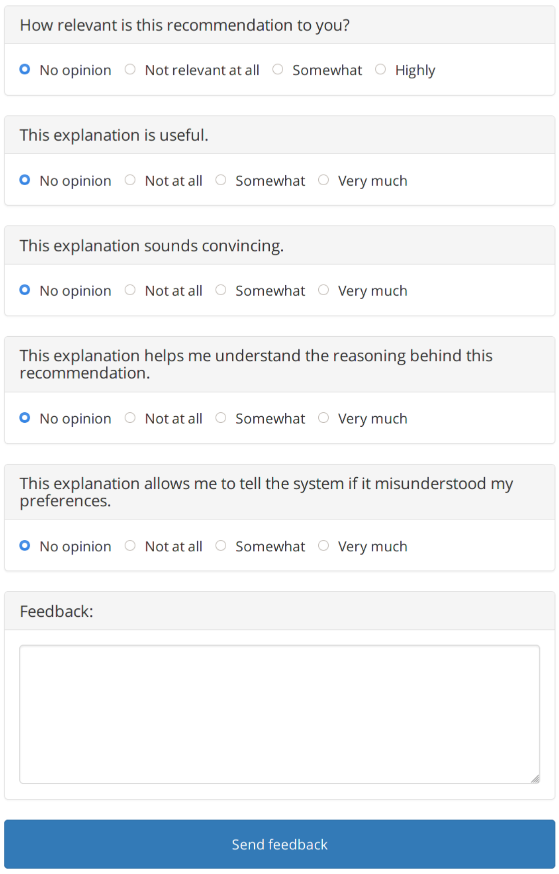
2.2.2 Baseline Recommender System

The baseline recommender system implemented by Gingstad, Jekteberg, and Balog uses Elasticsearch to score papers. The system indexes the papers that are candidates for recommendation in Elasticsearch, queries the index for each of the user's topics of interest, chooses the top-k topics for each paper based on the relevance scores returned by the index, and assigns scores to the papers equal to the sum of their relevance to the top-k topics.

2.2.3 Evaluation Methodology

Central to the evaluation methodology of arXivDigest are *impressions* and the notion of *reward*. An impression is an interleaving of recommendations from multiple systems that has been seen and potentially interacted with by the user. Multiple user interactions can be associated with a single impression, and different types of interactions result in different amounts of reward points: the user saving a recommended paper in their library gives 5 points, clicking a recommended paper on the website or in an email gives 3 points, and seeing a recommended paper on the website or in an email gives 0 points. Given an interleaving of recommendations, the reward of a system equals the sum of the reward points resulting from all the user interactions with the recommendations submitted by the system. The *normalized reward* of the system is equal to the system's reward divided by the total reward of all the systems in the interleaving, such that the normalized rewards of all the systems add up to 1.

The performance of systems is monitored continuously, and system owners can see how the numbers of impressions and the *mean normalized rewards* of their systems progress over time on the arXivDigest website. The mean normalized reward of a system is the



How relevant is this recommendation to you?

[✓ No opinion] Not relevant at all Somewhat Highly

This explanation is useful.

[✓ No opinion] Not at all Somewhat Very much

This explanation sounds convincing.

[✓ No opinion] Not at all Somewhat Very much

This explanation helps me understand the reasoning behind this recommendation.

[✓ No opinion] Not at all Somewhat Very much

This explanation allows me to tell the system if it misunderstood my preferences.

[✓ No opinion] Not at all Somewhat Very much

Feedback:

Send feedback

Figure 2.2: Recommendation feedback form on the arXivDigest website.

mean of the system's normalized rewards for each interleaving in a selected period of time.

In addition to the types of explicit and implicit user feedback used to calculate system reward, users can also provide detailed feedback on specific recommendations and/or their explanations through a form, which is shown in Fig. 2.2. Informed by [4], this form asks the user about the *relevance* of the recommendation and how *satisfactory*, *persuasive*, *transparent*, and *scrutable* they find the explanation [19].

2.3 Scientific Knowledge Graphs

Most digitally published scientific articles are nothing more than analogues of their physical counterparts [41]. Organizing scientific knowledge (or scholarly data) in semantic, interlinked graphs is a more structured and machine-readable alternative to the current mostly document-oriented approach [3]. Scientific knowledge graphs have garnered attention from many teams of researchers, such as the ScienceGRAPH project with the Open Research Knowledge Graph [24], the Microsoft Academic Knowledge Graph [18], and Semantic Scholar and its scientific literature graph [2].

2.3.1 Semantic Scholar

Semantic Scholar is a team of researchers at the non-profit Allen Institute for AI working on reducing information overload in the scientific community by applying artificial intelligence to extract meaning from scientific literature [37]. Since the launch of Semantic Scholar in 2015, more than 180 million papers have been added to the project's scientific literature graph, which can be explored through a search engine available at the project's website[3], the Semantic Scholar API, and the Semantic Scholar Open Research Corpus (S2ORC) [31].

The Scientific Literature Graph

Semantic Scholar's scientific literature graph is constructed using a combination of traditional natural language processing techniques, such as sequence labeling, entity linking, and relation extraction [2]. The graph contains several types of nodes: papers, authors, entities representing unique scientific concepts, and entity mentions representing textual references of entities in papers. Citation edges exist between paper nodes, authorship edges exist between author and paper nodes, entity linking edges exist between entity mentions and entities, mention-mention edges exist between mentions of entities occurring in the same contexts, and entity-entity edges exist between related entities.

PDFs and metadata of papers are obtained by Semantic Scholar through partnerships with publishers, pre-publishing services, and web crawling. The metadata provided by the paper sources is often incomplete, and the papers obtained through web crawling have no associated metadata at all. A system based on recurrent neural networks is used to extract all missing metadata from the paper PDFs [2]. This system extracts titles, lists of authors, and lists of references, where each reference contains a title, a list of authors, a venue, and a year. Once the metadata of a paper is complete, a paper node and nodes for its authors are added to the literature graph (if not already present) together with citation and authorship edges. Duplicate papers are detected and removed based on metadata similarity. Once the paper and author nodes and citation and authorship edges are in place for a paper, entities and entity mentions are extracted and linked

[3]`https://www.semanticscholar.org/`

```
{
    "aliases": ["K. Balog", "Krisztian Balog"],
    "authorId": "1680484",
    "dblpId": null,
    "influentialCitationCount": 336,
    "name": "K. Balog",
    "papers": [
        {
            "paperId" :"fd26c7254eb81124148e84e3cf02dbd88bbc5623",
            "title" :"Formal models for expert finding in enterprise corpora",
            "url" :"https://www.semanticscholar.org/paper/fd26c7254eb81124148e84e3cf02d⌋
            ↪  bd88bbc5623",
            "year": 2006
        }
    ],
    "url": "https://www.semanticscholar.org/author/1680484"
}
```

Listing 2.1: Response JSON data from the Semantic Scholar API for the author with ID 1680484. The `papers` property has been truncated due to its length.

using a system that combines statistical models, hand-engineered, deterministic rules, and off-the-shelf entity-linking libraries.

The Semantic Scholar API

Semantic Scholar offers access to its collection of author and paper records through a RESTful API[4]. The API has two endpoints: the `/author/[Author ID]` endpoint for author data and the `/paper/[Paper ID]` endpoint for paper metadata. Authors are accessed using their Semantic Scholar author IDs, and papers can be accessed using several different types of ID, such as arXiv and Semantic Scholar paper IDs.

Listing 2.1 shows the JSON response data for the author with ID 1680484, Krisztian Balog. The response contains, among other things, the author's name and aliases, brief metadata for each paper published by the author, and the author's *influential citation count*, which is the sum of the influential citation counts of all their published papers. The influential citation count of a paper is inferred by a machine learning model and is supposed to reflect a paper's influence — the degree to which the paper is used or extended by citing papers — more precisely than a pure citation count [40].

[4]Documentation available at `https://api.semanticscholar.org`.

Listing 2.2 shows the JSON response data for the paper listed in the `papers` property in Listing 2.1. The response contains, among other things, the paper's title, abstract, authors, citing papers, referenced papers, year of publication, venue, topics, and influential citation count.

The Semantic Scholar Open Research Corpus

Semantic Scholar's paper records are also available through the Semantic Scholar Open Research Corpus (S2ORC) dataset [31]. In 2019, this dataset contained 81.1 million records — 8.1 million of which included machine-readable full text extracted from paper PDFs. The size of the dataset has since increased and continues to increase as periodic (often monthly) updates are released. An example paper record from the dataset is shown in Listing 2.3. It is similar but not identical in structure (and in the naming of properties) to the paper records returned by the API.

```
{
    "abstract": "Searching an organization's document repositories for experts...",
    "arxivId": null,
    "authors": [
        {
            "authorId": "1680484",
            "name": "K. Balog",
            "url": "https://www.semanticscholar.org/author/1680484"
        }
    ],
    "citationVelocity": 25,
    "citations": [],
    "corpusId": 8226656,
    "doi": "10.1145/1148170.1148181",
    "fieldsOfStudy": ["Computer Science"],
    "influentialCitationCount": 56,
    "isOpenAccess": true,
    "isPublisherLicensed": true,
    "numCitedBy": 652,
    "numCiting": 21,
    "paperId": "fd26c7254eb81124148e84e3cf02dbd88bbc5623",
    "references": [],
    "title": "Formal models for expert finding in enterprise corpora",
    "topics": [],
    "url": "https://www.semanticscholar.org/paper/fd26c7254eb81124148e84e3cf02dbd88bbc5
    ↪  623",
    "venue": "SIGIR",
    "year": 2006
}
```

Listing 2.2: Response JSON data from the Semantic Scholar API for the paper with Semantic Scholar ID fd26c7254eb81124148e84e3cf02dbd88bbc5623. The `authors` and `abstract` properties have been truncated and the `citations`, `references`, and `topics` properties have been emptied due to their lengths.

```json
{
    "id": "38f271d026ff9c20042ca8b49588f6cee0d6bd2a",
    "title": "Building A Vietnamese Dialog Mechanism For V-DLG~TABL System",
    "paperAbstract": "This paper introduces a Vietnamese automatic dialog...",
    "authors": [
        { "name": "An Hoai Vo", "ids": ["66339053"] },
        { "name": "Dang Tuan Nguyen", "ids": ["1748994"] }
    ],
    "inCitations": [],
    "outCitations": [],
    "year": 2014,
    "s2Url":
    ↪  "https://semanticscholar.org/paper/38f271d026ff9c20042ca8b49588f6cee0d6bd2a",
    "sources": [],
    "pdfUrls": ["http://airccse.org/journal/ijnlc/papers/3114ijnlc04.pdf"],
    "venue": "",
    "journalName": "",
    "journalVolume": "3",
    "journalPages": "31-42",
    "doi": "10.5121/IJNLC.2014.3104",
    "doiUrl": "https://doi.org/10.5121/IJNLC.2014.3104",
    "pmid": "",
    "fieldsOfStudy": ["Computer Science"],
    "magId": "2327911789",
    "s2PdfUrl": "",
    "entities": []
}
```

Listing 2.3: JSON paper record from the S2ORC dataset.

Chapter 3

Related Work

This chapter takes a closer look at the different classes of methods described in the existing literature within the field of research paper recommendation, and introduces explainable recommendation — personalized recommendation where recommendations are accompanied by the reasoning behind them.

3.1 Research Paper Recommendation

In their survey of the research paper recommendation field in 2015, Beel et al. [8] used seven different classes to classify the recommendation methods described in the existing literature: stereotyping, content-based filtering, collaborative filtering, co-occurrence recommendation, graph-based recommendation, global relevance, and hybrid recommendation approaches. Hundreds of papers have been published in the field since the 1990s, and there is no clear evidence of any class being better than the others [8]. In fact, each of the recommendation classes has been shown to perform best in at least one evaluation. This section describes and compares these recommendation classes.

3.1.1 Stereotyping

Stereotyping is one of the early recommendation classes, first used by Rich [34] in a recommender system for novels. Inspired by the stereotypes used in psychology to classify people based on limited information, the Grundy recommender system classified users based on collections of frequently occurring characteristics among the users. One of the biggest drawbacks of stereotyping is pigeonholing of users: users are assigned stereotypes that already exist in the system, no matter how well or bad their characteristics match any of the stereotypes [8].

