UNIVERSITY OF THE PELOPONNESE & NCSR “DEMOCRITOS”
MSC PROGRAMME IN DATA SCIENCE

Recommending Scientific Papers: A Survey and A Hybrid Approach

by

Grigoris Bouziotopoulos

A thesis submitted in partial fulfillment
of the requirements for the MSc
in Data Science

Supervisor: Thanasis Vergoulis
Scientific Associate

Co-supervisors: Anastasia Krithara, Nikos Platis
Research Assosiate, Assistant Professor

Athens, January 2023

UNIVERSITY OF THE PELOPONNESE & NCSR "DEMOCRITOS"
MSC PROGRAMME IN DATA SCIENCE

Recommending Scientific Papers: A Survey and A Hybrid Approach

by

Grigoris Bouziotopoulos

A thesis submitted in partial fulfillment
of the requirements for the MSc
in Data Science

Supervisor:	Thanasis Vergoulis Scientific Associate
Co-supervisors:	Anastasia Krithara, Nikos Platis Research Assosiate, Assistant Professor

Approved by the examination committee on January, 2023.

(Signature)	(Signature)	(Signature)
....	....	....
Thanasis Vergoulis Scientific Associate	Anastasia Krithara Research Associate	Nikos Platis Assistant Professor

Athens, January 2023

Declaration of Authorship

(1) I declare that this thesis has been composed solely by myself and that it has not been submitted, in whole or in part, in any previous application for a degree. Except where states otherwise by reference or acknowledgment, the work presented is entirely my own.

(2) I confirm that this thesis presented for the degree of Bachelor of Science in Informatics and Telecommunications, has

   (i) been composed entirely by myself

   (ii) been solely the result of my own work

   (iii) not been submitted for any other degree or professional qualification

(3) I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or processional qualification except as specified.

(Signature)

. . . .

Grigoris Bouziotopoulos

Athens, January 2023

Acknowledgments

I would like to acknowledge Dr. Thanasis Vergoulis for the invaluable guidance and support. I would also like to thank Dr. Anastasia Krithara and Dr. Nikos Platis for agreeing to be part of the examination committee and for evaluating the thesis. I would also like to gratefully acknowledge Dr. Serafeim Chatzopoulos, co creator of the Veto recommendation tool, and Dr. Ilias Kanellos, research associates at "Athena" R.C. for participating in the thesis evaluation experiments as well as supporting me and providing valuable feedback along the way. Last but not least, special thanks go to my family, friends, and especially Alexia, for their encouragement and support.

To my family and friends.

Περίληψη

Τα συστήματα συστάσεων άρθρων είναι σημαντικά εργαλεία για να βοηθούν τους μελετητές και τους ερευνητές να ανακαλύπτουν σχετικά και ενδιαφέροντα άρθρα ενόψει του αυξανόμενου όγκου των δημοσιευμένων ερευνών. Σε αυτή τη μελέτη, διεξάγουμε μια έρευνα σχετικά με τις υπάρχουσες προσεγγίσεις συστάσεων άρθρων και αξιολογούμε τις επιδόσεις της δικής μας υλοποίησης, του συστήματος *ExtendedPaperVeto*. Η έρευνά μας καλύπτει μια σειρά προσεγγίσεων και μελετών και εντοπίζει κοινές τάσεις και προκλήσεις στον τομέα. Η αξιολόγηση της υλοποίησής μας, συγκρίνει την απόδοση της προσέγγισης *ExtendedPaperVeto* με δύο άλλες προσεγγίσεις, τις προσεγγίσεις *MongoFTS* και *PaperVeto*, χρησιμοποιώντας διάφορες μετρικές αξιολόγησης. Τα ευρήματά μας δείχνουν ότι οι υβριδικές προσεγγίσεις είναι η περισσότερο χρησιμοποιούμενη προσέγγιση συστάσεων, στις μελέτες που αξιολογήσαμε, ακολουθούμενη από το φιλτράρισμα βάσει περιεχομένου, τις προσεγγίσεις γράφων και το συνεργατικό φιλτράρισμα. Ωστόσο, εντοπίζουμε επίσης αρκετές προκλήσεις και περιορισμούς στο τρέχων πεδίο, συμπεριλαμβανομένης της έλλειψης αναπαραγωγιμότητας και επεκτασιμότητας πολλών προσεγγίσεων, τη περιορισμένη εξέταση της οπτικής του χειριστή τέτοιου συστημάτων, του χαρακτήρα του χρήστη, και την εξάρτηση από αξιολογήσεις ¨εκτός σύνδεσης¨. Στην αξιολόγηση της υλοποίησής μας, παρατηρούμε ότι η προσέγγιση *ExtendedPaperVeto* είχε ελαφρώς χειρότερη επίδοση από την την προσέγγιση *MongoFTS* όσον αφορά τις μετρικές NDCG και AR, αλλά είχε καλύτερη επίδοση σε σχέση με την προσέγγιση *PaperVeto*. Στα συμπεράσματά μας, συζητάμε πιθανές κατευθύνσεις για τις μέλλοντικές έρευνες, που θα μπορούσαν να αντιμετωπίσουν τις προκλήσεις και τους περιορισμούς που εντοπίστηκαν στην έρευνά μας και στην αξιολόγηση της υλοποίησης μας.

Abstract

Paper recommendation systems are important tools for helping scholars and researchers discover relevant and interesting papers in the face of the growing volume of published research. In this study, we conduct a survey of paper recommendation approaches and evaluate the performance of our own implementation, the *ExtendedPaperVeTo* recommender. Our survey covers a range of approaches and studies, and identifies common trends and challenges in the field. Our implementation evaluation compares the performance of the *ExtendedPaperVeTo* approach to two other approaches, the *MongoFTS* and *PaperVeTo* approaches, using various evaluation metrics. Our findings indicate that hybrid approaches are the most common type of paper recommendation approach (PRA) used in the reviewed studies, followed by content-based filtering, graph-based approaches, and collaborative filtering. However, we also identify several challenges and limitations in the current state of the field, including the lack of reproducibility and scalability of many approaches, the limited consideration of the operator's perspective and user characteristics, and the reliance on offline evaluations. In our implementation evaluation, we observe that the *ExtendedPaperVeTo* approach was slightly outperformed by the *MongoFTS* approach in terms of NDCG and AR scores, but performed better than the *PaperVeTo* approach. In the conclusion, we discuss potential directions for future research that could address the challenges and limitations identified in our survey and implementation evaluation.

Contents

List of Tables

List of Figures

List of Abbreviations

CBF	Content Based Filtering
CF	Collaborative Filtering
G	**General** information on the approach
D	already existing **Data** directly taken from the papers used
M	**Methods** which might create or (re-)structure data, which are part of the approach
UP	User Profile
KP	Key Phrase
TM	Topic Model
KG	Knowledge Graph
MP	Metapath
RW	Random Walk
RWR	Random Walk with Restart
AML	Advanced Machine Learning
CS	Cosine Similarity
GCN	Graph Citation Networks

BERT	Bidirectional Encoder Representations from Transformers
p	paper input
k	keywords input
u	user input
o	other input
?	unkown input
TF-IDF	Term Frequency-Inverse Document Frequency
CF-IDF	Concept Frequency-Inverse Document Frequency
HCF-IDF	Hierarchical Concept Frequency-Inverse Document Frequency
CNN	Convolution Neural Network
NN	Neural Network
PRA	Prominent Recommendation Approach
POI	Paper of Interest
ALS	Alternating Least Squares
HIN	Heterogeneous Information Network
SKW	Semantic Keyword Weighting
SBERT	Sentence Bidirectional Encoder Representations from Transformers
LSTM	Long Short-Term Memory
LOD	Linked Open Data
NCN	Neural Citation Network

MART	Multiple Additive Regression Trees
L2R	Learning 2 Rank
RF	Random Forests
SPRD	Scholarly Paper Recommendation Dataset
DBLP	dblp computer science bibliography
ArnetMiner	AMiner
ACM	ACM Digital Library
MAG	Microsoft Academic Graph
SRDP	Serendipity Score
NDCG	normalised discounted cumulative gain
AR	average rating
CTR	click-through rate
MRR	mean reciprocal rank
MAP	mean average precision
CTM	Citation Translation Model
RNN	Recurring Neural Network
TDNN	Time Delay Neural Network
AKR	author-specified keywords based retrieval
wAKR	weighted author-specified keywords based retrieval
TP	True Positive
FP	False Positive
TN	True Negative

FN	False Negative
MSCN	multi stage citation network
CAR	citation article recommendation
ML-BDTR	machine learning-based dynamic treatment regimen
GAN-HBNR	Generative Adversarial Network - Human Behavioural Network Representation
BNR	bibliographic network representation
BPRMF	Bayesian Personalized Ranking Matrix Factorization
CML	Collaborative Metric Learning
KGA	Knowledge Graph Attention
DKA	Deep Knowledge-Aware Network
AUC	Area Under the ROC curve
RRF	Reciprocal Rank Fusion
FTS	Full Text Search
MongoFTS	MongoDB full text search
DBLP-ArtSim	DBLP Article Similarities
N/A	Not available
PRS	Paper Recommendation System

Chapter 1

Introduction

Paper recommendation systems have become increasingly important tools aiding scholars and researchers discover relevant and interesting papers in the face of the growing volume of published research [8, 9]. These systems use a variety of approaches to recommend papers to users, including content-based filtering (CBF) [10, 11], collaborative filtering (CF) [3, 12], graph-based approaches [4, 13], and hybrid approaches [12, 14]. However, the field of paper recommendation is still in its early stages and there are several challenges and limitations that need to be addressed [7, 15].

1.1 Problem description

The lack of reproducibility and scalability of many paper recommendation approaches, the limited consideration of the operator's perspective and user characteristics, and the reliance on offline evaluations are indicative challenges of the field [8, 16, 17]. Overall, these challenges and limitations highlight the need for more comprehensive and realistic evaluations of paper recommendation approaches, as well as the development of more reproducible, scalable, and user-centered systems.

The first objective of this thesis is to conduct a survey of paper recommendation system (PRS) approaches. The survey aims at identifying the most common types of state-of-the-art PRS, classifying them based on the approach they follow, and identifying challenges and limitations in the current state of the field. The survey covers various datasets, evaluation methods, and prominent recommendation approaches such as content-based filtering, collaborative filtering, graph-based, and hybrid approaches.

The second objective of this thesis is to develop a novel PRS. Due to the high volume of published research, graph-based recommendation methods are becoming more popular [4, 5, 7, 13, 14, 18–21]. Therefore, we propose a hybrid recommendation method, based on graphs. The proposed approach is open-source, and will be evaluated by conducting a user study, as well as using evaluation metrics.

In summary, the proposed research will contribute to the field of PRS in two ways: (a) by conducting a recommendation survey, which provides a comprehensive understanding of the current state of the field, its limitations and the most common approaches used, and (b) by providing an open-source PRS hybrid implementation which users can use as a baseline for their research or even expand it.

1.2 Thesis structure

The rest of this thesis is organised as follows. Chapter 2 introduces concepts to assist the user through the remainder of this thesis. Chapter 3 provides a thorough look into the short paper recommendation survey we have conducted, the classification of recommendation approaches, the overview of the methods, datasets and evaluations used, as well as, a survey summary. Chapter 4 presents our work in recommending scientific papers, based on users' interests. Chapter 5 presents our evaluation methodology, experimental setup for the aforementioned work, as well as a discussion on the results and limitations. Finally, chapter 6 concludes this thesis, summarising our work, while providing suggestions for future works.