Few have applied stereotyping for research paper recommendation. Beel et al. [7] used a stereotyping approach as a baseline and fallback in their reference management system Docear when other approaches failed to produce recommendations. They reported mediocre performance, with a CTR of 4 % — versus a CTR greater than 6 % for their content-based filtering approaches.

3.1.2 Content-Based Filtering

Content-based filtering is the most widely used recommendation class for research paper recommendation [8]. Items are represented by their prominent features, which are based solely on their contents (e.g., *n*-grams or tokens if the items are text documents), and users are recommended items that are similar to the ones they have already expressed interest in. Features are often represented using the vector space model, and similarity between documents is calculated using, e.g., the cosine similarity. Items that the user is interested in are typically items that the user has saved or liked in some way. In the case of research paper recommendation, papers authored or cited by the user and papers the user has other types of relations to could also be considered items of interest.

3.1.3 Collaborative Filtering

Collaborative filtering, as it is known today, was introduced by Konstan et al. [28]. Instead of recommending items that are similar to items the user has already expressed interest in, like content-based filtering does, collaborative filtering recommends items that like-minded users have expressed interest in [8]. Like-minded users are users that have rated items similarly. When either of two like-minded users expresses interest in an item by rating it positively, that item is recommended to the other user. Collaborative filtering depends on the ratings provided by users, but users often lack the motivation to provide ratings of meaningful volume. This is often referred to as the cold-start problem. Another challenge associated with collaborative filtering, especially for research paper recommendation, is sparsity: the number of items can be very high compared to the number of users.

3.1.4 Co-Occurrence Recommendation

Co-occurrence recommendations are produced by recommending items that frequently appear together with some source items in some way [8]. An advantage of co-occurrence recommendation over content-based filtering is the focus on relatedness — how coupled items are — instead of similarity. The co-occurrence of items can mean many different things. Small [39] introduced the co-citation measure for research papers. His idea was that the relatedness of two papers would be reflected by the frequency of them appearing together in the bibliographies of other papers. Small's idea of co-citation was further developed to take into consideration the proximity of citations within the body texts of papers and used for research paper recommendation by Gipp and Beel [20]. Other approaches for research paper recommendation based on co-occurrence have looked at how often papers are co-viewed during browsing sessions [8].

3.1.5 Graph-Based Recommendation

Graph-based recommendation exploits the inherent connections that exist between items [8]. In the context of research paper recommendation, the connections between items are used to construct graphs that show, e.g., how papers are connected by citations, as in the scientific literature graph of Semantic Scholar, which was described in Section 2.3.1. Edges in a graph can also represent connections that are not inherent to items, such as the co-citation strength or text similarity of papers. Typically, graph-based methods for research paper recommendation take one or several papers as input and perform random walks to find relevant papers in their graphs [8].

3.1.6 Global Relevance

Recommendation based on *global relevance* does not take into consideration the specific characteristics of each user, but assumes that generally popular items are likely to be of interest to the user [8]. No research paper recommendation approaches are exclusively built on this idea, but several have used global popularity metrics as additional ranking factors for recommendations produced with other methods. Some of these approaches use content-based filtering to first produce user-specific recommendations and then use global metrics (such as citation counts, venues' citation counts, citation counts of user affiliations, and paper age) as weights for the recommendations.

3.1.7 Hybrid Approaches

The recommendation methods classified as *hybrid approaches* combine the other six recommendation classes in different ways. Of all existing research paper recommendation methods, many have hybrid characteristics, but few are true hybrids — i.e., most have a primary recommendation approach and few rely equal parts on different approaches [8]. One graph-based method with hybrid characteristics mentioned by Beel et al. draws inspiration from content-based filtering methods and includes terms extracted from paper titles in its graph. The methods referred to in Section 3.1.6 that combine content-based filtering and global metrics also have hybrid characteristics.

3.2 Explainable Recommendation

For certain types of recommender systems, such as ones based on latent factor models, it can be hard to explain why a specific item has been recommended to the user beyond simply saying that the recommended item was assigned a higher score than other items by the system [43]. The focus of explainable recommendation is to develop transparent recommender systems with increased persuasiveness, effectiveness, trustworthiness, and user satisfaction. In their survey on explainable recommendation, Zhang and Chen [43] adopt a two-dimensional taxonomy to categorize explainable recommendation methods. The first dimension is the information source or format of explanations — the information used to produce and the way in which explanations are conveyed to the user. The second dimension is the model that is used to produce explanations.

3.2.1 Explanation Information Sources and Formats

In the early stages of explainable recommendation, systems based on collaborative filtering explained their recommendations to the user by simply telling them that the recommended items were similar to some other items the user had rated highly (or that similar users had rated the recommended items highly). Zhang and Chen [43] refer to this type of explanation as *relevant-item explanation* (or *relevant-user explanation*). Another type is *feature-based explanation*. Explanations of this type are produced similarly to content-based (filtering) recommendations. One way to produce feature-based explanations is to tell the user which of the recommended item's features match the user's profile, which is made up of the items that the user has expressed interest in earlier. *Opinion-based explanation* is another type. Explanations of this type are either *aspect-level* or *sentence-level*. Aspect-level explanation is similar to feature-based explanation, except that aspects (such as color and quality) are usually not directly available in items or user profiles, but are instead extracted or learned by the recommendation model [43]. Sentence-level explanation can be further divided into *template-based* and *generation-based explanation*. Template-based sentence explanation relies on predefined sentence templates, which are filled in to produce personalized explanations for the user. This is is the approach used by the baseline arXivDigest recommender system described in Section 2.2.2. Generation-based sentence explanation does not use templates but instead generates explanations automatically using, e.g., machine learning models trained on user review corpora [43]. Other types of explanation is *visual explanation*, which conveys explanations using, e.g., images with or without highlighted areas of interest, and *social explanation*, which provides explanations based on the social relations of the user.

3.2.2 Explainable Recommendation Models

Explainable recommendation is either *model-intrinsic* or *model-agnostic* (or *post hoc*) [43]. Model-intrinsic approaches use models that are based on transparent decision making and are inherently explainable. In model-agnostic approaches, the decision making is more of a black box, and explanations are produced after-the-fact by separate explanation models.

Model-Intrinsic Explainable Recommendation

The use of collaborative filtering described in Section 3.2.1 is one example of model-intrinsic explainable recommendation. Due to the difficulties associated with explaining recommendations produced using latent factor models, Zhang et al. [44] introduced explicit factor models for model-intrinsic explainable recommendation, based on the idea of tracking the favorite features (or aspects) of the user and recommending items that perform well on these features [43]. Knowledge graphs have also been used for explainable recommendation. Catherine et al. [12] used a Personalized PageRank algorithm to jointly rank movies and entities (actors, genres, etc.) in a way that allowed the entities to serve as explanations. Ai et al. [1] adopted the use of knowledge graph embeddings learned over a graph containing different types of user, item, and entity relations, such as item purchases made by users [43]. Their approach recommended items for purchase based on their similarity to already purchased items, and explanations could be produced by finding the shortest path between the user and recommended items in the graph. Rule mining has also been used for explainable recommendation. Balog, Radlinski, and Arakelyan [5] proposed a set-based user modeling approach, which allowed for natural language explanations to easily be formed based on the preferences captured by the user models. Many other model-intrinsic explainable recommendation approaches based on, e.g., topic modeling and deep learning have also been proposed [43].

Model-Agnostic Explainable Recommendation

If the model used to produce recommendations is too complex to explain, explanations can be produced post hoc [43]. In some cases, simple statistical information about the recommended items is adequate. As an example, an e-commerce system might explain a recommendation post hoc with “this item has been bought by five of your friends”. Post hoc explanations have also been produced using association rule mining. Peake and Wang [33] treated the recommendation model — in their case, one based on latent factor models — as a black box, and trained association rules on model transactions — pairs of model input (a user model) and output (recommendations). The learned association rules were then used to explain the recommendations produced by the recommendation model (and could also be used to produce the same recommendations). Many other methods have also been used for post hoc explainable recommendation [43].

Chapter 4

Linking Users to a Scientific Knowledge Graph

This chapter addresses the first problem defined in Section 1.1 and explores methods that can be used to link users to appropriate entries in a scientific knowledge graph. Chapter 1 mentioned that only a small minority of arXivDigest users have registered their Semantic Scholar profiles. The users in this group are, conveniently, linked to the appropriate author nodes in the scientific literature graph of Semantic Scholar. The remaining majority of users is not. The methods of this chapter surface on the arXivDigest platform as a suggestion feature for Semantic Scholar profiles. The ultimate goal is to increase the number of users with links to the correct scientific knowledge graph entries, so that the rich semantic information stored there can be exploited for generating scientific literature recommendations in Chapter 5.

4.1 Methods

This section starts with a formal description of the *profile matching* task. The S2ORC dataset introduced in Section 2.3.1 serves as the data foundation for the methods of this chapter. After defining the profile matching task, we describe how we simplify and optimize searching through the large amounts of data in this dataset and how the data is used by different methods for profile matching.

4.1.1 Problem Statement

We define profile matching to be the task of producing a ranking of the author nodes in a scientific knowledge graph based on their likelihoods of representing the same person as an arXivDigest user. If we let V_A denote the set of author nodes in a scientific knowledge graph, then the likelihood that any author node $a \in V_A$ represents the same person as user u is numerically estimated by a score function $score(a, u)$.

4.1.2 Research Paper Index

We want to search through and filter the data in the S2ORC dataset in a way that allows for efficient profile matching. To accomplish this, we create a searchable index of the paper records in the dataset using Elasticsearch[1] and its default BM25 ranking model. We refer to this searchable index as the *research paper index*.

4.1.3 Profile Matching

We look at two methods for profile matching. The methods define the score function mentioned in Section 4.1.1 slightly differently. Using either method, generating a ranking of author nodes for a user involves querying the research paper index (one or more times, depending on the method) using information about the user, such as their name, finding the set of author nodes present in the query results, and picking the top-k author nodes based on their scores.

Frequency-Based

This simple method is designed to favor authors with a high number of publications. We define the score of author a for user u:

$$score(a, u) = \sum_{p \in \mathcal{P}_u} \mathbb{1}(a \in \mathcal{A}_p), \tag{4.1}$$

where $\mathcal{P}_u$ is the top-k set of papers returned by the research paper index when querying for user u's name, $\mathcal{A}_p$ is paper p's set of authors, and $\mathbb{1}(a \in \mathcal{A}_p)$ evaluates to 1 if author $a \in \mathcal{A}_p$ and 0 if not.

Score-Based

This method uses not only the name of the user but also the user's topics of interest, and — instead of simply counting the number of occurrences of authors in the query results returned by the research paper index — its output is based on the relevance

[1] `https://www.elastic.co/`

scores returned by the research paper index. We define the score of author a for user u:

$$score(a, u) = \sum_{t \in \mathcal{T}_u} \sum_{p \in \mathcal{P}_{u,t}} score(p) \mathbb{1}(a \in \mathcal{A}_p), \tag{4.2}$$

where $\mathcal{T}_u$ is user u's topics of interest, $\mathcal{P}_{u,t}$ is the top-k set of papers returned by the research paper index when querying for topic t together with user u's name, and $score(p)$ is the score of paper p as returned by Elasticsearch.

Post-Filtering

It is possible that none of the author nodes that are appropriate for a user are contained in the results returned by the research paper index for the query (or queries) made by either profile matching method, and it is also possible that no appropriate author nodes exist. To exclude author nodes that are obviously incorrect from the user's ranking and increase the probability that the author nodes that are actually present in the ranking are relevant, we filter the ranking based on the edit (Levenshtein) distances between the names of the author nodes and the user's name.

4.2 Implementation

Our methods are implemented as part of the arXivDigest codebase. Since arXivDigest itself is implemented in Python, Python was a natural choice of language. We use the Python Elasticsearch Client[2], which is a low-level wrapper around the Elasticsearch API, to interface with Elasticsearch in our code. The code is available in the arXivDigest GitHub repository[3], and all file paths in this section are relative to the root of this repository.

4.2.1 Research Paper Index

Indexing the S2ORC dataset in Elastcsearch is handled by the `scripts/index_open_research_corpus.py` script. This script uses the bulk helper functions of the Python Elasticsearch Client to read the dataset from disk and index it. It accepts three options:

`--index` is used to specify the Elasticsearch index.
`--host` is used to specify the Elasticsearch host.

[2]Documentation available at `https://elasticsearch-py.readthedocs.io`.
[3]`https://github.com/iai-group/arXivDigest`

```
{ "query": { "match": { "authors.name": "John Doe" } } }
```

Listing 4.1: Example of an Elasticsearch query used in the frequency-based method for a user with name John Doe.

`--path` is used to specify the location of the S2ORC dataset. The path should be a directory containing gzipped batch files with one JSON paper record per line.

4.2.2 Profile Matching

The two profile matching methods are implemented by the `scripts/gen_semantic_scholar_suggestions.py` script. This script generates rankings of Semantic Scholar author IDs for all users who have not registered links to Semantic Scholar profiles and have not previously accepted or discarded any profile suggestions (through a suggestion feature that will be described in Section 4.2.3). Rankings are stored in the `semantic_scholar_suggestions` database table. The script accepts several options:

`--index` is used to specify the Elasticsearch index.
`--host` is used to specify the Elasticsearch host.
`--method` is used to specify which profile matching method should be used and accepts either `score` or `frequency`.
`--max-suggestions` is used to limit the size of the user rankings. This option defaults to 5.
`-k` is used to specify the number of query results from the research paper index (top-k) to take into consideration for each query that is made. This option defaults to 50.
`--max-edit-distance` is used to specify the max edit distance (Levenshtein distance) between the user's name and the names of the profiles in their ranking. This option defaults to 1.
`--output` is used to direct the output of the script (the generated suggestions) to a file instead of writing them directly to the database. If this option is provided, rankings are generated for all users (not just the ones with missing profile links) and are output in a TREC suggestion format that will be described in Section 4.3.1.

Frequency-Based

Listing 4.1 shows what the Elasticsearch queries used to query the research paper index for potential author nodes look like using the frequency-based method.

```
{
    "query": {
        "bool": {
            "must": [
                { "match": { "authors.name": "John Doe" } },
                {
                    "multi_match": {
                        "query": "database system",
                        "fields": ["title", "paperAbstract", "fieldsOfStudy"]
                    }
                }
            ]
        }
    }
}
```

Listing 4.2: Example of the Elasticsearch queries used in the score-based method for a user with name John Doe and an interest t = "database system".

Score-Based

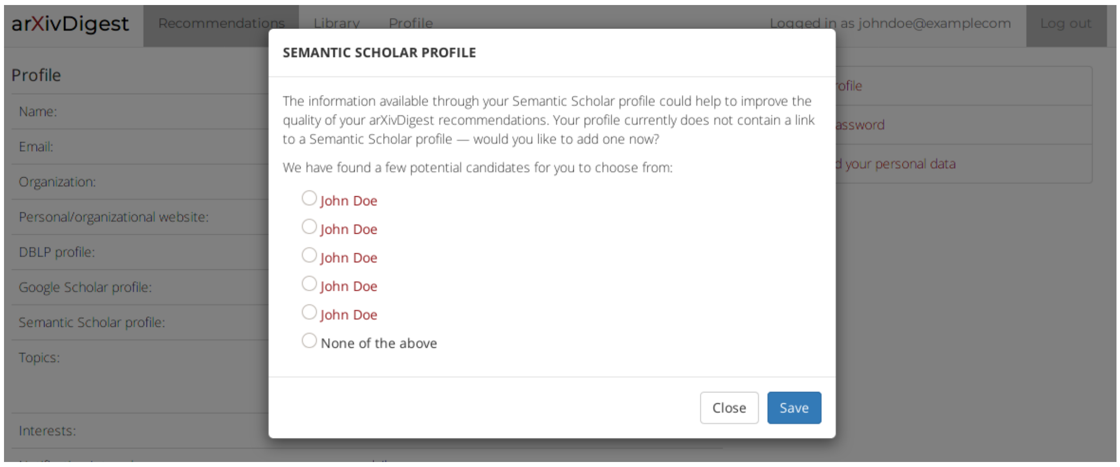
Listing 4.2 shows what the Elasticsearch queries used to query the research paper index for potential author nodes look like using the score-based method.

4.2.3 Profile Suggestion Feature

The ranking produced for a user is displayed to them in a popup on the arXivDigest website as a list of suggested Semantic Scholar profiles, as shown in Fig. 4.1. The popup, which is displayed upon login, contains a form with one radio button for each profile present in the ranking, and each radio button contains the name of its respective profile as a link to the profile. The user can choose to accept one of the suggestions as their profile or discard them all by selecting the "None of the above" option. Their choice is logged to the `semantic_scholar_suggestion_log` database table.

4.3 Evaluation

This section describes the methodology we adopt to evaluate our methods and presents the results of our evaluations.



SEMANTIC SCHOLAR PROFILE

The information available through your Semantic Scholar profile could help to improve the quality of your arXivDigest recommendations. Your profile currently does not contain a link to a Semantic Scholar profile — would you like to add one now?

We have found a few potential candidates for you to choose from:

- ◯ John Doe
- ◯ John Doe
- ◯ John Doe
- ◯ John Doe
- ◯ John Doe
- ◯ None of the above

Figure 4.1: The user is able to choose between the available Semantic Scholar profile suggestions through a popup which is displayed upon login.

4.3.1 Evaluation Methodology

We evaluate our methods in two steps. First, we perform offline evaluations of rankings generated using several different configurations of the profile matching script. The configuration that comes out on top in the offline evaluations is then used to generate rankings that are presented to users on the arXivDigest website as profile suggestions. These suggestions are subject to user feedback, which we look at in our online evaluations.

Offline Evaluation

The most precise way to evaluate profile suggestions is to look at measures such as recall and mean reciprocal rank (MRR). Calculating these measures requires access to some sort of ground truth. ArXivDigest has a small user base, so creating a ground truth table containing the actual Semantic Scholar author IDs of all arXivDigest users is, therefore, a feasible task. We create the ground truth the following way. Starting off, the contents of the users table in the arXivDigest database are dumped to a file. A method parses the database dump and extracts the Semantic Scholar author IDs present in the user-provided Semantic Scholar profile links. For the users who have not provided profile links themselves, the method prompts for a link, which we look up manually with Semantic Scholar's search engine. Using this method, we are able to create a ground truth containing the Semantic Scholar author IDs of 84 users (three of which have two IDs).

We use the trec_eval tool[4] to calculate offline metrics for us. This tool expects its input files — in our case, a ground truth file and a file with rankings — to be of certain formats. The ground truth must be formatted as a qrels[5] file containing four space-separated columns: topic number (arXivDigest user ID), iteration (always zero and not used), document number (Semantic Scholar author ID), and relevance (always 1). The rankings file must follow a TREC suggestion format where each line represents an author node (a Semantic Scholar author ID) and contains six space-separated columns: query ID (arXivDigest user ID), iteration (always Q0), document number (Semantic Scholar author ID), rank, relevance/score, and run ID. The trec_eval tool is able to calculate many different metrics. The ones we are the most interested in are: *num_q*, which is the number of users (queries) with rankings; *num_rel_ret*, which is the number of correct Semantic Scholar author IDs (relevant docment numbers) that are present in the rankings; *recall_5*, which is recall@5; and *recip_rank*, which is the MRR.

We generate and evaluate four sets of rankings against the ground truth: two sets based on the score-based method, using max edit distances of 1 and 2, and two sets based on the frequency-based method, using max edit distances of 1 and 2. The commands used to generate the rankings are available in Appendix A.

Online Evaluation

In our online evaluations, we analyze the feedback submitted by users through the profile suggestion feature described in Section 4.2.3. In particular, we look at the reciprocal ranks and mean reciprocal rank of accepted and rejected suggestions.

4.3.2 Results

We now present the results of our evaluations.

Offline Evaluation

The offline evaluation results can be seen in Table 4.1. We observe that the score-based method achieves an MRR that is roughly 0.1 higher than the MRR of the frequency-based method for both max edit distances, and that the smaller max edit distance gives the best results across the board. Out of the 84 users represented in the ground truth, the score-based and frequency-based methods generate rankings for 79 and 78 users,

[4]`https://github.com/usnistgov/trec_eval`
[5]Format described here: `https://trec.nist.gov/data/qrels_eng`.

Table 4.1: Evaluation results for rankings generated using both the score-based and frequency-based methods, with max edit distance of both 1 and 2. num_q is the number of users with rankings and num_rel_ret is the number correct Semantic Scholar author IDs that are present in the rankings.

(a) Max edit distance of 1.

Profile matching method	num_q	num_rel_ret	Recall@5	MRR
Score-based	79	75	0.93	0.88
Frequency-based	78	74	0.92	0.79

(b) Max edit distance of 2.

Profile matching method	num_q	num_rel_ret	Recall@5	MRR
Score-based	79	74	0.92	0.87
Frequency-based	78	74	0.92	0.77

respectively. The frequency-based method fails to produce a ranking for one user in addition to the same five users that the score-based method fails to produce rankings for. After taking a closer look at these users, nothing about them stands out when compared to the others, and it is difficult to say why neither method was able to produce rankings for them.

Online Evaluation

The score-based method paired with a max edit distance of 1 achieved the best results in the offline evaluations. The rankings generated with this method were displayed as profile suggestions to users on the arXivDigest website. After six weeks, nine users had interacted with the suggestions: one user rejected all suggestions (equivalent to accepting the suggestion with rank 0) and eight users accepted one of their suggestions. Of the users who accepted suggestions, five accepted the suggestions with rank 1 (the top suggestions) and three accepted the suggestions with rank 2. We get an MRR of 0.72.

Table 4.2 contains the rankings that were presented as suggestions to the users who did not accept suggestions with rank 1. We can see that the rankings of both David Corney and John Kane contain suggestions (at the bottom ranks) that are obviously wrong due to incorrect first name initials. After closer inspection, the top suggestion for David Corney appears to be a duplicate of the accepted suggestion (but a different profile). Both the first name initial and last name of Martin Uray's only (rejected) suggestion

Table 4.2: The rankings that were presented as suggestions to the users who did not accept suggestions with rank 1. The name of the suggestion with rank 1 for user Daniel Gayo-Avello is blank because the suggested profile no longer exists.

User	Suggestion			
	Rank	Score	Name	Accepted
David Corney	1	960	D. Corney	No
	2	51	D. Corney	Yes
	3	29	M. Corney	No
	4	23	L. Corney	No
Martin Uray	1	160	M. Uray	No
Daniel Gayo-Avello	1	1700	–	No
	2	170	Daniel Gayo-Avello	Yes
John Kane	1	190	J. Kane	No
	2	170	J. Kane	Yes
	3	39	J. Kane	No
	4	26	J. Keane	No
	5	24	D. Kane	No

are correct. For Daniel Gayo-Avello, the top suggestion is a profile that no longer exists. This error could be a result of our method using a slightly outdated version of the S2ORC dataset. For John Kane, the top three suggestions all have correct first name initials and last names, and it is not surprising that our method failed to properly rank these.

4.4 Summary

The score-based method combined with a max edit distance of 1 performed the best in the offline evaluations. In the online evaluations, this configuration produced rankings containing the correct Semantic Scholar author IDs for eight out of the nine users who interacted with the profile suggestion feature on arXivDigest during a period of six weeks. A few of these users were suggested profiles with incorrect first name initials. Suggestions like these could have been caught with an additional post-filtering step.

The online evaluations resulted in an MRR of 0.72, which is considerably lower than the MRR of 0.88 achieved with the same profile matching configuration in the offline evaluations. Since feedback was submitted by only nine users in the online evaluations (compared to the 84 users used in the offline evaluations), it is difficult to compare these two results. Still, as mentioned in Section 2.1.2, a disconnect between the results of offline and online evaluations is, typically, to be expected.