Chapter 2

Background

In this chapter, we discuss some core concepts that are used in the reminder of this thesis. Initially, we provide a historic overview on paper recommendation methods (section 2.1). Afterwards, we describe the notion of heterogeneous information networks (HINs) and present the formal definition of HINs (section 2.2). We also describe scholarly information networks, since they are a specific version of a HIN (section 2.3), along with the concept of metapaths, which is instrumental for mining, analysing and exploring such networks (section 2.4). Finally, we outline the concept of full text search (section 2.5).

2.1 Historic Overview of Paper Recommendation Methods

The field of paper recommendation systems (PRS) has been an active area of research for several decades. Early PRS methods were based on simple techniques such as keyword-based matching and citation analysis [22]. These methods relied on matching keywords between papers and user queries, or identifying papers that were frequently cited by other papers. However, these early methods had limitations in terms of the accuracy and scalability of the recommendations.

In the late 1990s and early 2000s, content-based filtering (CBF) methods were proposed [23], which used the content of papers and user profiles to recommend papers. These methods represented papers and users as vectors of features and used similarity metrics such as cosine similarity to recommend papers that were most similar to the user's preferences or interests. CBF methods improved the accuracy of recommendations by considering the content of papers, but they still had limitations in terms of scalability and the ability to handle new users or papers.

Collaborative filtering (CF) methods [24], which used the ratings or preferences of users to recommend papers, were also introduced around this time. These methods represented users and papers as matrices of ratings and used techniques such as matrix factorization to estimate the ratings of unrated items. CF methods improved the scalability and the ability to handle new users or papers, but they still had limitations in terms of the sparsity of the rating data and the cold-start problem for new users or items.

In the late 2000s and early 2010s, graph-based methods [25], which used the relationships between papers and authors to recommend papers, gained popularity. These methods represented papers and authors as nodes in a graph, and used techniques such as random walk and PageRank to recommend papers that were most central or influential in the graph. Graph-based methods improved the ability to handle new papers or authors, but they still had limitations in terms of the completeness and quality of the graph data and the interpretability of the recommendations.

In recent years, hybrid methods [26], which combine multiple techniques, have become the most common type of PRS method. Hybrid methods combine the strengths of different techniques such as content-based, collaborative, and graph-based methods to improve the accuracy, scalability, and diversity of the recommendations. Hybrid methods are able to overcome the limitations of the individual techniques, but they still have challenges in terms of the complexity and interpretability of the methods.

2.2 Heterogeneous information networks (HINs)

Heterogeneous Information Networks (HINs) are complex networks that consist of nodes and edges representing different types of objects and their relationships. These networks can be used to represent various kinds of information, such as scientific papers, authors, and citations in the context of scholarly paper networks [1, 27–29].

HINs can be used to model complex real-world systems and facilitate the discovery of hidden patterns and relationships [1, 27, 29–31]. Researchers have proposed various methods for mining and analyzing HINs, such as metapath-based similarity search [27], community search [30], cohesive subgraph search [29], and probabilistic model for linking named entities [31].

In addition, HINs can be used to support various kinds of queries, such as relational community detection and search [32], structure-aware parameter-free group query [33], and classification using metapath contexts [34]. These methods have demonstrated their effectiveness and efficiency on various real-world HINs, and have the potential to facilitate the discovery of new insights and knowledge in various domains. According to [1], a HIN is formally defined as follows :

Definition 1 (HIN) *A* HIN *is a tuple* $\mathcal{H} = \langle V, E, O, R, \phi, \psi \rangle$*, where* V *and* E *are the nodes and edges of a directed multigraph, respectively;* O *and* R $(|O| > 1, |R| > 1)$ *are the sets of the node and edge types, respectively, while* $\phi : V \to O$ *and* $\psi : E \to R$ *are the mapping functions that determine the type of each node* $v \in V$ *and each edge* $e \in E$*, respectively.*

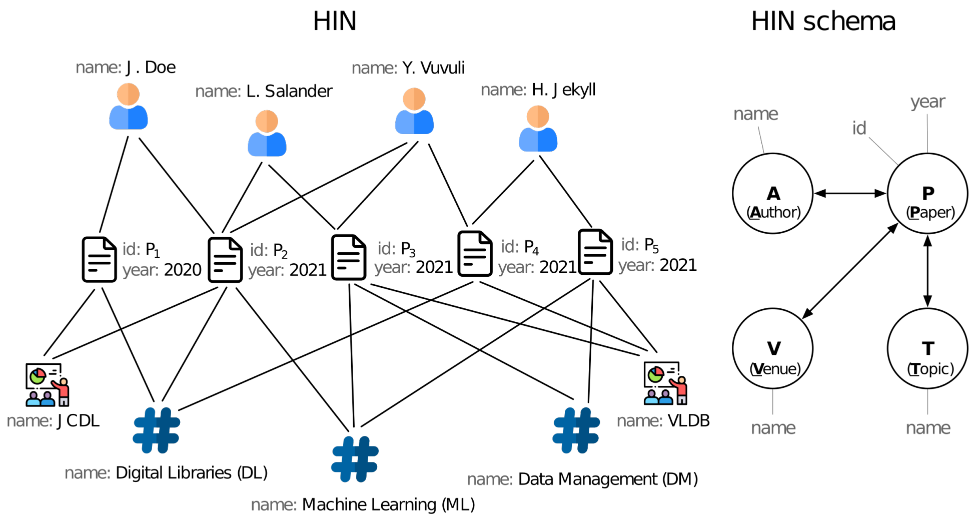


Figure 2.1: Example of a scholarly information network, modelled as a HIN (with its schema), containing nodes for authors, papers, topics and venues [1].

Figure 2.1 illustrates an example HIN (and its schema) capturing scholarly knowledge. It consists of nodes representing papers (`P`), authors (`A`), venues (`V`), and topics (`T`). Three types of (bidirectional) edges are present in this example network: edges between authors and papers, denoted as `AP` or `PA`, edges between papers and topics, denoted as `PT` or `TP`, and edges between papers and venues, denoted as `PV` or `VP`. The first edge type captures the authorship of papers, the second one encodes the information that a particular paper is written on a particular topic, while the last one captures the fact that a paper has been published in a particular venue.

2.3 Scholarly Information Networks

Scholarly information networks are systems of interconnected entities that enable the sharing and dissemination of scholarly research and knowledge. For instance, these networks can include academic journals, databases, research institutions, and other physical and digital resources, and may be organized around specific disciplines or fields of study. It is evident that such networks can be modelled as HINs (see section 2.2).

In recent years, there have been a number of efforts to create more comprehensive and interconnected scholarly information networks by using knowledge graphs. For example, the Open Research Knowledge Graph project aims to create a "next-generation infrastructure for semantic scholarly knowledge" by using a knowledge graph to connect research papers, authors, institutions, and other related entities [35]. The Literature Graph, constructed by the Semantic Scholar team, is another example of a knowledge graph for scholarly literature, which aims to improve the discoverability and accessibility of research [36]. The Microsoft Academic Graph is yet another example of a knowledge graph for scholarly literature, which aims to provide a comprehensive overview of research in various fields [37, 38].

Other efforts to create more comprehensive scholarly information networks include the use of linked data and ontologies to represent research concepts and relationships [37, 38]. The OpenAIRE Research Graph is an example of such an effort, which uses a data model and linked data to represent research resources and their relationships [39, 40]. These approaches can help to improve the interoperability and reuse of research data, as well as the tracking and assessment of research impact [37].

In addition to these more formalized approaches, there are also more informal scholarly information networks, such as academic social networks, which are created through the relationships between researchers and their collaborations [41]. These networks can also play a role in the dissemination and exchange of research and knowledge. In this thesis, we model such networks, as the ones outlined above, as heterogeneous information networks (HINs).

2.4 Metapaths and Metapath-based Similarity

A metapath is a sequence of edges that connect nodes in a HIN [1, 27, 28, 34]. Metapaths can capture rich semantic relationships between nodes that are not captured by individual edges, and can be used to define various kinds of paths in HINs [27–29, 34]. For example, in the HIN of Figure 2.1, the metapath $\langle \texttt{P} \xrightarrow{\texttt{PA}} \texttt{A} \xrightarrow{\texttt{AP}} \texttt{P} \rangle$ relates papers with common authors e.g., paper P2 has two authors ”L.Salander” and ”Y. Vuvuli” in common with P3. Also, a metapath corresponds to a path on the schema of the HIN.

Given a metapath m, every path in the HIN that complies to the sequence of node and edge types specified by m is referred to as an *instance* of m. More formally [1]:

Definition 2 (Metapath & metapath instance) *Given a HIN* $\mathcal{H} = \langle V, E, O, R, \phi, \psi \rangle$, *a* metapath m *on* $\mathcal{H}$ *is a sequence* $m = \langle o_1 \xrightarrow{r_1} o_2 \dots \xrightarrow{r_{n-1}} o_n \rangle$, *where* $o_i \in O$, $r_j \in R$ $\forall i, j$, *and* $n-1$ *is the metapath length. An* instance *of* m *is any path* $\langle v_1 \xrightarrow{e_1} v_2 \dots \xrightarrow{e_{n-1}} v_n \rangle$ *for which* $v_i \in V$, $e_j \in E$, $\phi(v_i) = o_i$, *and* $\psi(e_j) = r_j$ $\forall i, j$.

For the sake of simplicity [1, 27], we denote a metapath without edge types as $m = \langle o_1 o_2 \dots o_n \rangle$, when there is only a single edge type between any pair of node types. Based on this convention, the metapath $m = \langle \texttt{P} \xrightarrow{\texttt{PA}} \texttt{A} \xrightarrow{\texttt{AP}} \texttt{P} \rangle$, that connects papers with common authors, can be represented by its more compact form as $m = \langle \texttt{PAP} \rangle$, with the path ⟨“P2” → “L. Salander” → “P3”⟩ being an instance of m. It is also worth noting that metapaths essentially define *metapath-*

based views of the HIN, each containing only nodes of the first and last entity types in the respective metapath and having one (weighted) edge to represent metapath instances connecting these entities.

As mentioned before, metapaths can be used for various kinds of analysis and mining tasks in HINs, such as similarity search [27, 29], community search [29, 30], and classification [34]. Metapath-based similarity refers to a measure of the similarity between two nodes in a HIN based on the metapaths that connect them [27, 29, 34]. Metapath-based similarity can capture complex and higher-order relationships between nodes that are not captured by traditional similarity measures, such as node degree or common neighbors [27, 29]. As demonstrated on the first paragraph, papers “P2” and “P3” in the network of 2.1 seem fairly similar since they have two common authors, namely ”L. Salander” and ”Y. Vuvuli”(PAP metapath).

There are various methods for calculating metapath-based similarity, depending on the specific characteristics and needs of the HIN and the application. Some common methods include PathSim [27], which calculates metapath-based similarity using the normalized frequency of common metapaths between two nodes; MetaSim [29], which calculates metapath-based similarity using a weighted sum of common metapaths between two nodes, where the weights reflect the importance or relevance of each metapath; MetaPath2Vec [34], which calculates metapath-based similarity using a deep learning approach, which learns node representations based on metapath contexts and uses these representations to measure the similarity between nodes; and JoinSim [1, 42], which calculates metapath-based similarity using cosine similarity at some point. Cosine similarity (CS) [43] between two vectors **A** and **B** is defined as:

$$\text{Cosine Similarity (CS)} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{|\mathbf{A}|\,|\mathbf{B}|}$$

where $\mathbf{A} \cdot \mathbf{B}$ is the dot product of the vectors, $|\mathbf{A}|$ and $|\mathbf{B}|$ are the Euclidean norms of the vectors. Cosine similarity ranges from 0 (indicating complete dissimilarity) to 1 (indicating complete similarity). It is often used as a measure of similarity for text data, as it takes into account the frequency of terms in the documents, but it can also be used for any type of vector data [44].