The goal of this chapter was to increase the number of users with links to the correct scientific knowledge graph entries. Albeit only by eight users, our methods have managed to increase this number.

Chapter 5

Research Paper Recommendation

The methods of Chapter 4 have, if only slightly, increased the number of arXivDigest users with links to appropriate scientific knowledge graph entries. It is time to approach the second problem defined in Section 1.1 and explore applications of the data available through scientific knowledge graphs in the realm of research paper recommendation. We describe several methods for explainable research paper recommendation and their implementations.

5.1 Methods

This section starts with a formal description of the *explainable research paper recommendation* task. After this, we introduce six methods for explainable research paper recommendation. Each method defines a score function $score(p, u)$, which attempts to quantify the relevance of research paper p for user u, and a sentence template (or sentence templates), which is filled in to produce a personalized explanation of the score that is assigned to the paper. The first three methods score candidate papers based on their venues of publication, the venues the papers' authors have published papers at, and the venues the user has published papers at. The last three methods score candidate papers based on the citation graphs of their authors, the user, and the user's collaborators. Table 5.1 summarizes the information that is used by the different recommendation methods.

Using the terminology of Section 3.1, all of our methods are graph-based, and two have hybrid characteristics. Frequent Venues, Venue Co-Publishing, Previously Cited, and Previously Cited by Collaborators rely only on the connections that exist in a scientific knowledge graph, and are purely graph-based. Weighted Influence incorporates the (influential) citation counts of papers and has additional characteristics typical for global relevance methods. Previously Cited and Topic Search has additional characteristics that are typical for content-based methods. Section 3.2 mentioned that a distinction could

Table 5.1: Information used by the different recommendation methods to calculate the score of candidate paper p for user u.

Recommendation method	Information used by $score(p, u)$
Frequent Venues	Venues of publication for p and the papers published by u.
Venue Co-Publishing	Venues of publication for all papers published by the authors of p and the papers published by u.
Weighted Influence	Venues of publication for all papers published by the authors of p and the papers published by u.
Previously Cited	Authors of p and authors cited by u in their published papers.
Previously Cited by Collaborators	Authors of p and authors cited by the previous collaborators (co-authors) of u in their published papers.
Previously Cited and Topic Search	Authors of p and authors cited by u in their published papers.

be made between model-intrinsic and model-agnostic explainable recommendation [43]. With transparent decision making and no use of additional explanation models to produce explanations post hoc, all of our methods are model-intrinsic.

5.1.1 Problem Statement

We define explainable research paper recommendation to be the task of producing a ranking of the papers that are candidates for recommendation at arXivDigest based on their relevance to a user. The relevance of each candidate paper p is numerically estimated for user u by a score function $score(p, u)$. In addition to a score, each ranked paper is accompanied by an explanation for its score.

5.1.2 Frequent Venues

It is not uncommon for researchers to publish numerous papers at the same venue over time. This method is based on the assumption that a paper published at a venue that a user frequently publishes at is more relevant to the user than other papers.

We maintain a set of venues that is continually updated as new venues are discovered during the recommendation process. We refer to the size of this set as N. Users are represented as N-dimensional vectors, where each value corresponds to a certain venue

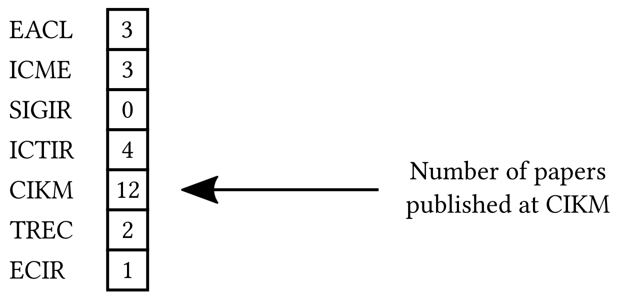


Figure 5.1: Vector representation of an author based on the number of papers published by the author at different venues.

and represents the number of papers the user has published there. Figure 5.1 shows an example of such a user vector. Papers are also represented as N-dimensional vectors, where the value corresponding to a paper's venue is 1 and all other values are zero. We define the score (or relevance) of research paper $\boldsymbol{p}$ for user $\boldsymbol{u}$:

$$score(\boldsymbol{p}, \boldsymbol{u}) = \boldsymbol{p} \cdot \boldsymbol{u}. \tag{5.1}$$

This method is very simple, and it is easy to explain to the user why a certain paper has been recommended. Users are presented with explanations of the form "This article is published at [venue], where you have published [x] papers."

5.1.3 Venue Co-Publishing

In the Frequent Venues method, papers are represented by the venues they are published at. This method represents papers by their authors and is based on the assumption that a paper is relevant to a user if the authors of the paper have published at the same venues as the user. Both users and authors are represented using the same type of N-dimensional vector used for users in the Frequent Venues method. We define the score of research paper p for user $\boldsymbol{u}$:

$$score(p, \boldsymbol{u}) = \max_{a \in \mathcal{A}_p} sim(\boldsymbol{a}, \boldsymbol{u}), \tag{5.2}$$

where $\mathcal{A}_p$ is paper p's set of authors and $sim(\cdot, \cdot)$ is the cosine similarity.

If the user has published at the same venue as one of the authors of a candidate paper more than once, explanations are of the form "You have published [x] times at [venue], …, and [z] times at [venue]. [author] has also published at these venues." The [venue]s are placeholders for the common venues at which the user has published the most papers. If a user has published no more than once at any of the venues that the paper authors have published at, explanations are of the form "You and [author] have both published at [venue], …, and [venue]." In this case, the [venue]s are placeholders for any of the common venues.

5.1.4 Weighted Influence

This method is similar to the Venue Co-Publishing method. The only difference is that the similarities between the user and the authors of a paper are weighted by the authors' influential citation counts — or vice versa: the influential citation counts are weighted by the similarities. We define the score of research paper p for user u:

$$score(p, \boldsymbol{u}) = \max_{\boldsymbol{a} \in A_p} \sum_{v \in \mathcal{V}_u} influence(\boldsymbol{a}, v) \times sim(\boldsymbol{a}, \boldsymbol{u}), \tag{5.3}$$

where $\mathcal{V}_u$ is the set of venues that user u has published papers at and $influence(\boldsymbol{a}, v)$ is the sum of the influential citation counts of the papers that author $\boldsymbol{a}$ has published at venue v. The influential citation count metric was explained in Section 2.3.1. In short, it is a more accurate metric for a paper's influence on citing papers than a pure citation count.

Explanations for recommendations produced with this method are of the form "[author] has had influential publications at [venue], …, and [venue], which are venues you also publish at." [author] is a placeholder for the name of the author producing the greatest score and the [venue]s are placeholders for the venues for which the author has the greatest influential citation counts.

5.1.5 Previously Cited

The previous methods score papers based on which venues they are published at, which venues their authors have published papers at, and which venues the user has published papers at. This method moves away from venues and is based on the assumption that users are interested in new publications from authors they have previously cited. We define the score of research paper p for user u:

$$score(p, u) = \max_{a \in \mathcal{A}_p} numcites(u, a), \tag{5.4}$$

where $\mathcal{A}_p$ is paper p's set of authors and $numcites(u, a)$ is the number of times user u has cited author a.

Explanations for recommendations produced with this method are of the form "This article is authored by [author], who you have cited [x] times." [author] is a placeholder for the name of the author that the user has cited the most times.

5.1.6 Previously Cited by Collaborators

This method is similar to the Previously Cited method, but instead of looking at whether the user has cited the authors of a paper, it looks at whether the user's previous collaborators have done so. We define the score of research paper p for user u:

$$score(p, u) = \max_{a \in \mathcal{A}_p, c \in C_u} numcites(c, a), \tag{5.5}$$

where $\mathcal{A}_p$ is paper p's set of authors, C_u is user u's set of collaborators, and $numcites(c, a)$ is the number of times collaborator c has cited author a.

Explanations for recommendations produced with this method are of the form "This article is authored by [author], who has been cited by your previous collaborator [collaborator] [x] times." [author] is a placeholder for the name of the author that has been cited the most times by any one of the user's previous collaborators and [collaborator] is a placeholder for the name of this collaborator.

5.1.7 Previously Cited and Topic Search

This method combines Previously Cited with the approach of the baseline arXivDigest recommender system described in Section 2.2.2, which queries an Elasticsearch index containing the candidate papers for the user's topics of interest. We define the score of research paper p for user u:

$$score(p, u) = \sum_{t \in \mathcal{T}_u} score(p, t) \times \max_{a \in \mathcal{A}_p} numcites(u, a), \tag{5.6}$$

where $\mathcal{T}_u$ is user u's topics of interest, $score(p, t)$ is the score of paper p as returned by Elasticsearch when querying for topic t, $\mathcal{A}_p$ is paper p's set of authors, and $numcites(u, a)$ is the number of times user u has cited author a. If p is not returned by the index for a given t, $score(p, t)$ is considered 0.

Explanations are of the following format, which combines the explanation formats of the base arXivDigest recommender system and Previously Cited: "This article seems to be about [topic], ..., and [topic], and is authored by [author], who you have cited [x] times." The [topic]s are placeholders for the topics for which Elasticsearch returns the greatest scores and [author] is a placeholder for the author that the user has cited the most times.

5.2 Implementation

This section describes how our methods have been implemented and how the implementations are configured and deployed. As in Chapter 4, Python is the programming language of choice. The code is available at `https://github.com/olafapl/arxivdigest_recommenders`. All relevant code is contained in the `arxivdigest_recommenders` package in this repository, and all file paths in this section are relative to the root of this package.

5.2.1 The Semantic Scholar API

The Semantic Scholar API was introduced in Section 2.3.1. Our systems use this API as an interface to the scientific literature graph of Semantic Scholar to retrieve additional information about users, papers, and authors that is not available through the arXivDigest API. By default, the Semantic Scholar API has a rate limit of 100 req/5 min. This limit is a bit low for us, as we want to access information about thousands of records in relatively short periods of time. We reached out to Semantic Scholar, who was generous and provided us with a private API key with a rate limit of 100 req/s.

Network Traffic Analysis

All the described recommendation methods require information about a substantial number of papers and authors to be available. Getting hold of this information involves querying the Semantic Scholar API, and the total number of queries quickly adds up when we are dealing with thousands of papers and authors. In this section, we perform an analysis of the complexity of the situation.

We denote the number of papers that are candidates for recommendation n. Retrieving information about all the candidate papers involves making n requests to the Semantic Scholar API. The Venue Co-Publishing and Weighted Influence methods also need information about the authors of the candidate papers. If papers on average have a authors, retrieving this information involves making an additional na requests. These methods need information about the authors' published papers beyond their titles and years of publication, and must therefore also retrieve information about all of the papers published by each author. If authors have p published papers on average, this adds another nap requests. All the recommendation methods need information about the users of arXivDigest (with known Semantic Scholar author IDs) and their published papers. If there are u users, retrieving this information involves making $u + up$ requests. The Previously Cited by Collaborators method also needs information about the users'

Table 5.2: Equations for the total number of Semantic Scholar API requests made by each of the recommendation methods. n is the number of papers that are candidates for recommendation, a is the average number of authors per paper, p is the average number of papers published by authors, u is the number of users, and c is the average number of unique collaborators for users.

Recommendation method	Requests
Frequent Venues	$n + u(1 + p)$
Venue Co-Publishing	$(na + u)(1 + p)$
Weighted Influence	$(na + u)(1 + p)$
Previously Cited	$n + u(1 + p)$
Previously Cited by Collaborators	$n + u(1 + p)(1 + c)$
Previously Cited and Topic Search	$n + u(1 + p)$

collaborators and their published papers. If the users have c unique collaborators on average, this leads to an additional $u(c + cp)$ requests. Table 5.2 contains the final equations for the total number of requests needed for each of the recommendation methods.

The arXivDigest API typically returns between 3000 and 4000 paper recommendation candidates. To estimate a and p, we implement a method that (1) selects a random sample of 1000 papers from the set of papers that are candidates for recommendation, (2) fetches the metadata of the papers, their authors, and the authors' published papers from the Semantic Scholar API, and (3) calculates the average number of authors per paper and the average number of papers published per author at any time and in the last five years. Using the averages calculated by this method, we get $a = 5.34$, $p = 120$ when looking at papers published at any time, and $p = 41.5$ when only looking at papers published in the last five years. To estimate c, we implement another method that fetches the paper metadata of the published papers for all users with known Semantic Scholar author IDs and calculates the average number of unique collaborators. Using this method, we find that 23 users have registered Semantic Scholar profiles, and that the average number of collaborators for these users is 77. Table 5.3 shows the results of plugging $n = 3500$, $a = 5.34$, both $p = 120$ and $p = 41.5$, $u = 23$, and $c = 77$ into the equations in Table 5.2. It also contains estimates for how long it would take each method to complete its estimated number of requests based on a constant request rate of 100 req/s. We can see that the Venue Co-Publishing and Weighted Influence methods produce considerably more traffic than the other methods. The Previously Cited by Collaborators method has the greatest absolute increase in requests per extra user of all the methods, but because its baseline number of requests — i.e., its number of requests for $u = 0$ — is low and because we are only dealing with 23 users, this will not pose any

Table 5.3: Estimates of the total number of Semantic Scholar API requests needed by each of the recommendation methods and the time needed to complete the requests. The request estimates are based on the equations in Table 5.2, using $n = 3500$, $a = 5.34$, $u = 23$, and $c = 77$. The time estimates are based on a request rate of 100 req/s.

Recommendation method	Requests		Time	
	$p = 120$	$p = 41.5$	$p = 120$	$p = 41.5$
Frequent Venues	6.3×10^3	4.5×10^3	1.0 min	45 s
Venue Co-Publishing	2.3×10^6	8.0×10^5	6.3 h	2.2 h
Weighted Influence	2.3×10^6	8.0×10^5	6.3 h	2.2 h
Previously Cited	6.3×10^3	4.5×10^3	1.0 min	45 s
Previously Cited by Collaborators	2.2×10^5	8.0×10^4	37 min	13 min
Previously Cited and Topic Search	6.3×10^3	4.5×10^3	1.0 min	45 s

problems for us.

Network Request Caching

The recommender systems implementing the different recommendation methods are intended to run on the same machine at the same time, and many of the requests made to the Semantic Scholar API by the different systems are going to be for the same resources. As the set of papers that are candidates for recommendation one day is very similar to the set of papers that are candidates for recommendation the next one, there is also going to be a significant overlap between the sets of resources requested from the API on consecutive days. The set of arXivDigest users is also likely to stay mostly the same from day to day, with the odd additions. We choose to cache author details for seven days and paper metadata for 30 days in order to lower the number of requests actually sent to Semantic Scholar.

Python Wrapper

There were no existing Python wrappers for the Semantic Scholar API with support for response caching, asynchronous requests, and rate limiting, so we implemented one ourselves. The `SemanticScholar` wrapper class is located in the `semantic_scholar.py` module. Internally, it uses the `ClientSession` of aiohttp[1] to make requests. Responses are cached in either MongoDB using the *motor*[2] MongoDB Python driver or Redis using

[1] `https://docs.aiohttp.org/en/stable/`

[2] `https://motor.readthedocs.io/en/stable/`

```
async with SemanticScholar() as s2:
    author = await s2.author("1680484")
    author_papers = await s2.author_papers("1680484")
    arxiv_paper = await s2.paper(arxiv_id="2009.11576")
    s2_paper = await s2.paper(s2_id="006d90b7e05d261cb5c3dd27f27e02806d664ffa")
```

Listing 5.1: Using the Semantic Scholar API wrapper to retrieve information about an author, all the author's published papers, and a paper using both its arXiv and Semantic Scholar IDs.

the *aioredis*[3] Redis client library. Rate limiting is handled by the `AsyncRateLimiter` in the `util.py` module. The API wrapper is implemented as an asynchronous context manager, and is used internally by the recommender systems that will be described in Section 5.2.2 as shown in Listing 5.1.

5.2.2 Recommender Systems

We implement one recommender system for each of the six recommendation methods described in Section 5.1. The locations and names of the recommender system classes implementing the different methods are listed in Table 5.4. At the base of each system is the `ArxivdigestRecommender` superclass located in the `recommender.py` module. This class does most of the heavy lifting and leaves only the scoring of papers to be defined by its subclasses. Subclasses define the scoring of papers by implementing the `score_paper` method, which takes as arguments the data of a user, the user's Semantic Scholar author ID, and the arXiv ID of a paper that should be scored. The method should return a dictionary containing the arXiv ID of the paper, the paper's score, and an explanation for the paper's score. This method is called by `ArxivdigestRecommender` for each candidate paper when the `recommend` method of a recommender system is called. The candidate papers are then sorted by score, papers with a score of 0 are filtered out, and the top-k papers get submitted to arXivDigest as recommendations for the user.

The arXivDigest API for experimental recommender systems was introduced in Section 2.2.1. Our recommender systems use this API to retrieve user information and the arXiv IDs of candidate papers, and to submit recommendations. To summarize, the process used by our systems looks like this:

1. Retrieve the arXiv IDs of the papers that are candidates for recommendation from the `/articles` endpoint.

[3] `https://aioredis.readthedocs.io/en/latest/`

Table 5.4: Modules and names of the recommender system classes implementing the different recommendation methods.

Recommendation method	Module	Class
Frequent Venues	`frequent_venues.py`	`FrequentVenuesRecommender`
Venue Co-Publishing	`venue_copub.py`	`VenueCoPubRecommender`
Weighted Influence	`weighted_inf.py`	`WeightedInfRecommender`
Previously Cited	`prev_cited.py`	`PrevCitedRecommender`
Previously Cited by Collaborators	`prev_cited_collab.py`	`PrevCitedCollabRecommender`
Previously Cited and Topic Search	`prev_cited_topic.py`	`PrevCitedTopicSearchRecommender`

2. Retrieve the IDs for a batch of users from the `/users?from=[Start ID]` endpoint, additional information about the users in the batch from the `/user_info?ids=[User IDs]` endpoint, and the arXiv IDs of the papers that have already been recommended for each user in the batch from the `/user_feedback/articles?user_id=[User IDs]` endpoint.
3. Assign a score to each candidate paper together with an explanation for the score, filter out the papers that have already been recommended, and submit the top papers for each user to the `/recommendations/articles` endpoint.
4. Repeat the two previous steps until recommendations have been submitted for all users.

All interaction with the API is done through the `ArxivdigestConnector` class of the `arxivdigest.core.connector` module in the arXivDigest GitHub repository[4].

5.2.3 Deployment

We have implemented six recommender systems and need to distinguish between them in the evaluation and user feedback dashboard on the arXivDigest website. To accomplish this, we use separate API keys for all the systems. The API keys, together with everything else that is configurable, are configured through a JSON file located in one of three locations (in order of priority):

1. `%cwd%/config.json`; or
2. `~/arxivdigest-recommenders/config.json`; or
3. `/etc/arxivdigest-recommenders/config.json`.

[4] `https://github.com/iai-group/arXivDigest`

The default configuration can be seen in Listing 5.2. All the configuration options are explained in Table 5.5. If any of these options are redefined in a file in one of the three locations, their defaults are overridden.

Each recommender system can be run by executing the module the recommender system class is located in. As an example, the system implementing the Frequent Venues method can be run by executing the `arxivdigest_recommenders.frequent_venues` module. The configured Semantic Scholar API rate limit works only on a per-process basis, meaning that if two recommenders are run at the same time using this method, the effective rate limit will be double that of what is configured. To avoid this problem, the recommenders can be run in the same process, as in Listing 5.3.

5.3 Evaluation

This section presents the methodology we adopt to evaluate our methods and presents the results of the evaluations.

5.3.1 Evaluation Methodology

We evaluate our methods in two steps. First, we compare the recommendations produced by the six implemented recommender systems and the baseline recommender system described in Section 2.2.2. This shows us if the systems tend to recommend the same papers and checks if any of them stand out. After this, we move on to a user study, where we ask for feedback from users on recommendations and explanations produced by each system.

ArXivDigest is a relatively small platform, and only a small amount of traffic has been generated by the users in the period that our systems have been operational. This means that there is not a whole lot of data (i.e., user interactions and feedback) for us to use and analyze in online evaluations. Our evaluation methodology does, therefore, not include online evaluations.

Recommendation Overlap and Uniqueness

We look at the *overlap* between the sets of papers that have been recommended to the same users by the different recommender systems, and the *uniqueness* of the systems' recommendations. If A and B are the sets of papers recommended to the same user by two systems a and b, then we define the overlap between A and B to be $|A \cap B|/|A \cup B|$.

```
{
    "arxivdigest_base_url": "https://api.arxivdigest.org/",
    "mongodb": {
        "host": "127.0.0.1",
        "port": 27017
    },
    "redis": {
        "host": "127.0.0.1",
        "port": 6379
    },
    "elasticsearch": {
        "host": "127.0.0.1",
        "port": 9200
    },
    "semantic_scholar": {
        "api_key": null,
        "max_requests": 100,
        "window_size": 300,
        "cache_responses": true,
        "cache_backend": "redis",
        "mongodb_db": "s2cache",
        "mongodb_collection": "s2cache",
        "paper_cache_expiration": 30,
        "author_cache_expiration": 7
    },
    "max_paper_age": 5,
    "max_explanation_venues": 3,
    "venue_blacklist": ["arxiv"],
    "frequent_venues_recommender": { "arxivdigest_api_key": null },
    "venue_copub_recommender":  { "arxivdigest_api_key": null },
    "weighted_inf_recommender": {
        "arxivdigest_api_key": null,
        "min_influence": 20
    },
    "prev_cited_recommender": { "arxivdigest_api_key": null },
    "prev_cited_collab_recommender": { "arxivdigest_api_key": null },
    "prev_cited_topic_recommender": {
        "arxivdigest_api_key": null,
        "index": "arxivdigest_papers",
        "max_explanation_topics": 3
    },
    "log_level": "INFO"
}
```

Listing 5.2: Default configuration of the recommender systems.

Table 5.5: Explanations of the properties present in the configuration file in Listing 5.2.

Property	Explanation
`arxivdigest_base_url`	Base URL of the arXivDigest API.
`mongodb`	MongoDB host and port.
`redis`	Redis host and port.
`elasticsearch`	Elasticsearch host and port.
`semantic_scholar`	Options for the Semantic Scholar API wrapper. `max_concurrent_requests` defaults to 100. The default `max_requests` of 100 and `window_size` of 300 (seconds) equate to a max request rate of 100 req/5 min and can be used without an API key. `cache_responses` is used to toggle caching, and `cache_backend` is used to toggle between the MongoDB ("mongodb") and Redis ("redis") cache backends.
`max_paper_age`	Papers older than this (in years) are filtered out when looking at an author's published papers. In light of the network traffic analysis performed in Section 5.2.1, and because newer papers are likely to best reflect an author's current publishing patterns, the default value of this option is 5.
`max_explanation_venues`	Max number of venues included in explanations by the Venue Co-Publishing and Weighted Influence recommenders.
`venue_blacklist`	Case-insensitive list of venues ignored by the Venue Co-Publishing and Weighted Influence recommenders. Many papers are not published at a specific venue, such as TREC, but are instead available directly on pre-publishing services, such as arXiv. If two authors who have never published at the same venues both have many papers with arXiv listed as their venues of publication, the similarity between these authors is artificially high if arXiv is not filtered out as a venue.
`frequent_venues_recommender`	Frequent Venues recommender options.
`venue_copub_recommender`	Venue Co-Publishing recommender options.
`weighted_inf_recommender`	Weighted Influence recommender options. If the sum of influential citation counts for all papers published by author $\boldsymbol{a}$ at venue v is smaller than `min_influence`, $influence(\boldsymbol{a}, v)$ is considered 0.
`prev_cited_recommender`	Previously Cited recommender options.
`prev_cited_collab_recommender`	Previously Cited by Collaborators recommender options.
`prev_cited_topic_recommender`	Previously Cited and Topic Search recommender options. `index` is the Elasticsearch index for candidate paper indexing and topic search. `max_explanation_topics` specifies the max number of topics included in explanations.
`log_level`	Minimum logging level ("FATAL", "ERROR", "WARNING", "INFO", or "DEBUG").

```python
async def main():
    fv = FrequentVenuesRecommender()
    vcp = VenueCoPubRecommender()
    await asyncio.gather(*[fv.recommend(), vcp.recommend()])


asyncio.run(main())
```

Listing 5.3: Running two recommender systems at the same time.