Therefore, JoinSim similarity can be defined more formally [1, 45]; consider a knowledge graph $\mathcal{G}$ and one of its metapaths $m = F - \cdots - L$, where $F, \ldots, L$ are valid entity/node types in $\mathcal{G}$. In addition, let $\{f_1, \ldots, f_n\}$ and $\{l_1, \ldots, l_k\}$ be the sets of $\mathcal{G}$ nodes of type F and L, respectively. For each f_i (with $i \in [1, n]$), let $\mathbf{v^m_{f_i}}$ be a vector of size k, where $\mathbf{v^m_{f_i}}[j]$ corresponds to the number of m instances connecting f_i to l_j. Then, the JoinSim similarity of f_α and f_β according to m is given by the cosine similarity of the vectors $\mathbf{v^m_{f_\alpha}}$ and $\mathbf{v^m_{f_\beta}}$, i.e.,:

$$sim(f_\alpha, f_\beta, m) = \frac{\mathbf{v^m_{f_\alpha}} \cdot \mathbf{v^m_{f_\beta}}}{||\mathbf{v^m_{f_\alpha}}||\ ||\mathbf{v^m_{f_\beta}}||}$$

The intuition of this formula is that the JoinSim similarity of f_α and f_β is large if they are connected with a comparable number of paths to a similar set of nodes of type L [45]. In the example of Figure 2.1, given the metapath `PAP`, JoinSim first constructs for each paper a vector containing other papers with the same author, and then calculates similarity scores between the papers based on these vectors.

2.5 Full Text Search

Full text search (FTS) is a powerful tool for retrieving information from large text collections, allowing users to search for specific words or phrases within the full text of a document rather than just the metadata or titles.

Google Scholar [46] is one example of a platform that utilizes full text search, allowing users to search for scholarly literature across a range of disciplines and sources. In addition to searching for particular keywords, one can augment a full text search with search features like fuzzy-text and synonyms. Therefore, suppose you are looking for a dish on a restaurant menu, the results for a word such as “pasta” would return not only items such as “Pasta with meatballs” but could also return items like “Fettuccine Carbonara” using a synonym, or “Bacon and pesto flatbread” using a fuzzy search. Figure 2.2 illustrates such an example[1].

[1]Image retrieved from url `https://www.mongodb.com/basics/full-text-search`

There are several approaches to implementing full text search, including traditional string searching [47], indexing [48], vector space modeling [49], and probabilistic models [50].

String searches are algorithms that search for consecutive characters in a larger text field. Those searches will be performed character per character and can be relatively slow. Another technique often used for string searches is the use of regular expressions. Those expressions represent a search pattern and are supported by most modern programming languages. Some algorithms exist to increase the speed of those searches if the text to be searched is more significant. The *Rabin-Karp algorithm*, which looks for matching substrings, is fast and easy to implement, the *Knuth-Morris-Pratt algorithm* looks for all instances of a matching character, increasing the speed for multiple matches in a string and the *Boyer-Moore algorithm* which performs explicit character comparisons at certain "character windows" instead of a brute-force search [47].

Vector space modeling represents documents and queries as vectors in a high-dimensional space and uses techniques from linear algebra to retrieve relevant documents [49].

Probabilistic models, such as Latent Dirichlet Allocation (LDA), use statistical techniques to identify the underlying themes or topics in a collection and retrieve relevant documents based on those themes [50].

Indexing can be done in different ways, such as batch indexing or incremental indexing. The index then acts as an extensive glossary for any matching documents. Various techniques can then be used to extract the data. Apache Lucene [2], the open source search library, uses an inverted index to find the matching items [48].

[2] `https://lucene.apache.org/`

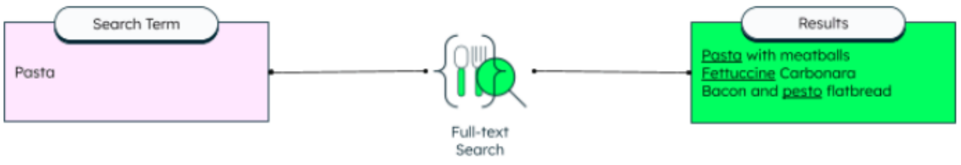


Figure 2.2: Example of a full text search.

MongoDB [3] is a popular NoSQL database that also supports full text search using inverted indexing. The key to an efficient full text search is index creation [51]. Essentially, the index creation process goes through each text field of a dataset. For each word, it will start by removing any diacritics (marks placed above or below letters, such as é, à, and ç in French). Then, based on the used language, the algorithms will remove filler words and only keep the stem of the terms. This way, "to eat," "eating," and "ate" are all classified as the same "eat" keyword. It then changes the casing to use only either uppercase or lowercase. An example of such an indexing process (for MongoDB) is presented on figure 2.3 [4].

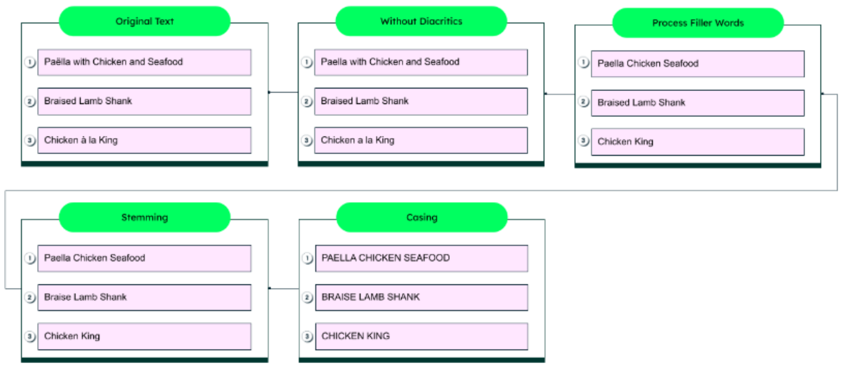


Figure 2.3: Example of the inverted indexing process for MongoDB part 1.

As presented in figure 2.4, the index is then created by adding each of these words with a reference to which document it can be found in [4].

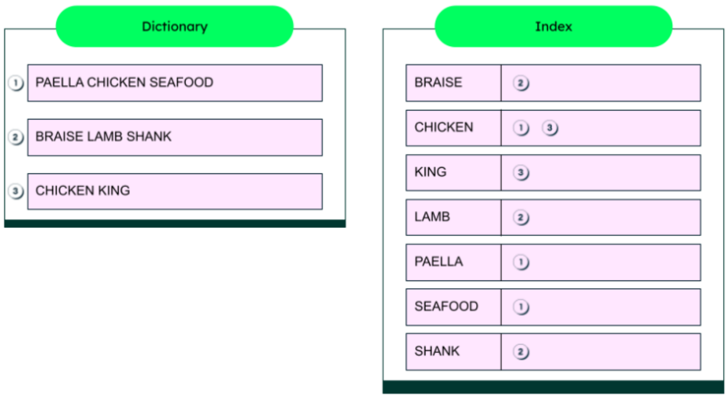


Figure 2.4: Example of the inverted indexing process for MongoDB part 2.

[3]https://www.mongodb.com/
[4]Image retrieved from https://www.mongodb.com/basics/full-text-search

Chapter 3

Paper Recommendation Survey

In this chapter, we present our literature survey. We initially outline the scope of our survey, as well as some meta information for the selected papers (section 3.1). Afterwards, we classify the selected scientific papers based on the approach they follow (section 3.2), discussing the advantages and disadvantages of the most prominent ones (section 3.3). Additionally, we provide an overview of the reviewed article approaches (section 3.4). Furthermore, we present the datasets (section 3.5) and the different evaluation types used by the reviewed papers (section 3.6). The chapter concludes with a short summary of the survey and a discussion of potential directions for future work (section 3.7).

3.1 Scope and Metadata

Our literature search was performed on the following digital libraries: BIP! Finder [52], Google-Scholar [1] and Springer [2]. We considered publications containing the words "paper", "article" or "publication" along with derivative words of "recommend" in their titles. Note that we have selected only papers written in English and published between 2017 and 2022. We assessed their relevance based on their title and abstract; in case it was still unclear, we also considered the full text (if available). Referenced publications were also considered. We refrained from includ-

[1]`https://scholar.google.com/`
[2]`https://link.springer.com/`

ing works where authors proposed a recommendation system, but did not proceed to an implementation to evaluate the proposed solution. Such an example is [53]. In total 40 papers were briefly evaluated. We chose the 20 most relevant ones that met the aforementioned criteria.

Table 3.1 illustrates some interesting metadata of the selected papers, rounded to the nearest integer value. Specifically, the average publication year was 2018.6, the average page count was 9.2, the average citation count was 29.8, the average reference count was 29.3 and the average author count was 3.8. For each of those five metadata (publication year, page count, citation count, reference count, author count) a separate distribution is presented on figure 3.1.

	Year	Page Count	Citation Count	Reference Count	Author Count
AVG	2019	9	30	29	4

Table 3.1: Average Metadata Values.

3.2 Classification

In this section, we present an existing classification based on the recent literature surveys (see 3.2.1) as well as the classification we have selected (subsection 3.2.2).

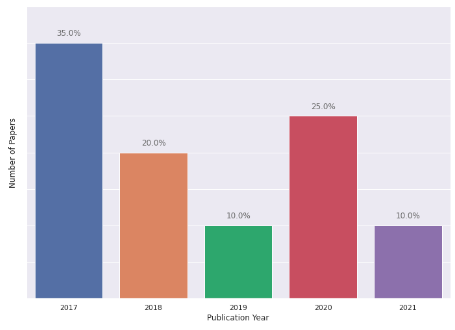
3.2.1 Existing Classification

To the best of our knowledge there have been 4 recent literature surveys reviewing paper recommendation systems: two were conducted on 2019 [2, 9], one on 2020 [44] and the most recent one on 2022 [16]. There is also another older survey [8], yet quite influential,[3] that classifies papers based on their underlying recommendation principle, into seven categories: stereotyping, content-based filtering (CBF), collaborative filtering (CF), co-occurrence, graph-based, global relevance and hybrid models. In [2] the classes content-based filtering, collaborative filtering, graph-based methods, hybrid methods and other models are used. [9] survey utilises the classes content-based recommendation, hybrid recommendation, graph-based recommendation and

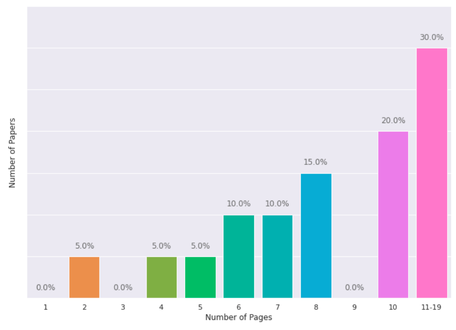
[3]Based on the number of citations.

recommendation based on deep learning. [44] distinguishes four categories where the identification of relevant documents is based on content, metadata, collaborative filtering, and citations. The authors of the most recent survey [16] argue that the classification of papers into only four categories: content-based filtering, collaborative filtering, graph-based and hybrid systems does not seem sufficient. That is due to two main reasons:

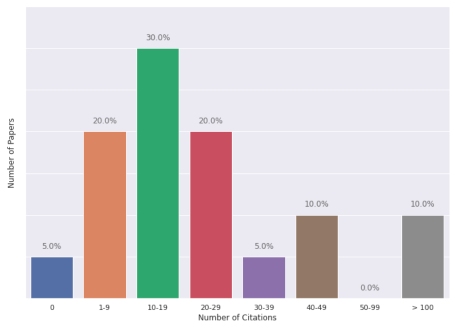
1. Works may not always clearly state the method they are using or individual



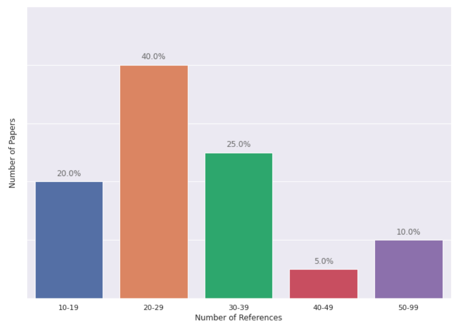
(a) Publication Year



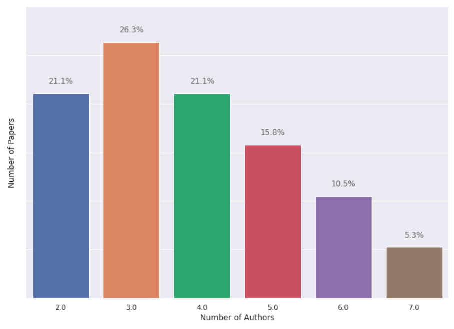
(b) Page Count



(c) Citation Count



(d) Reference Count



(e) Author Count

Figure 3.1: Paper Metadata about author, page, citation, reference counts and publication year.

category definitions by the authors may not always be accurate or disregard other possible ones.

2. CBF, CF and hybrid methods can be defined precisely but graph based ones cannot. That is because recent graph methods tend to use a graph only as one part of the recommendation and may use user-interaction data or descriptions of paper features which would also render them as both CBF and CF [16]. One example from our review is [19] where the authors apart from the graph model also leverage papers' citation proximity, authors' collaboration proximity, venues' information, labeled information, and topical relevance to generate personalized paper recommendations.

Therefore, the authors propose a new type of categorization based on twenty different dimensions split into three main pillars: by general information on the approach (G), already existing data directly taken from the papers used (D) and methods which might create or (re-)structure data, which are part of the approach (M) [16]. The dimensions provided by [16] are presented below for reference:

- **(G) Personalised**: The approach produces personalised recommendations. The recommended items depend on the person using the approach, if personalisation is not considered, the recommendation solely depends on the input keywords or paper. This dimension is related to the existence of user profiles.

- **(G) Input**: The approach requires some form of input, either a paper (p), keywords (k), user (u) or something else, e.g. an advanced type of input (o). Hybrid forms are also possible. In some cases the input is not clearly specified throughout the paper so it is unknown (?).

- **(D) Title**: The approach utilises titles of papers.

- **(D) Abstract**: The approach utilises abstracts of papers.

- **(D) Keyword**: The approach utilises keywords of papers. These keywords are usually explicitly defined by the authors of papers, contrasting key phrases.

- **(D) Text**: The approach utilises some type of text of papers which is not clearly specified as titles, abstracts or keywords. In the evaluation this approach might utilise specified text fragments of publications.

- **(D) Citation**: The approach utilises citation information, e.g. numbers of citations or co-references.

- **(D) Historic Interaction (HI)**: The approach uses some sort of historic user-interaction data, e.g. previously authored, cited or liked publications. An approach can only include historic user-interaction data if it also somehow contains user profiles.

- **(M) User Profile (UP)**: The approach constructs some sort of user profile or utilises profile information. Most approaches using penalisation also construct user profiles but some do not explicitly construct profiles but rather encode user information in the used structures.

- **(M) Popularity**: The approach utilises some sort of popularity indication, e.g. CORE rank, numbers of citations or number of likes.

- **(M) Key Phrase (KP)**: The approach utilises key phrases. Key phrases are not explicitly provided by authors of papers but are usually computed from the titles and abstracts of papers to provide a descriptive summary, contrasting keywords of papers.

- (M) **Embedding**: The approach utilises some sort of text or graph advanced embedding technique, e.g. Bidirectional Encoder Representations from Transformers (BERT) or Doc2Vec. Simple form of embeddings such as bag of words [11] etc. are not considered in this dimension.

- **(M) Topic model (TM)**: The approach utilises some sort of topic modeling, e.g. Latent Dirichlet Allocation (LDA).

- **(M) Knowledge Graph (KG)**: The approach utilises or builds some sort of knowledge graph. This dimension surpasses the mere incorporation of a graph which describes a network of nodes and edges of different types. A knowledge graph is a sub-category of a graph.

- **(M) Graph**: The approach actively builds or directly uses a graph structure, e.g. a knowledge graph or scientific heterogeneous network. Utilisation of a neural network is not considered in this dimension.
- **(M) Metapath (MP)**: The approach utilises metapaths. They usually are composed from paths in a network.
- **(M) Random Walk (with Restart) (RW(R))**: The approach utilises Random Walk or Random Walk with Restart.
- **(M) Advanced Machine Learning (AML)**: The approach utilises some sort of advanced machine learning component in its core such as a neural network. Utilisation of established embedding methods which themselves use neural networks (e.g. BERT) are not considered in this dimension. The authors did not consider traditional and simple ML techniques such as k means in this dimension but rather mention methods explicitly defining a loss function, using multi-layer perceptrons or Graph Citation Networks (GCNs).
- **(M) Crawling**: The approach conducts some sort of web crawling step.
- **(M) Cosine Similarity (CS)**: The approach utilises cosine similarity at some point or a similar similarity method such as Jaccard similarity or Pearson correlation coefficient.

3.2.2 Selected Classification

Taking into consideration the classification approaches presented in the previous recommendation surveys [2, 8, 9, 16, 44], we initially classify the existing approaches into four classes, based on the prominent recommendation approach (PRA) they use. The PRA is essentially the predominant class an approach belongs to. The four classes are:

1. Content Based Filtering

2. Collaborative Filtering

3. Graph-based

4. Hybrid

Out of the 20 reviewed papers:

- 6 used pure content based filtering approaches [10, 11, 15, 54–56] while 5 used content based filtering as part of their hybrid approach [6, 7, 19, 20, 57]

- only 2 utilised pure collaborative filtering approaches [3, 12], while 6 used collaborative filtering as part of their hybrid approach [6, 14, 17, 20, 21, 57]

- 4 went with pure graph based approaches [4, 5, 13, 18], while 5 [7, 14, 19–21], used graphs as part of their hybrid approach

- 8 used hybrid approaches [6, 7, 14, 17, 19–21, 57]

Table 3.2 illustrates the distribution of papers based on the prominent recommendation approach they used.

CBF	CF	Graphs	Hybrid
30%	10%	20%	40%

Table 3.2: Distribution of papers per PRA.

Additionally, we also classify approaches based on the dimensions described in [16] due to sharing similar concerns with the authors (see section 3.2.1 for more details). However, we add three more dimensions on the data (D) pillar, that we believe are equally important: topic, venue and author, defined as follows:

1. **(D) Topic**: The approach utilises topic of papers.

2. **(D) Author**: The approach utilises authors of papers.

3. **(D) Venue**: The approach utilises venues of papers.

Therefore, we perform a composite classification where a paper can use a PRA as well as having certain of the aforementioned dimensions. Table 3.3 classifies the observed approaches according to the PRAs they used as well as their respective dimensions.

Work	PRA	General		Data									Methods											
		Personalised	Input	Title	Abstract	Keyword	Text	Citation	Author	Topic	Venue	HI	UP	Popularity	KP	Embedding	TM	KG	Graph	Path	RW(R)	AML	Crawling	CS
[10]	CBF	✓	p	✓				✓	✓			✓	✓	✓								✓		
[6]	Hybrid		p	✓	✓			✓																✓
[54]	CBF	✓	p	✓	✓	✓			✓		✓			✓	✓									✓
[57]	Hybrid		k	✓	✓	✓								✓								✓	✓	
[5]	Graph	✓	u				✓	✓	✓	✓	✓	✓	✓					✓	✓	✓	✓	✓		
[11]	CBF		p					✓							✓							✓		
[56]	CBF		p	✓		✓		✓	✓														✓	✓
[20]	Hybrid	✓	pu							✓		✓	✓			✓	✓		✓					
[19]	Hybrid	✓	pu				✓	✓	✓	✓	✓	✓	✓			✓	✓	✓	✓			✓		✓
[4]	Graph	✓	ko	✓	✓	✓										✓		✓	✓				✓	✓
[55]	CBF	✓	pu	✓								✓	✓										✓	✓
[13]	Graph	✓	p					✓				✓	✓						✓		✓			
[3]	CF	✓	pu				✓					✓	✓				✓							
[18]	Craph		p	✓	✓		✓	✓						✓				✓	✓			✓		
[15]	CBF		p				✓									✓								✓
[17]	Hybrid	✓	pu	✓	✓	✓						✓	✓	✓										✓
[12]	CF	✓	u	✓	✓							✓	✓			✓						✓		
[21]	Hybrid		p					✓											✓					✓
[14]	Hybrid	✓	pu	✓	✓	✓						✓	✓						✓		✓			
[7]	Hybrid	✓	u	✓	✓	✓		✓				✓	✓			✓		✓	✓			✓		

Table 3.3: Classification of Existing Recommendation Approaches.

3.3 Prominent Recommendation Approaches

In this section, we describe the four PRAs: content based filtering (subsection 3.3.1), collaborative filtering (subsection 3.3.1), graph (subsection 3.3.3) and hybrid (subsection 3.3.4) as well as the advantages and disadvantages of each approach.

3.3.1 Content Based Filtering (CBF)

Content Based Filtering is one of the most widely used methods in the recommendation system domain and one of the oldest [9, 22]. Its aim is to infer the user's interest through the item that the user interacts with [8, 9]. "Interaction" is typically established through actions, such as downloading, buying, authoring, or tagging an item [8] and user profiles are constructed based on those actions. Items are represented by a content model containing the items' features [8]. Features are typically word-based, i.e., single words, phrases, or ngrams such as in [56]. Typically, only the most descriptive features are used to model an item and users, and these features are commonly weighted [8]. Once the most discriminative features are identified, they are stored, often as a vector that contains the features and their weights [8]. The user model typically consists of the features of a user's items[8]. 3.2 shows the general structure of the content-based recommendation systems. From it one can break down the content based filtering methods into 3 main steps: *item representation, user profile learning* and *recommendation generation* [2].

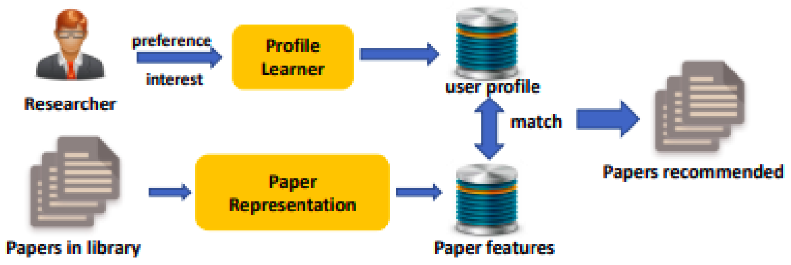


Figure 3.2: Example of a general content based recommendation system [2].