If A is the set of papers recommended to a user by system a and C is the set of papers recommended to the same user by all other systems, then we define the uniqueness of A to be $|A - C|/|A + C|$.

The recommendations we look at from each system are produced from the same pool of 3957 candidate papers for the same 23 users. To enable us to properly compare our systems against the baseline, we choose to consider only recommendations made to users with registered Semantic Scholar profiles.

User Study

Inspired by the recommendation feedback form on the arXivDigest website, which was described in Section 2.2.3, we perform a user study where we ask users about the relevance of recommendations and how satisfactory, persuasive, transparent, and scrutable they find the explanations. Nine users are asked to participate. Some of these have not registered Semantic Scholar profiles or accepted any of the profile suggestions generated for them in Chapter 4. For these, we use Semantic Scholar author IDs from the ground truth table created in Section 4.3.1.

Each subject is asked questions related to one recommendation produced by each of the seven recommender systems. These are the top recommendations (by score) produced for the subject by each system inside a time window of one week. In addition to the one explanation that is associated with each recommendation, alternative explanations are produced for the recommended paper by the six other systems, so that seven explanations are associated with each recommendation in total. The subjects are asked two questions about the recommendations themselves, and three questions about each of their associated explanations. All questions are answered using a linear scale from 1 to 5 — commonly referred to as a *Likert scale*. The extremes of the scale are accompanied by labels that differ slightly from one question to another. The recommendation-related questions (and their labels) are:

1. "How relevant is this recommendation to you?" ("Not relevant at all", "Very relevant")
2. "How useful is this recommendation to you?" ("Not useful at all", "Very useful")

The explanation-related questions (and their labels) are:

1. "How convincing does this explanation sound to you?" ("Not convincing at all", "Very convincing")
2. "Does the explanation help you understand the reasoning behind this recommendation?" ("Not at all", "Very much")
3. "Does the explanation enable you to tell if the system has misunderstood your preferences?" ("Not at all", "Very much")

We present the results of the user study using Likert plots. We also look at the correlations between the responses to questions within each category (recommendation-related and explanation-related) and the internal consistencies of the subjects by analyzing their responses to each of the five question types.

5.3.2 Results

We now present the results of our evaluations.

Recommendation Overlap and Uniqueness

Table 5.6 contains the overlaps between the sets of recommendations made by the seven recommender systems, and Table 5.7 contains the uniqiueness of the recommendations made by each system. We observe that the overlap between the Previously Cited and Previously Cited and Topic Search systems is much greater than the overlap between any other systems. This is not surprising, as the second method is an extension of the first. Two other closely related systems are Venue Co-Publishing and Weighted Influence, but, for these two, the overlap is much smaller. The baseline system stands out from the rest, with 95 % of its recommendations being unique. Additionally, the overlap between its recommendations and any other system's is 1 % at the highest.

User Study

The Frequent Venues system did not produce any recommendations for any subjects, and was also not able to produce explanations for any of the recommendations made by the other systems. This recommender is, therefore, not included in any of the presented results.

Table 5.6: Mean overlaps (in %) between the sets of papers that have been recommended by the seven recommender systems to the same users. The recommendations 23 different users.

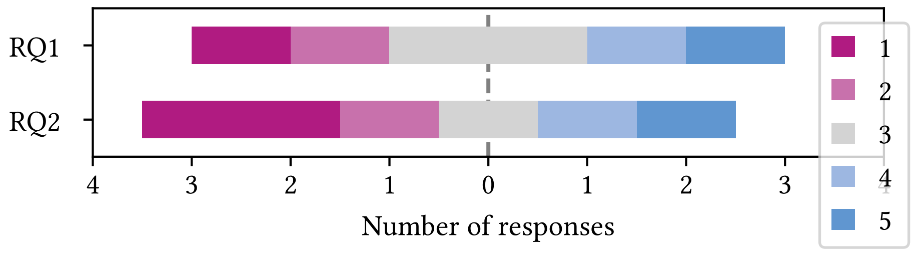
	B	FV	VCP	WI	PC	PCC	PCTS
B	100	0	1	0	0	1	1
FV	0	100	1	0	2	1	0
VCP	1	1	100	4	6	3	9
WI	0	0	4	100	10	8	7
PC	0	2	6	10	100	20	56
PCC	1	1	3	8	20	100	15
PCTS	1	0	9	7	56	15	100

Table 5.7: Mean uniqueness of the recommendations produced by the seven recommender systems.

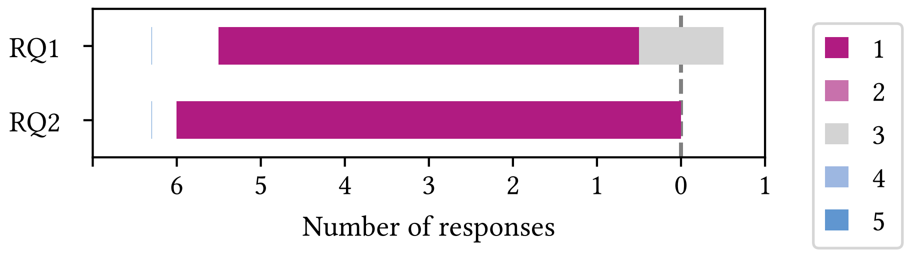
	B	FV	VCP	WI	PC	PCC	PCTS
Unique recommendations (%)	95	65	80	55	19	61	12

Six of the nine users asked to participate responded. Figure 5.2 shows the Likert plots of these users' responses to the recommendation-related questions, and Fig. 5.3 shows the Likert plots of their responses to the explanation-related questions. Each data point in these plots represents a response from a subject. The labels used to represent questions in the plots are based on the numbering of the questions in Section 5.3.1. As an example, RQ1 represents the first recommendation-related question and EQ1 represents the first explanation-related question. From the Likert plots of the responses to the explanation-related questions, it is clear that the number of data points varies for each system. This is not a result of subjects skipping questions, but stems from the fact that our systems are unable to produce explanations for papers that they assign scores of 0. Because of this, fewer than six (or seven, if we include the Frequent Venues system) different explanations were associated with most of the recommendations presented to the subjects.

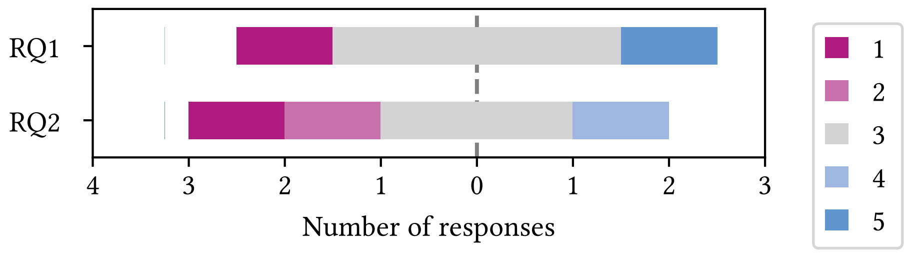
There are great contrasts between the systems in Fig. 5.2. The baseline system appears to receive the most positive feedback, with the Weighted Influence system following closely behind. Even though these systems receive the best feedback, they still appear to produce just as many bad as good recommendations. With almost all responses being 1 to both RQ1 and RQ2, the Venue Co-Publishing system receives the worst feedback of all the systems. The feedback for the Previously Cited and Previously Cited by Collaborators systems are not much better. The Previously Cited and Topic Search



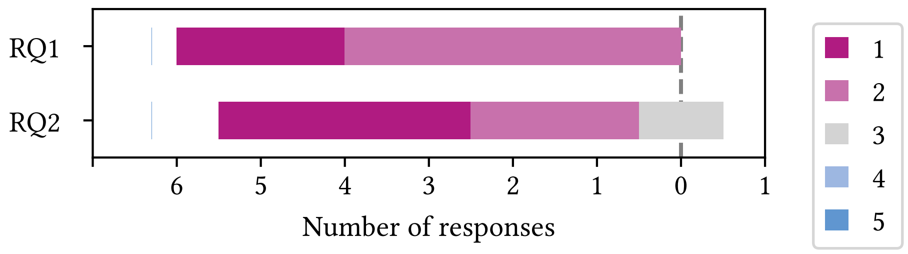
(a) Baseline.



(b) Venue Co-Publishing.

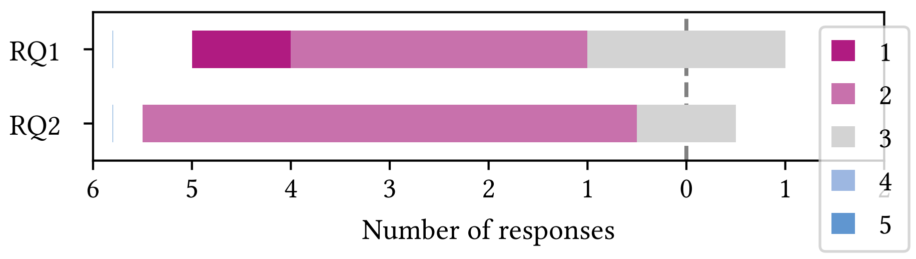


(c) Weighted Influence.

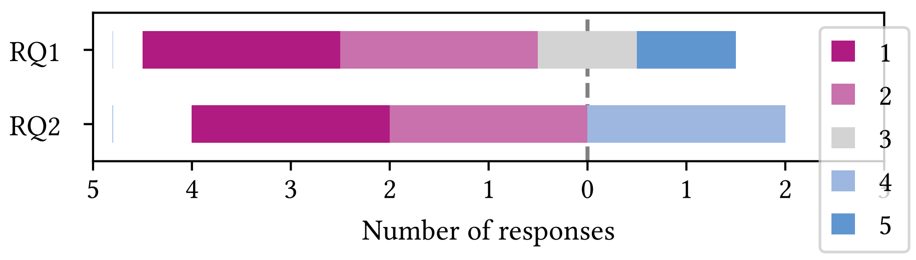


(d) Previously Cited.

Figure 5.2: The subjects’ responses to the questions concerning the recommendations produced by each system.



(e) Previously Cited by Collaborators.



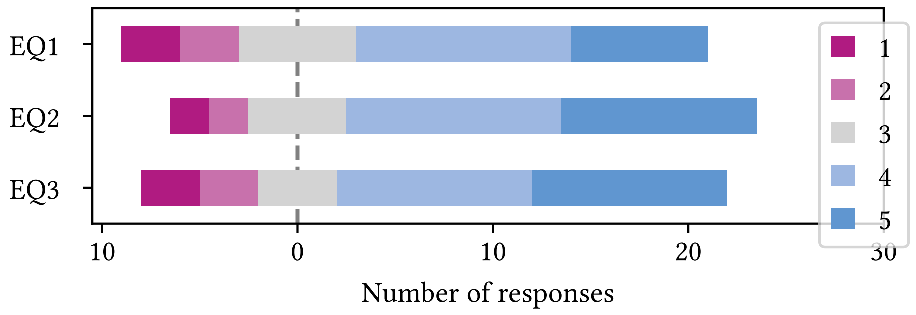
(f) Previously Cited and Topic Search.

Figure 5.2: The subjects' responses to the questions concerning the recommendations produced by each system (cont.).

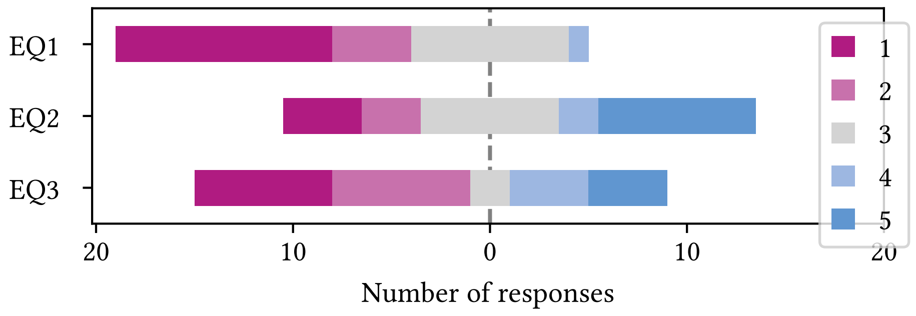
system receives slightly better feedback than these three.