In paper recommendation systems, items are the papers in the digital library and users are typically the researchers [2]. User interest is usually inferred from contextual paper metadata characteristics such as title, abstract, citations, topics, authors and venues. For example some researchers may be interested in the work of specific authors. There are works who try to identify the most prominent authors [10] and others who rely on more than one of the aforementioned metadata to make personalised recommendations [6, 11, 14, 19]. Other works may utilize full search text methods [54], use researchers' tweets [55] or even explore co-citations [56] to identify paper similarities. According to [2], one popular representation model in CBF approaches is the TF-IDF model (term frequency-inverse document frequency) [7, 55, 57]. Other models include: Topic Models (TM), such as LDA [17, 19, 20], Key Phrase (KP) [11, 54] and Neural Network (NN) models [10, 11]. The relevance between the papers' similarities can be obtained through a similarity measure such as cosine similarity (CS) [6, 11, 15, 17, 19, 20, 55, 57].

CBF offers some advantages: it essentially provides personalised user recommendation to each individual user rather than making generic recommendations [8]. Moreover, if the paper representations are based solely on contextual metadata, then it relies on just a few key terms ((title, author, topic, etc.) and less effort is needed to compute the relatedness [2]. Additionally, it requires less up-front classification work, since user models can be created automatically [8].

On the downside, CBF requires high computing power. Each item must be analyzed for its features, user models must be built, and similarity calculations must be performed. If there are many users and many items, these calculations require significant resources [8]. Another weakness of content-based filtering is its low serendipity and overspecialization leading it to recommend items as similar as possible to the ones a user already knows [8, 22]. Another criticism of content-based filtering is that for example two documents “A” and “B” will be considered as very relevant if both have similar terms and talking about different things as well as irrelevant even if both are talking about a similar thing and using a different set of vocabularies [2].

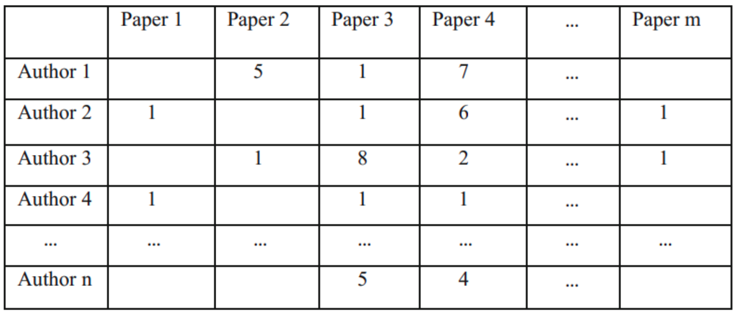
3.3.2 Collaborative Filtering (CF)

The term Collaborative Filtering was originally proposed on 1992 by Goldberg et al. [8, 58], who proposed that “Collaborative filtering simply means that people collaborate to help one another perform filtering by recording their reactions to documents they read”. In contrast to CBF, the content of the recommended paper is not considered, since recommendations depend on the ratings made by users [2]. When like-minded users are identified, items that one user rated positively are recommended to other users, and vice versa [8]. In other words, CF is the process of recommending items using the opinions of other users [2]. Social reference management websites such as Mendeley [4] or publicly available datasets such as citeulike-t [5] and citeulike-a [6] can be used to obtain information on user ratings or opinions. Another simpler approach is to ask users to fill in a questionnaire [2].

According to [2], a collaborative filtering system locates the peer utilises rating history of users to find similar users. Afterwards, the neighbourhood of those users is used to make recommendations. The neighbourhood is usually depicted in a user-

[4]`https://www.mendeley.com/`
[5]`https://github.com/js05212/citeulike-t`
[6]`https://github.com/js05212/citeulike-a`

item matrix to represent the users' ratings or comments on items. User interest is represented by this matrix. Figure 3.3 shows an author paper utility matrix used by [3]. In this matrix the items are the papers, the users are the authors and the rating factor is the number of citations the papers have received [3]. The CF system will compute the similarities based on the aforementioned matrix in order to find "neighbour users"[2]. A structure of such a system is presented on figure 3.4.



	Paper 1	Paper 2	Paper 3	Paper 4	...	Paper m
Author 1		5	1	7	...	
Author 2	1		1	6	...	1
Author 3		1	8	2	...	1
Author 4	1		1	1	...	
...	...	...	...	...	...	...
Author n			5	4	...	

Figure 3.3: Example of an author paper utility matrix [3].

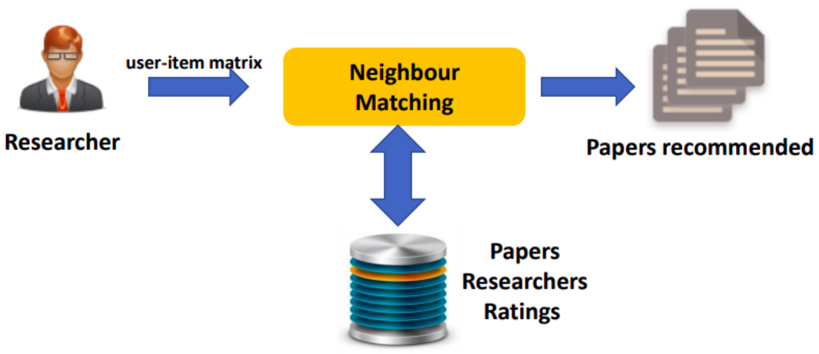


Figure 3.4: Example of a general collaborative filtering recommendation system [2].

CF is consisted of two methods based on the approach followed [2]:

1. **User-based**: User based approaches focus their recommendations based on the users' interests. For example [3] authors build the profiles of authors and suggests paper recommendations based on "neighbour authors'" paper citations and references. [12] learn semantics between titles and abstracts of papers on word- and sentence-level to represent user preferences. According to [2], in the user-based systems, users are divided into the several groups, the users in the same group share the same or similar interests on some items.

in our example [3], aggregated author groups were created to generate recommendations. Another work [20] defines the researchers' relevance model based on the topics present in their profile and in the profiles of similar researchers and organises the researchers into communities with similar interests to make recommendations.

2. **Item-based**: Item based approaches center their recommendations around the relationships of items, instead of the users themselves [2]. For example in [21] the authors accept as input a paper of interest (POI) to the user and generate recommendations based on its citations, co-citations and references.

Collaborative filtering process, both user and item based, can be summarised in 3 steps [3, 6, 21, 57]:

1. Identify the neighbours of the target user/item
2. Use the neighbours to rank the items/users
3. Recommend top N items to the user

To measure the similarity of neighbours, 2 works utilised the Jaccard similarity coefficient [6, 21]. Other works included topic models such as Alternating Least Squares (ALS) algorithm [3], long term short memory networks (LSTMs) [12], k-NN (k nearest neighbours) [17] and custom methods [14]. Ranking methods such as BM25 or variations of it (BM25-T, BM25-A, BM25-K) [57] were also utilised.

CF offers its own advantages. Since collaborative filtering does not depend on content, the error-prone processing of items is eliminated. Furthermore, ratings are based on humans. Therefore, real quality assessments are taken into account [8]. Finally, CF recommendations are serendipitous, users are offered more generic recommendations, since they stem from user rather than item similarity [2, 8].

However, CF has also some major disadvantages. A quite common one is the cold start problem [3–5, 7, 17, 19, 20, 55, 59]: finding relevant information for newly added items or users proves highly challenging since there is not enough information. If a new user rates few or no items, no like-minded users can be found and, therefore, no recommendations will be provided. Likewise, if an item is new and has not rated yet by at least one user, it cannot be recommended. In a new community, there are usually no rated items, resulting in zero recommendations for the users. As a result the incentive for a user to rate an item is low [8]. Another main disadvantage of CF methods is the data sparsity problem [3–8, 12, 14, 19–21, 59]: there is a different ratio of users and items. In the domain of recommendation systems, there is an abundance of papers but typically few users. Even fewer users have rated the same papers. As a result, finding like-minded users is often not possible [8]. In addition, many papers may not be rated at all and, consequently, cannot be recommended. Just to capture this difference, [60] compared the implicit user ratings on Mendeley [7] (research papers) and Netflix [8] (movies), and found that sparsity on Netflix was three orders of magnitude lower than on Mendeley. Due to depending on user input, there are some additional problems as well:

- **Gray sheep**: This refers to the people whose opinion remains gray, they do not agree or disagree with any group of people and thus do not benefit from CF [44, 59].
- **Black sheep**: that group is the exact opposite of gray sheep. Their unique taste makes recommendations impossible. However, Black sheep are an acceptable failure since even manual recommendation may face similar problems [44, 59].
- **Shilling attack**: In cases where anyone can provide recommendations, these can be biased. Users may promote their own work while give negative recommendations to other works [44, 59].
- **Privacy issues**: People may not want to make their habits publicly available [44, 59].
- **Manipulation problems**: ratings maight be manipulated to promote specific products so they are recommended more often [8].

[7]`https://www.mendeley.com/`
[8]`https://www.netflix.com/`

- **Scalability problems**: as the data grows CF models are less efficient [19].

To overcome the above problems, researchers utilise other recommendation techniques such as graph-based and hybrid [4, 7, 14, 17, 19, 20].

3.3.3 Graph-based approaches

As their name suggests, graph based approaches rely on the construction of a graph to make recommendations. Graph structures are typically heterogeneous information networks (HINs) [5, 19], knowledge graphs (KGs) [4, 18], citation network graphs [13, 21], social network graphs [14] or can be simpler, such as author-paper [7] or author-topic [20] graphs. Figure 3.5 illustrates a simple graph approach, figure 3.6 a knowledge graph (KG) while figure 3.7 a specific type of a knowledge graph (HIN).

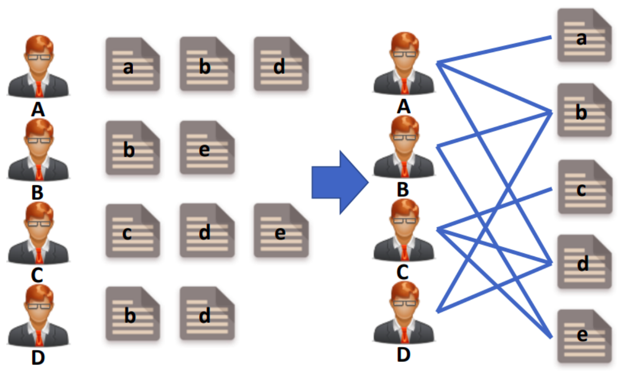


Figure 3.5: Example of a simple graph approach [2].

The edges of the graph can be weighted and represent the relevance degree between the graph's objects [4, 5, 7, 13, 19–21]. For the observed approaches that mention a weighting scheme, the TF-IDF weighting scheme[3, 5, 7, 18], semantic keyword weighting (SKW) [14] and BM25 or variations of it (BM25-T, BM25-A, BM25-K) [57] were used. As for algorithms or methods used on these graphs to calculate similarity, approaches used random walk (with restart) (RW(R)) [5, 14], topic models such as LDA [20], sentence bidirectional encoder representations from transformers (SBERT) embeddings [19], metapaths (see section 2.4) [5], knowledge graphs [4, 7, 18], LocRank a variation of PaperRank [13] and Jaccard Coefficient

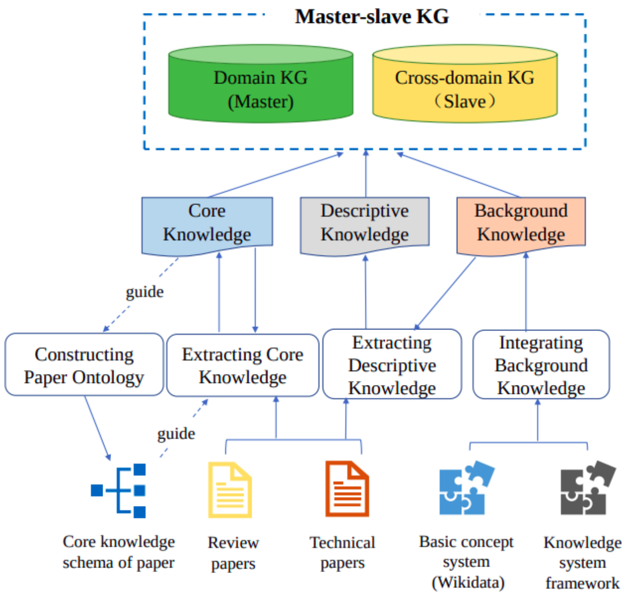

Master-slave KG
Domain KG (Master)
Cross-domain KG (Slave)
Core Knowledge
Descriptive Knowledge
Background Knowledge
guide
Constructing Paper Ontology
Extracting Core Knowledge
Extracting Descriptive Knowledge
Integrating Background Knowledge
guide
Core knowledge schema of paper
Review papers
Technical papers
Basic concept system (Wikidata)
Knowledge system framework

Figure 3.6: Example of a master - slave knowledge graph [4].