Figure 5.3 shows smaller contrasts between the systems than Fig. 5.2. Compared to the other systems, the baseline receives quite good feedback also on its explanations. The Previously Cited and Topic Search system receives the most positive feedback this time around. In general, all the systems appear to receive consistently more positive responses to the explanation-related questions than the recommendation-related ones.

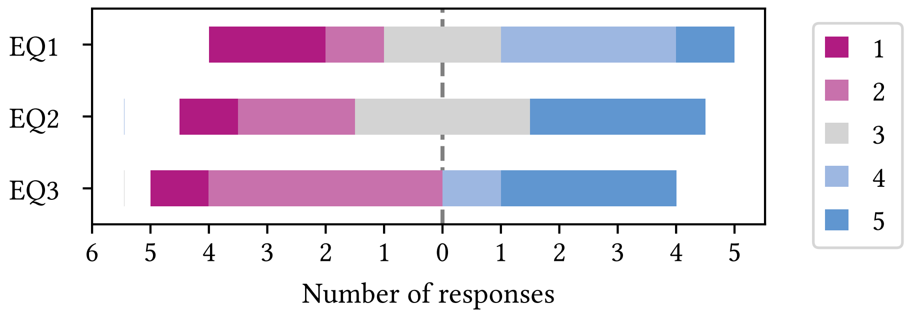
The correlations between the responses to the questions within each category (recommendation-related and explanation-related) are contained in Table 5.8. Unsurprisingly, responses to questions within the same category tend to move in the same direction. Table 5.9 contains the means and standard deviations of each subject's responses to all five types of questions. A few interesting tendencies are revealed by this data. None of the subjects respond positively on average to the questions regarding recommendation usefulness and relevance, except for subject 3, who has a (*very*) slightly positive mean for RQ2. When it comes to the explanation-related questions, subject 3 responds very positively and with high consistency to all questions, and especially to questions EQ2 and EQ3. Subject 5 is the least consistent subject for all the question



(a) Baseline.

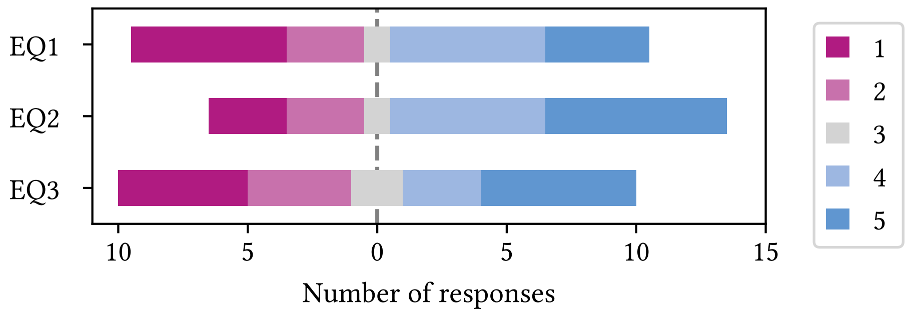


(b) Venue Co-Publishing.

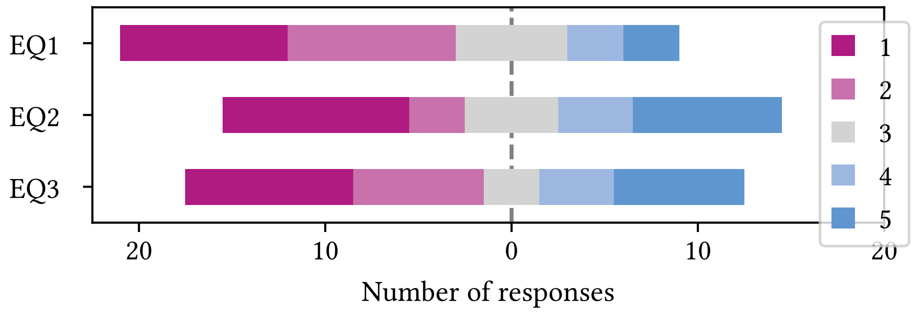


(c) Weighted Influence.

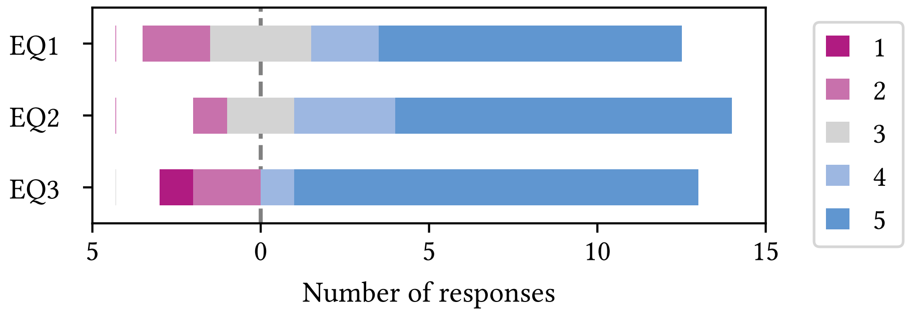
Figure 5.3: The subjects' responses to the questions concerning the explanations produced by each system.



(d) Previously Cited.



(e) Previously Cited by Collaborators.



(f) Previously Cited and Topic Search.

Figure 5.3: The subjects' responses to the questions concerning the explanations produced by each system (cont.).

Table 5.8: Correlations between the responses to the questions within each category.

(a) Recommendation-related questions.

	RQ1	RQ2
RQ1	1	0.84
RQ2	0.84	1

(b) Explanation-related questions.

	EQ1	EQ2	EQ3
EQ1	1	0.61	0.57
EQ2	0.61	1	0.61
EQ3	0.57	0.61	1

Table 5.9: The means and standard deviations of all responses to the five types of questions for each subject.

Subject	RQ1		RQ2		EQ1		EQ2		EQ3	
	$\bar{r}$	σ	$\bar{r}$	σ	$\bar{r}$	σ	$\bar{r}$	σ	$\bar{r}$	σ
1	2.4	0.9	2.4	0.9	3.4	1.0	3.6	0.9	3.2	1.1
2	1.9	0.9	1.8	0.4	2.0	1.0	2.4	1.2	3.6	1.5
3	2.5	0.7	3.1	0.8	3.7	1.2	4.8	0.5	4.9	0.4
4	2.7	1.4	2.0	1.2	1.9	1.1	2.0	1.0	1.6	0.6
5	2.8	1.9	2.5	1.7	3.6	1.9	4.3	1.6	3.2	2.0
6	1.6	0.8	1.2	0.4	2.6	1.2	4.2	0.8	3.7	1.1

types, with standard deviations hovering around 2 across the board.

5.4 Summary

The Frequent Venues recommender system was, practically speaking, excluded from the user study because it did not recommended any papers and was not able to produce any explanations for any of the participants. When the study was already nearing its end, a bug was discovered in the implementation of this system that had lead to candidate papers being unintentionally disregarded. Due to time constraints, however, it was not possible to redo the user study with a fixed version of this system.

The user study showed us that our recommendation methods are not much better at recommending papers than the baseline itself (and in some cases worse). One of our methods, Previously Cited and Topic Search, did, however, appear to be better than the baseline at explaining recommendations. We said in Section 5.1 that all of our recommendation methods are examples of model-intrinsic explainable recommendation. That was not wrong, but an interesting observation to make is that, when the methods are used to produce explanations for recommendations they have not made themselves (as in the user study), they are, in fact, applied for model-agnostic explainable recommendation.

The perhaps biggest flaw with the user study was its limited size. Section 2.1.2 mentioned that user studies performed on information retrieval systems should involve at least two dozen users for any results to be significant [8]. With only six participants, our study did not meet this requirement. The results we got have, nevertheless, given us an idea of how our systems stack up against each other and the baseline system when evaluated by real users. All of the recommender systems used in the user study, except for the baseline and Previously Cited and Topic Search, rely solely on data retrieved from Semantic Scholar to score papers. In theory, it would be possible to perform a similar but larger user study without these two systems, with participants selected from the large pool of researchers represented in the scientific literature graph of Semantic Scholar. This pool would allow not only for a larger group of participants but also for a more heterogeneous one than our study's, which consisted mostly of researchers in computer science and related fields. Another flaw with the user study was that, for most of the recommendations presented to the participants, at least one of the systems (and often several systems) were unable to produce explanations. For the evaluation of explanations, this meant that the number of data points varied for each system. With more time, it would have been possible to find relevant papers for all the participants for which all the systems were able to produce explanations.

The user study was the main focus of our evaluations, and offline and online evaluations were not performed. Online evaluations were excluded due to low traffic on arXivDigest and too little user feedback; during the first couple weeks that our recommender systems were operational, very few of the submitted recommendations were seen by users at all, and no explicit feedback was submitted. Without an increase in overall user activity, it is unlikely that things would have improved noticeably with more time.

With regard to offline evaluations, it would have been possible for us to create a test collection based on historical interactions. Tables 5.6 and 5.7, from which we can see that almost none of the recommendations made by the baseline have been made by our systems, illustrate well some of the issues associated with this type of evaluation. Bellogin, Castells, and Cantador [9] discuss methodologies for comparing top-N recommendations using offline test collections. One approach, i.e., the *TestRatings* methodology is to only consider items that have been rated by the user in the test collection. This, however, "does not test the recommender's ability to identify interesting items from a large pool" [17] and is thus problematic.

Chapter 6

Conclusion

In this thesis, we have introduced novel methods for explainable research paper recommendation that exploit the rich semantic information that is stored in scientific knowledge graphs. To enable these methods to access the information that is available about the users they are producing recommendations for, we have also introduced methods that link users to appropriate entries in scientific knowledge graphs. This chapter summarizes our work by answering the research questions posed in Section 1.2 and concludes this thesis by discussing potential improvements to and directions in which our work can be extended in the future.

6.1 Answering the Research Questions

Section 1.2 posed three research questions. This section revisits and summarizes our experimental chapters to answer these.

RQ1 How can an arXivDigest user be linked to an appropriate entry (or author node) in a scientific knowledge graph?

We have introduced two methods for profile matching, which was defined to be the task of finding the author node (or nodes) in a scientific knowledge graph that is the most likely to represent the same person as a user profile. Both methods search through the Semantic Scholar Open Research Corpus (S2ORC) dataset, which contains the paper records of the scientific literature graph of Semantic Scholar (the specific scientific knowledge graph being used), using Elasticsearch and its default BM25 ranking model. One method searches for author nodes that match the user's name. The other method searches for author nodes that match the user's name and are listed as authors of papers that match the user's topics of interest.

RQ2 In what ways can the information that is available about an arXivDigest user through an external scientific knowledge graph be used to produce scientific literature recommendations for the user?

We have introduced six recommendation methods that use different pieces of information about the user. Their information requirements were laid out in the introduction of Section 5.1 and summed up by Table 5.1. Three methods look at the venues at which the user has published papers at. Two methods look at the authors that the user has cited in the past. The last method looks at who the user has collaborated with (co-authors) and the authors that have been cited by them.

Speaking metaphorically, our recommendation methods have only touched the tip of the iceberg when it comes to exploiting the information stored in scientific knowledge graphs. This applies not only to information about users, but also to information about candidate papers, which we will now address in our answer to the last research question.

RQ3 In what ways can a scientific knowledge graph be used as an external source of information about papers that are candidates for recommendation at arXivDigest when producing scientific literature recommendations? Is it feasible to get hold of the necessary information in reasonable time?

We focus first on the initial part of the question and address the feasibility of retrieving information afterwards. The recommendation methods we have proposed use different pieces of information about the papers that are candidates for recommendation. Their information requirements were laid out in the introduction of Section 5.1 and summed up by Table 5.1. One method simply looks at the venues candidate papers are published at, and does not exploit any of the connections that exist between the papers and other entries in the graph. Two methods look at the authors of candidate papers and the venues at which they have published papers. One of these methods also looks at the citation counts (the derived *influential citation count* metric, to be more specific) of the authors' published papers. The three last methods look at who the authors of candidate papers have been cited by.

With regard to the feasibility of retrieving the information used by the recommendation methods, Section 5.2.1 gave an estimate of the network traffic associated with the retrieval of this information from the Semantic Scholar API, which acted as our interface to the scientific literature graph of Semantic Scholar — the specific scientific knowledge graph used in our implementation. For the two most information-hungry methods, an estimated 6.3 h would be needed to retrieve all needed information, or 2.2 h if papers older than five years were filtered out. That is quite a bit of time, but nothing in the extreme. Section 5.2.1 also discussed how network traffic could (and would) be reduced with caching. Ultimately, and despite some growing pains, all our systems are currently up and running. The answer to the second part of this research question is, evidently, *yes* — it is feasible to get hold of the necessary information in reasonable time.

6.2 Future Work

The are many ways to further develop our work. This section briefly describes three of them.

6.2.1 New User Study

The user study performed in Chapter 5 involved only six subjects. Additionally, one of the recommendation methods that was supposed be evaluated in the user study was excluded due to a bug in its implementation. Section 5.4 mentioned the possibility of performing a larger user study using, e.g., the researchers represented in the scientific literature graph of Semantic Scholar as a participant pool. Possibilities for future work include redoing the user study and/or performing a larger one.

6.2.2 Offline Evaluation at arXivDigest

We mentioned the three requirements for evaluation infrastructures by Schaible et al. [36] in Chapter 2. Their first requirement is the possibility for performing both online and offline evaluations. In Chapter 5, it was a part of our initial plan was to perform both online and offline evaluations of our recommendation methods, using the existing abilities of the arXivDigest evaluation infrastructure for online evaluations. Due to low traffic on arXivDigest, we eventually had to rule online evaluations out. Offline evaluations were not performed either, in part due to them not being possible to perform with the arXivDigest evaluation infrastructure. Future work could involve developing arXivDigest to better facilitate offline evaluations.

6.2.3 Explainable Research Paper Recommendation

The methods of Chapter 5 are all examples of model-intrinsic approaches to explainable recommendation. Section 5.4 pointed out how the the recommendation methods were used in the user study for both model-intrinsic and model-agnostic explainable recommendation. The participants of the study expressed the most satisfaction toward the explanations produced by one of our methods. Future work could involve further exploring and experimenting with ways to use the information stored in scientific knowledge graphs for both model-intrinsic and model-agnostic explainable research paper recommendation.

Bibliography

[1] Qingyao Ai et al. “Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation”. In: *Algorithms* 11.9 (2018), p. 137.

[2] Waleed Ammar et al. “Construction of the Literature Graph in Semantic Scholar”. In: *arXiv preprint arXiv:1805.02262* (2018).

[3] Sören Auer et al. “Towards a Knowledge Graph for Science”. In: *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics.* 2018, pp. 1–6.

[4] Krisztian Balog and Filip Radlinski. “Measuring Recommendation Explanation Quality: The Conflicting Goals of Explanations”. In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 2020, pp. 329–338.

[5] Krisztian Balog, Filip Radlinski, and Shushan Arakelyan. “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.* 2019, pp. 265–274.

[6] Krisztian Balog et al. “Expertise Retrieval”. In: *Foundations and Trends in Information Retrieval* 6.2–3 (2012), pp. 127–256.

[7] Joeran Beel et al. “Exploring the Potential of User Modeling Based on Mind Maps”. In: *International Conference on User Modeling, Adaptation, and Personalization.* Springer. 2015, pp. 3–17.

[8] Joeran Beel et al. “Research-paper recommender systems: a literature survey”. In: *International Journal on Digital Libraries* 17.4 (July 2015), pp. 305–338.

[9] Alejandro Bellogin, Pablo Castells, and Ivan Cantador. “Precision-Oriented Evaluation of Recommender Systems: An Algorithmic Comparison”. In: *Proceedings of the fifth ACM conference on Recommender systems.* 2011, pp. 333–336.

[10] Chandra Bhagavatula et al. “Content-based citation recommendation”. In: *arXiv preprint arXiv:1802.08301* (2018).

[11] Timo Breuer et al. “STELLA: Towards a Framework for the Reproducibility of Online Search Experiments.” In: *OSIRRC@SIGIR.* Ed. by Ryan Clancy et al. Vol. 2409. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 8–11.

[12] Rose Catherine et al. "Explainable Entity-based Recommendations with Knowledge Graphs". In: *arXiv preprint arXiv:1707.05254* (2017).

[13] Hung-Hsuan Chen et al. "CollabSeer: A Search Engine for Collaboration Discovery". In: *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries.* 2011, pp. 231–240.

[14] Linn Marks Collins et al. "ScienceSifter: Facilitating activity awareness in collaborative research groups through focused information feeds". In: *First International Conference on e-Science and Grid Computing (e-Science'05).* IEEE. 2005, 8–pp.

[15] Susan T. Dumais and Jakob Nielsen. "Automating the Assignment of Submitted Manuscripts to Reviewers". In: *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval.* 1992, pp. 233–244.

[16] Travis Ebesu and Yi Fang. "Neural Citation Network for Context-Aware Citation Cecommendation". In: *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval.* 2017, pp. 1093–1096.

[17] Michael D Ekstrand and Vaibhav Mahant. "Sturgeon and the Cool Kids: Problems with Random Decoys for Top-N Recommender Evaluation." In: *FLAIRS Conference.* 2017, pp. 639–644.

[18] Michael Färber. "The Microsoft Academic Knowledge Graph: A Linked Data Source with 8 Billion Triples of Scholarly Data". In: *Proceedings of the 18th International Semantic Web Conference.* ISWC'19. Auckland, New Zealand, 2019, pp. 113–129. DOI: `10.1007/978-3-030-30796-7\_8`. URL: `https://doi.org/10.1007/978-3-030-30796-7%5C_8`.

[19] Kristian Gingstad, Øyvind Jekteberg, and Krisztian Balog. "ArXivDigest: A Living Lab for Personalized Scientific Literature Recommendation". In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (Oct. 2020).

[20] Bela Gipp and Jöran Beel. "Citation Proximity Analysis (CPA): A new approach for identifying related work based on Co-Citation Analysis". In: *ISSI'09: 12th international conference on scientometrics and informetrics.* 2009, pp. 571–575.

[21] Sujatha Das Gollapalli, Prasenjit Mitra, and C Lee Giles. "Similar Researcher Search in Academic Environments". In: *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries.* 2012, pp. 167–170.

[22] Katja Hofmann, Lihong Li, and Filip Radlinski. "Online Evaluation for Information Retrieval". In: *Found. Trends Inf. Retr.* 10.1 (2016), pp. 1–117.

[23] Rolf Jagerman, Krisztian Balog, and Maarten De Rijke. “OpenSearch: Lessons Learned from an Online Evaluation Campaign”. In: *Journal of Data and Information Quality (JDIQ)* 10.3 (2018), pp. 1–15.

[24] Mohamad Yaser Jaradeh et al. “Open Research Knowledge Graph: Next Generation Infrastructure for Semantic Scholarly Knowledge”. In: *Proceedings of the 10th International Conference on Knowledge Capture.* 2019, pp. 243–246.

[25] Diane Kelly. “Methods for Evaluating Interactive Information Retrieval Systems with Users”. In: *Found. Trends Inf. Retr.* 3.1–2 (Jan. 2009), pp. 1–224.

[26] Samiya Khan et al. “A survey on scholarly data: From big data perspective”. In: *Information Processing & Management* 53.4 (2017), pp. 923–944.

[27] Ralf Klamma, Pham Manh Cuong, and Yiwei Cao. “You Never Walk Alone: Recommending Academic Events Based on Social Network Analysis”. In: *International Conference on Complex Sciences.* Springer. 2009, pp. 657–670.

[28] Joseph A Konstan et al. “Grouplens: Applying Collaborative Filtering to Usenet News”. In: *Communications of the ACM* 40.3 (1997), pp. 77–87.

[29] Huajing Li et al. “CiteSeerX: an Architecture and Web Service Design for an Academic Document Search Engine”. In: *Proceedings of the 15th international conference on World Wide Web.* 2006, pp. 883–884.

[30] Olaf Liadal. “Publication Metadata Extraction From Academic Homepages”. 2020.

[31] Kyle Lo et al. “S2ORC: The Semantic Scholar Open Research Corpus”. In: *arXiv preprint arXiv:1911.02782* (2019).

[32] Raymond J Mooney and Loriene Roy. “Content-Based Book Recommending Using Learning for Text Categorization”. In: *Proceedings of the fifth ACM conference on Digital libraries.* 2000, pp. 195–204.

[33] Georgina Peake and Jun Wang. “Explanation Mining: Post Hoc Interpretability of Latent Factor Models for Recommendation Systems”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 2018, pp. 2060–2069.

[34] Elaine Rich. “User Modeling via Stereotypes”. In: *Cognitive science* 3.4 (1979), pp. 329–354.

[35] Mark Sanderson. “Test Collection Based Evaluation of Information Retrieval Systems”. In: *Found. Trends Inf. Retr.* 4.4 (2010), pp. 247–375.

[36] Johann Schaible et al. “Evaluation Infrastructures for Academic Shared Tasks”. In: *Datenbank-Spektrum* 20.1 (Feb. 2020), pp. 29–36.

[37] Semantic Scholar. *Semantic Scholar | About Us.* URL: `https://pages.semanticscholar.org/about-us` (visited on 05/22/2021).

[38] Ayush Singhal et al. “Leveraging Web Intelligence for Finding Interesting Research Datasets”. In: *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*. Vol. 1. IEEE. 2013, pp. 321–328.

[39] Henry Small. “Co-Citation in the Scientific Literature: A New Measure of the Relationship Between Two Documents”. In: *Journal of the American Society for information Science* 24.4 (1973), pp. 265–269.

[40] Marco Valenzuela, Vu A. Ha, and Oren Etzioni. “Identifying Meaningful Citations”. In: *AAAI Workshop: Scholarly Big Data*. 2015.

[41] Herbert Van de Sompel and Carl Lagoze. “All Aboard: Toward a Machine-Friendly Scholarly Communication System”. In: *The Fourth Paradigm* (2009), p. 193.

[42] Zaihan Yang and Brian D Davison. “Venue Recommendation: Submitting Your Paper with Style”. In: *2012 11th International Conference on Machine Learning and Applications*. Vol. 1. IEEE. 2012, pp. 681–686.

[43] Yongfeng Zhang and Xu Chen. “Explainable Recommendation: A Survey and New Perspectives”. In: *Found. Trends Inf. Retr.* 14.1 (2020), pp. 1–101.

[44] Yongfeng Zhang et al. “Explicit Factor Models for Explainable Recommendation based on Phrase-level Sentiment Analysis”. In: *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. 2014, pp. 83–92.

Appendix A

Semantic Scholar Profile Ranking Generation

The commands used to generate Semantic Scholar profile rankings using both the frequency- and score-based methods with max edit distances of both 1 and 2 are shown in Listings A.1 to A.4.

```
python scripts/gen_semantic_scholar_suggestions.py \
    --method frequency \
    --output freq_1.txt
```

Listing A.1: Command used to run the frequency-based method with a max edit distance of 1.

```
python scripts/gen_semantic_scholar_suggestions.py \
    --method frequency \
    --max-edit-distance 2 \
    --output freq_2.txt
```

Listing A.2: Command used to run the frequency-based method with a max edit distance of 2.

```
python scripts/gen_semantic_scholar_suggestions.py --output score_1.txt
```

Listing A.3: Command used to run the score-based method with a max edit distance of 1.

```
python scripts/gen_semantic_scholar_suggestions.py \
    --max-edit-distance 2 \
    --output score_2.txt
```

Listing A.4: Command used to run the score-based method with a max edit distance of 2.