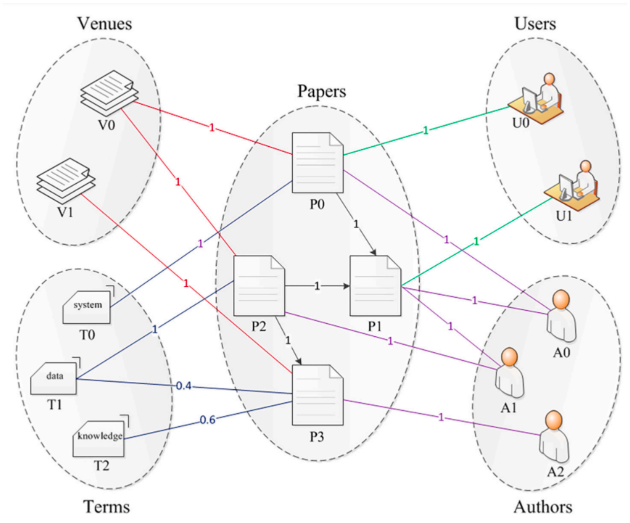


Figure 3.7: Example of a heterogeneous information network graph approach [5].

[21].

One major advantage of graph-based methods is that they can capture complex relationships between papers and researchers, such as co-authorship and citation relationships [5, 13], which may not be captured by other methods. This can result in more accurate recommendations [2]. Moreover, they can overcome problems of the previous approaches such as scalability: graphs can handle large-scale data sets and are able to scale well with increasing data size [4] and the cold start problem [4]. Finally, once the graph has been built, recommendations can be really fast [13].

On the other hand, graph-based methods may require more computing resources and time to build and maintain the graph structure, compared to other methods such as collaborative filtering or content-based filtering [5]. They may not be as effective in situations where there is limited data available, or when the data is highly noisy or incomplete [5]. Finally, it can be difficult to incorporate additional features or constraints into the recommendation process, as the recommendation is typically based on the relationships in the graph rather than explicit features of the papers or researchers [4, 5, 13, 18].

3.3.4 Hybrid

Hybrid approaches combine 2 or more of the previous approaches to generate more accurate recommendations and also overcome previous problems. One downside is that since they combine more than one model, implementations can be complex and require significant resources [7]. Figure 3.8 illustrates the work of [6] which proposes a hybrid framework consisted of both content based and collaborative filtering methods. On Figure 3.9 the work of [7] is depicted. It combines content based and graph paper recommendation (CGPRec) methods by also utilising a convolution neural network (CCN) to emulate users' preferences directly from paper content.

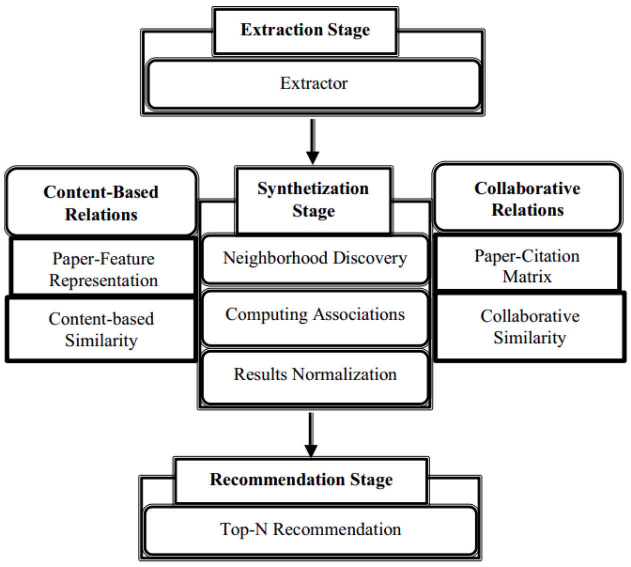
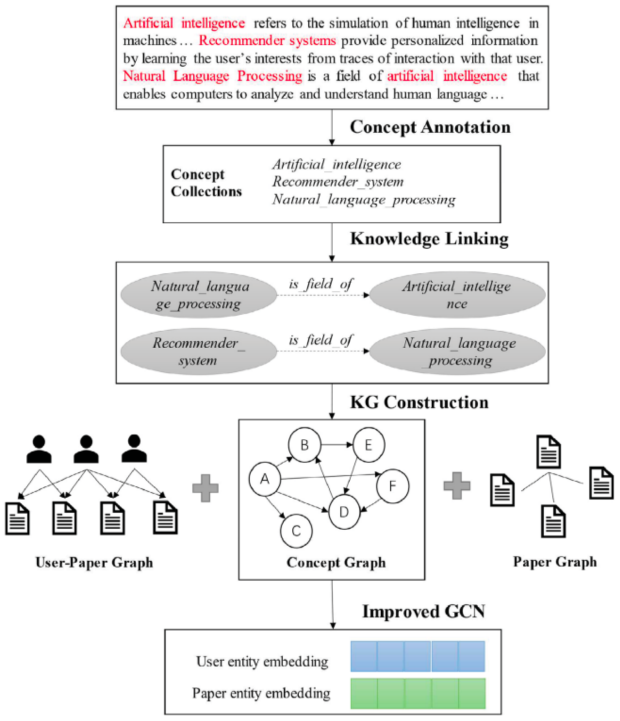


Figure 3.8: Example of a hybrid method utilising both CBF and CF [6].

Figure 3.9: Example of the CGPRec hybrid method [7].


Artificial intelligence refers to the simulation of human intelligence in machines… Recommender systems provide personalized information by learning the user's interests from traces of interaction with that user. Natural Language Processing is a field of artificial intelligence that enables computers to analyze and understand human language…
Concept Annotation
Concept Collections
Artificial_intelligence
Recommender_system
Natural_language_processing
Knowledge Linking
Natural_language_processing
is_field_of
Artificial_intelligence
Recommender_system
is_field_of
Natural_language_processing
KG Construction
User-Paper Graph
Concept Graph
Paper Graph
Improved GCN
User entity embedding
Paper entity embedding

3.4 Method Overview

Below we provide an overview on the scientific directions associated with the categories presented on section 3.2.2, while also focusing on the methods used. Table 3.4 illustrates the distribution of papers based on the method they used. Since papers can combine more than one method, the percentages do not add up to 100.

UP	Popularity	KP	Embedding	TM	KG	Graph	Path	RW(R)	AML	Crawling	CS
55%	25%	10%	30%	15%	25%	45%	10%	15%	40%	20%	30%

Table 3.4: Distribution of papers by method.

- **Personalised**: 13 works claim to provide personalised recommendations [3–5, 7, 10, 12–14, 17, 19, 20, 54, 55].
- **Input**: 9 works utilised some form of input derived only from the given papers [6, 10, 11, 13, 15, 18, 21, 54, 56], such as its title, abstract, authors, citations, references etc, 3 works utilised information derived only from users [5, 7, 12] such as user interactions on papers, 6 works utilised input from both the users and the papers [3, 14, 17, 19, 20, 55], 1 work utilised only keyword input [57],

by breaking down the researchers' subject of interest into keywords and 1 work utilised keywords along with some other form of input [4].

- **Title**: 12 works utilised the papers' title to generate recommendations [4, 6, 7, 10, 12, 14, 17, 18, 54–57].
- **Abstract**: 9 works utilised data from the papers' abstract to generate recommendations [4, 6, 7, 12, 14, 17, 18, 54, 57].
- **Keyword**: 7 works utilised the papers' author-provided keywords to generate recommendations [4, 7, 14, 17, 54, 56, 57].
- **Text**: 5 works utilised data from the papers' text to generate recommendations [3, 5, 15, 18, 19].
- **Citation**: 10 works utilised data from the papers' citations, co-citations or references to generate recommendations [5–7, 10, 11, 13, 18, 19, 21, 56].
- **Author**: 5 works utilised papers' author information to generate recommendations [5, 10, 19, 54, 56].
- **Topic**: 3 works utilised papers' topic information to generate recommendations [5, 19, 20].
- **Venue**: 3 works utilised papers' venue information to generate recommendations [5, 19, 54].
- **Historic Interaction**: 11 works utilise some form of historic user interactions which can be derived from user actions: such as user tweets [55], features of clicked papers [7], rated papers [20], unspecified ones [5], papers of interest [17], users' article interests, area of expertise, publications [14], implicit interaction data such as user web browsing logs or purchasing records [3], user preferences by capturing the sequential sentence patterns in titles and abstracts [12] or can be derived from papers: such as author interest based on author's citations [10], authored papers [19], the finding of more references for a current work [13].

- **User Profile**: the aforementioned 11 works [3, 5, 7, 10, 12–14, 17, 19, 20, 55] constructed user profiles based on the above user interactions.
- **Popularity**: 5 works utilise some popularity indication such as author popularity [10], unspecified popularity indication [54], citation count popularity [18, 57] and discipline popularity [17].
- **Key Phrase**: 2 works utilise key phrases: [54] experiments with KP extraction while [11] implement an automatic key phrase extraction based on standard key phrase extraction systems.
- **Embedding**: 6 works utilise some form of embedding: [20] uses LDA to define the researchers' profiles and to find neighborhoods with similar topical interests, [19] uses SBERT to learn authors' embeddings, [15] uses Doc2Vec, [4] Doc2Vec of word pairs and [7, 12] use Word2vec.
- **Topic model**: 3 works utilised topic models: [19, 20] use LDA and [3] used Alternating Least Squares (ALS) model.
- **Knowledge Graph**: 5 works incorporate knowledge graphs. One uses a predefined one, the DDBpedia, [18], [4] builds two knowledge graphs, one in-domain and one cross-domain. The graphs are user-constructed and include representative papers for the different concepts.[5, 19] use a HIN, a specific form of knowledge graph. Finally, [7] build a KG with the linked open data (LOD) and then merge this knowledge graph with the user-paper graph.
- **Graph**: 9 works utilised graphs: such as paper-author [7], author-topic [20], user-article-keyword [14], author-paper-venue-label-topic [19], author-paper-venue-keyword [5], citation [13, 21], predefined [18], in-domain and cross-domain graphs [4].
- **Metapath**: only 1 work incorporates the usage of metapaths of a max length between users and papers [5].
- **Random Walk (with Restart)**: 3 walks utilise the random walk (with restart) algorithm: [5] uses RWR on metapaths, [13] uses RWR on subgraphs

and [14] utilised RWR-based filtering algorithm.

- **Advanced Machine Learning**: 8 works utilised some form of AML techniques: [10] used a neural citation network (NCN) based on the encoder-decoder architecture, some form of simple neural network [12], multi-layer perceptrons [7], gradient ascent or descent [4], [11] present a form of neural network ranking, called NNRank, which estimates the probability of a document running another, [19] proposed a bipartite network embedding model, [57] utilised multiple learning 2 rank (L2R) AML models such as Multiple Additive Regression Trees (MART), a.k.a. Gradient boosted regression tree), LambdaMART, Random Forests (RF), ListNet and RankNet, AdaRank, RankBoost, and Coordinate Ascent. Finally [18] used LambdaMART.

- **Crawling**: 4 works used some form of crawling to collect their data: such as papers from ACM, IEEE and EI [5], [57] extracted data from papers using the Scopus API [9], [55] crawled arXiv [10] paper repository to extract data while while [56] retrieved over 1200 papers from CiteSeer (current CiteSeerX [11]).

- **Cosine Similarity**: 6 works utilised cosine similarity [4, 15, 17, 19, 54, 55], [55] used IA-Select in addition to CS, 2 works utilised Jaccard similarity [6, 21] while 1 work used Pearson's similarity [56].

3.5 Datasets

In this section we present the datasets used by the reviewed papers. We do not discuss datasets that are not adequately described (i.e., only the data sources are mentioned but no further remarks are made regarding their size or composition [15]). Many approaches utilise their own modified version of a public dataset. Additionally, some datasets are available only few years after publication.

Out of the 20 works, one work [15] creates its dataset by crawling the text of freely available online papers and another work [54] is a recommendation-as-a-service

[9]`https://www.scopus.com/home.uri`
[10]`https://arxiv.org/`
[11]`https://citeseerx.ist.psu.edu/`

by indexing data from Sowiport. Therefore, these works are not considered in the dataset overview.

An overview of the 18 datasets of the considered methods is presented on table 3.5. Less than half of them (39%) are publicly available.

Work	Dataset(s)	Public?
[10]	Refceer	✓
[6]	SPRD	✓
[57]	Scopus	
[5]	AMiner + DBLP	
[11]	DBLP + PUBMED	
[56]	CiteSeer	
[20]	AMiner[12]	
[19]	DBLP + DBLP-Citation-network v11	✓
[4]	KGs	
[55]	ACM-Citation-network v8	✓
[13]	MAG + CiteSeerX + DBLP	
[3]	MAG[12]	
[18]	MAG + CiteSeerX	
[17]	Mendeley	
[12]	Citeulike-a + PRSDataset	✓
[21]	SPRD	✓
[14]	Scholarmate	
[7]	Citeulike-a	✓

Table 3.5: Overview of datasets utilised in reviewed work along with their public availability.

3.5.1 CiteSeerX-based datasets

CiteSeerX [61, 62] is a digital library focused on metadata and full-texts of open access [13]. It is the overhauled form of the former digital library CiteSeer. [56] retrieved over 1200 papers from CiteSeer by using different keywords

A CiteSeerX based dataset called Refseer was chosen by [10]. It is only available through a public github repository [14]. The dataset is consisted of 3 entities: *papers*, *citations* and *citationContexts*.

[12]Although the original sources are public, these specific datasets have been produced by pre-processing the original resources with the pre-processed/final version not being public.

[13]`https://citeseerx.ist.psu.edu/`

[14]`https://github.com/chbrown/refseer`

3.5.2 SPRD-based datasets

The Scholarly Paper Recommendation Dataset (SPRD), constructed by SugiYama and Kan [63, 64] is made publicly available [15]. It consists 2 datasets, one of which is used by [6, 21]. It contains 100,531 papers published in various kinds of proceedings '00 to '10 constructed from ACM Digital Library and it also includes 50 researchers' interests. Relevance assessments of papers relevant to their interests are also included.

3.5.3 Sowiport-based datasets

Sowiport was an open digital library containing information on publications from the social sciences and adjacent fields [65, 66]. The dataset linked papers by their attributes such as authors, publishers, keywords, journals, subjects and citation information. Via author names, keywords and venue titles the network could be traversed by triggering them to start a new search [65]. Sowiport was the first partner of Mr. DLib's recommendation-as-a-service system [54].

3.5.4 Scopus-based datasets

Scopus is a semi-open digital library containing metadata on authors, papers and affiliations in different scientific areas [16]. They offer an API to query for data. One work [57] utilised the api to collect titles, abstracts, year of publications, keywords and list of references from a total of 58,734 papers, all published in English, and the publication dates vary from 1970 to 2018.

3.5.5 DBLP-based datasets

The DBLP computer science bibliography (DBLP) provides open bibliographic information on major computer science journals and proceedings [67]. Publicly avail-

[15] `https://www.db.soc.i.kyoto-u.ac.jp/~sugiyama/SchPaperRecData.html`
[16] `https://www.scopus.com/home.uri`

able short-time stored daily and longer-time stored monthly data dumps are provided [17]. The data is also updated over time.

3.5.6 ACM-based datasets

The ACM Digital Library (ACM) is a semi-open digital library offering information on scientific authors, papers, citations and venues from the area of computer science [18]. They offer an API to query for information.

3.5.7 AMiner-based datasets

ArnetMiner (AMiner) [41] is an open academic search system modelling the academic network consisting of authors, papers and venues from all areas [19]. They provide an API to query for information. [5] utilises data from both DBLP as well as a DBLP AMiner dataset, but does not specify which one. It is used by [19] along with data from ACM.

The ACM-Citation-network v8 dataset [20] builds upon ACM. It contains 3,272,991 papers and 8,466,859 citation relationships (topics, authors, venues, published year, citation count and references).

The DBLP-Citation-network v11 dataset [21] builds upon DBLP . It contains 4,107,340 papers and 36,624,464 citation relationships (topics, authors, venues, published year, citation count and references). It is used by [19].

3.5.8 PUBMED-based datasets

PubMed is a free resource supporting the search and retrieval of biomedical and life sciences literature with the aim of improving health–both globally and personally [22].

[17]`https://dblp.uni-trier.de/xml/`
[18]`https://dl.acm.org/`
[19]`https://www.aminer.org/citation`
[20]`https://lfs.aminer.org/lab-datasets/citation/citation-acm-v8.txt.tgz`
[21]`https://lfs.aminer.cn/misc/dblp.v11.zip`
[22]`https://pubmed.ncbi.nlm.nih.gov/`

The PubMed database contains more than 34 million citations and abstracts of biomedical literature. [11] used PUBMED along with DBLP to extract citation data.

3.5.9 MAG-based datasets

The Microsoft Academic Graph (MAG)[23] [37] was an open source heterogeneous graph containing scientific publication records, citation relationships between those publications, as well as authors, institutions, journals, conferences, and fields of study. [3] used data only from MAG, [18] uses data from both MAG and CiteSeerX, while [13] created their dataset by utilising data from MAG, CiteSeerX and DBLP.

3.5.10 Mendeley-based datasets

Mendeley is a semi-open digital library containing metadata on authors, papers and affiliations in different scientific areas [24]. They offer an API to query for data [25]. One work [57] utilised user profile metadata, document metadata (e.g. title and abstract), and user libraries from the Mendeley reference manager [26].

3.5.11 CiteULike-based datasets

CiteULike [68] was a social bookmarking site for scientific papers. It contained papers and their metadata. Users were able to include priorities, tags or comments for papers on their reading list. There were daily data dumps available from which datasets could be constructed. Two variations of public CiteULike datasets are available:

1. *Citeulike-a* [69] [27] contains 5,551 users, 16,980 papers with titles and abstracts from 2004 to 2006 and their 204,986 interactions between users and papers.

[23]`https://www.microsoft.com/en-us/research/project/microsoft-academic-graph/`
[24]`https://www.mendeley.com/`
[25]`https://dev.mendeley.com/`
[26]`https://www.mendeley.com/reference-management/reference-manager`
[27]`https://github.com/js05212/citeulike-a`

Papers are represented by their title and abstract. It is used by [7].

2. *Citeulike-t* [69] [28] contains 7,947 users, 25,975 papers and 134,860 user-paper interactions. Papers are represented by their pre-processed title and abstract.

3.5.12 ScholarMate-based datasets

ScholarMate [29] is a semi-open a research social media-network eco-system that allows researchers to create profiles, build connections with research collaborators, upload and share knowledge (e.g. publications, patents, projects, research grants, researchers, and research organizations). One work [14] builds its KG, around ScholarMate.

3.5.13 Other

The following dataset has no common underlying data source:

PRSDataset [30] contains 2,453 users, 21,940 items and 35,969 pairs of users and items. This dataset contains user-item interactions. [12] uses PRSDataset along with Citeulike-a.

3.6 Evaluation methodologies

Evaluation methodologies are critical to assess the effectiveness of recommendation approaches. According to [8], appropriate evaluation methods, a sufficient number of study participants, and a comparison of the approach against one or more state-of-the-art approaches, are the key components for a thorough evaluation.

In this section, we follow the notion of [8] and classify the evaluation approaches into 3 classes: user studies, offline evaluations and online evaluations. Papers that do not explicitly mention an evaluation methodology, have performed it in a previous work or do not use evaluation metrics are not considered. Therefore, we excluded the papers [54, 56] from the above classification. In [54] the authors intend their

[28]`https://github.com/js05212/citeulike-t`
[29]`https://www.scholarmate.com/`
[30]`https://sites.google.com/site/tinhuynhuit/dataset`

recommendation as-a-service to tool to be used by other researchers to evaluate their work and have evaluated the tool in previous works. In [56], the authors create a visual interpretation of their recommendation results and have domain experts rank them. However, there is no evaluation metric reported, to support their results.

Overall, 89% of the works performed some form of offline evaluation while the rest 11% conducted user studies to evaluate their recommendation systems. Table 3.6 illustrates the type of evaluation per reviewed work.

Evaluation Type	**Work**
offline	[3–7, 10–13, 15, 17–21, 57]
user study (lab-based)	[14, 55]

Table 3.6: Type of evaluations per reviewed work.

3.6.1 User Studies

A user study in a paper recommendation system is a research method that involves collecting data from users of the system in order to assess its effectiveness and usability [8]. This can be done through various methods, such as surveys, interviews, or observations of users interacting with the system. The goal of a user study is to understand how users perceive and interact with the recommendation system, and to identify any issues or areas for improvement [8]. They can be distinguished between "lab" and "real-world" [8]. In a lab user study users are aware that they are participants, which may affect evaluation results to some extent (ex. introduce rating bias). On the other hand, in a real-world study user interaction is better depicted, since users provide ratings for their own benefit. For example the recommender system may improve recommendations based on user ratings [8].

Often, user studies are considered the optimal evaluation method [8]. However, results often depend on the questions asked. Moreover, a large number of participants is required (typically a few dozens), to receive statistically significant results. That also depends on the number of approaches being evaluated, on the number of recommendations being displayed, and on the variations in the results [8].

Out of the 2 observed works that utilised user studies:

The authors in [55] conducted an online survey and asked 22 users to rate the provided recommendation results as “interesting” or “not interesting”. They used a combination of 12 strategies combining the text mining method (TF-IDF, CF-IDF, HCF-IDF), the profiling source (own papers, twitter) and the ranking method (CS, IA-Select), retrieving the 5 top items for each strategy. Therefore, each user had to rate 60 papers. The Serendipity Score (SRDP) was utilised to evaluate the serendipity of recommendations as well as several statistical tests (ANOVA test, Muchly’s test, Mendoza test) to verify significant differences between strategies. Authors concluded that tweets do not improve the serendipity of recommendations, but IA-Select algorithm does [55].

In [14], the authors perform a survey in two phases. On phase one, they considered 100 active users, randomly selected, from the Scholarmate platform. Each user had at least three publications in the platform. The users were asked about the quality of the recommendation results obtained by the integrated recommendation method and by the two alternatives. 76 users responded. For the second round, recommendation lists were recomputed and again presented to users 1 month later, excluding articles from the first stage of the survey. In total, 45 users responded to the second survey. The average rating (AR) score and the normalised discounted cumulative gain (NDCG) were selected as performance metrics for the top 5 and 10 recommendations. Results showed promising performance in terms of accuracy metrics [14]. Authors also mentioned some limitations, such as the limited user participation in the study and the semantic ambiguity in keyword matching used in their work [14].

3.6.2 Online evaluations

Online evaluations measure the acceptance rates of recommendations in real-world recommender systems [8]. Acceptance rates are typically measured by click-through rates (CTR), i.e., the ratio of clicked recommendations to displayed recommendations. Other metrics may include the ratio of downloaded or bought items to the number of items displayed. Acceptance rate is typically interpreted as an implicit measure for user satisfaction on the assumption that when a user clicks, downloads, or buys a recommended item, the user liked the recommendation [8]. That is not always the case, since a user can buy an item and then rate it negatively [8]. However, metrics such as CTR can be explicit measures of effectiveness, in cases when the operator receives money, e.g., for clicks on recommendations [8].

However, online evaluations come with disadvantages. CTR and relevance are not always correlated and should be used cautiously [8]. Additionally, online evaluations require significantly more time, resources and are more costly than offline evaluations [8]. They also rely on having access on real-world recommender systems [8].

We found no work that utilised an online evaluation in our survey.

3.6.3 Offline evaluations

Offline evaluations typically measure the accuracy of a recommender system based on a ground truth [8]. Common evaluation metrics include precision, mean average precision (MAP), recall, F-measure, mean reciprocal rank (MRR), normalized discounted cumulative gain (nDCG), mean absolute error, and root mean square error [3–7, 10–13, 15, 17–21, 56, 57].

Criticism has been raised on the assumption that offline evaluations could measure an algorithm's effectiveness on real users [8]. More reasearch has shown that results from offline evaluations do not necessarily correlate with results from user studies or online evaluations [70]. This means that approaches that are effective in offline evaluations are not necessarily effective in real-world recommender systems [8]. Some reasearchers have even argued that offline evaluations are probably are not a suitable form of of evaluation of scientific research paper recommendation systems [70]. Despite the criticism, offline evaluations remain the predominant evaluation method in the scientific paper recommendation community [8, 16, 70].

Out of the 16 observed works that utilised offline evaluations:

In [10], the authors evaluated their work against 4 baselines: BM25, a Citation Translation Model (CTM), an encoder-decoder time delay neural network to recurring neural network (TDNN-to-RNN) and an encoder-decoder recurring neural network to recurring neural network (RNN-to-RNN), using the metrics of recall, MAP, MRR and NDCG on the test set. Results showed that the authors' approach NCN outperformed all baselines on every metric making the approach "a flexible architecture capable of incorporating author metadata and highlight a promising new direction for context-aware citation recommendation" [10].

In [6], the authors evaluated their work against 2 baselines: one CF baseline presented in a previous work and a hybrid approach presented in another work, by utilising the metrics of precision, recall, F1-score, MRR and MAP. Their methodology has seen a substantial improvement over other benchmark approaches in evaluating both the overall performance and the ability to return applicable and usable publications [6]. One limitation mentioned by the authors is that recommendations are not divergent and another one is that the approach does not include relevance/preference score for the candidate papers [6]. Therefore, there is a lot room for improvement [6].

In [57], the authors built 5 realistic target queries by extracting the author-specified keywords from papers among the dataset and utilised the metrics of precision and recall to evaluate their L2R models against baselines. The baselines were: content based methods such as s BM25, BM25-T, BM25-A and BM25-K, author-specified keywords based retrieval (AKR), weighted-author-specified keywords based retrieval (wAKR) as well as adaptions of two recommendation models from other authors which combine content-based information with information from a citation network by means of a collaborative filtering technique [57]. Results showed that, although there is no clear winner among the models, the techniques produced significant improvements over the baselines across all requirements [57].

In [5], the authors defined recommendations liked by users as True Positive (TP), and others as False Positive (FP). They also defined papers that were not recommended but liked by users as False Negative (FN), and others as True Negative (TN) [5]. As evaluation metrics, they chose precision and recall and evaluated their work against 4 baselines: a co-citation (CC) method, a multi stage citation network

(MSCN) method, a citation article recommendation (CAR) method and a metapath method [5]. Results showed that the authors' method outperformed other baseline methods in terms of precision and recall with some limitations [5]. Firstly, in order to discover user preference patterns and provide new recommendations, the model relies heavily on a user's historical preferences. Therefore recommendation performance may suffer when analyzing data from new users or those with few activities on the web [5]. Secondly, a user's recent preferences and previous preferences are considered to have the same impacts on their current interests [5].

In [11], the authors compared the 3 variants of their approach against 2 baselines: ClusCite (a heterogeneous graph of terms, authors and venues in order to find related documents) and BM25, by utilising the metrics of MRR and F1 score at 20 (F@20). Authors reported that their method obtained state of the art results on two citation recommendation datasets even without the use of metadata available [11].

In [20], the authors evaluated their work against 2 baselines: Relevance-based language modeling to CF and PageRank-weighted CF model, by performing a 5-fold cross-validation on the dataset and by measuring the average recall of their system. Their preliminary results show that their approach achieved better average recall values than the baselines and that the model does not suffer from the cold start problem [20].

In [19], the authors evaluated the performance of the 5 versions of the proposed model against 5 baseline methods: a LDA topic model, a machine learning-based dynamic treatment regimen (ML-DTR) model, a NNRank model, a generative adversarial network - human behavioural network representation (GAN-HBNR) model and a bibliographic network representation (BNR) model. The evaluation metrics used were recall, MAP, and MRR [19]. Results showed that the proposed model outperformed state-of-the art citation recommendation baseline models [19].

In [4], the authors conducted experiments by constructing their model by selecting a number of papers from 3 academic resource databases, Google Scholar, EI, and IEEE. The keywords they used for selecting the papers were: information retrieval and machine learning [4]. The authors evaluated the recommendations by utilising the metrics of TopN Effective Rate (TopN), FindN Discovery Rate (FindN) and MRR. The experimental results showed that their method was effective and has potential [4].

In [13], the authors evaluated their approach against 2 baselines: PageRank and a collaborative filtering (CF) method by using recall and and the mean execution runtime as performance metrics. Experiments showed the proposed method is up to 15x faster than PaperRank and up to 6x faster than CF, while maintaining similar performance as PageRank and having better performance than the CF method [13].

In [3], the authors extracted datasets for two research fields, namely machine learning and recommender systems from the MAG dataset (check section 3.5.9 for more details). On the TopN suggestions, they utilised the recall metric (number of articles the author cited in TopN divided by the total number of articles the author cited) to measure the system's effectiveness [3]. Based on the results, authors concluded the system can successfully recommend the Top-N papers in a research field based on the aggregated authors' profiles in the same field, thus providing personalised recommendations [3].

In [18], the authors evaluated their approach against 2 baselines: weighted tf-idf score on papers' abstracts, and Lucene's More Like This [31] score on terms in the papers' abstract and entities. They have also performed *random split per year* and *evaluating on the next year* methods to measure the quality of the global citations recommendations produced [18]. The evaluation metric used was NDCG at 10 (NDCG@10) [18]. Results showed that their approach outperformed the other 2 text-based baselines [18].

In [15], the authors evaluated their approach against a baseline model of content-based technique the authors implemented themselves. The evaluation metric used was NDCG at 10 (NDCG@10) [15]. Results showed that their proposed method performs well and is able to recommend related documents to most of the users [15].

In [17], the authors divided their users, according to whether they had library articles prior to a distinct time boundary, to warm and cold users. Afterwards, the authors took all data from user libraries and splited them into testing and training

[31] `https://lucene.apache.org/`

sets across the same time boundary. Their recommendation methods are assessed by testing how well recommendations generated from the training set are able to predict what a user is going to add to their library in the testing set [17]. Authors utilised the harmonic mean metric F1 to evaluate the overall results [17]. To the authors' surprise, recommendations based on a cold users' recent activity do not perform as well, especially when based on the most recently added article [17]. On the other hand, recommendations based on recent activity for warm users score well, indicting that users are following a path looking for similar items [17].

In [12], the authors compared variants of their model against 3 baseline methods: Bayesian Personalized Ranking (BPR) method, a Collaborative Deep Learning (CDL), and a Convolutional Matrix Factorization (ConvMF) method. The chosen evaluation metrics were Precision at n (Pre@n), MRR and NDCG [12]. Results showed that the proposed model outperformed consistently and significantly the other models in all ranking metrics [12].

In [21], the authors evaluated the quality of the proposed approach against 3 CF baseline methods that generate recommendations based on citation-relations. The authors performed 5 cross validation on the dataset and chose the evaluation metrics of precision, recall, F1 score, MRR and MAP [21]. Results showed that the proposed approach outperformed all the baseline approaches in terms of the chosen evaluation metrics [21].

In [7], the authors evaluated their work against 4 baselines: a Bayesian Personalized Ranking Matrix Factorization (BPRMF) method, a Collaborative Metric Learning (CML) method, a Knowledge Graph Attention (KGA) method and a Deep Knowledge-Aware Network (DKN) method. Authors chose F1 score and Area Under the ROC curve (AUC) as their evaluation metrics [7]. Results showed that the model was significantly better than the baselines in F1-score and AUC indicators, however the authors expressed some concerns that incomplete or inaccurate KGs could introduce errors into the recommendation process, and large data volume of KGs caused complicated calculation and further resources [7].

3.7 Summary

In this section we provide a summary of our review and we discuss some open problems / challenges some of which have been outlined by the previous studies [8, 16], as well as providing our view. We summarise our key findings in the following paragraphs.

The hybrid was the predominant prominent recommendation approach (PRA) in our study. From the 20 reviewed recommendation approaches, 8 (40%) used hybrid approaches, 6 (30%) utilised CBF approaches, 4 used graph-based approaches while only 2 went with CF approaches. Regarding the methods used (see section 3.4 and table 3.4 for more details), 55% of the works constructed a user profile or utilised profile information while graph models seemed to be the most popular since 45% of the works utilised a graph model at some point. Almost all works evaluated their approach except for 1 work, who did not provide sufficient evaluation [56]. In almost all the cases authors compared their work against state of the art baselines (see section 3.6). However, the vast majority of the works that provided a sufficient evaluation focused on offline evaluations (see section 3.6.3 for more details). As we have already discussed, offline metrics do not always correlate with user satisfaction. Therefore, we believe that offline evaluations should indeed be used as a means to identify a number of promising recommendation approaches but researchers should not stop there. After such approaches have been found, other more "real-world" evaluation methods, such as online evaluations or user studies, should be applied to get a more complete picture of the system's performance.

Another problem that we observed which was also raised by [8, 16] is the reproducibility of the experiments. Specifically, [8] states that "the reproducibility of experimental results is the fundamental assumption in science, and the cornerstone that allows drawing meaningful conclusions about the generalizability of ideas". Although in this study we did not try to reproduce the experiments of the works, almost all of them used modified datasets which were not made publicly available. Moreover, out of the 20 reviewed works, only 3 [10, 11, 54] made their implementation publicly available for others to use. Therefore, it is very hard to reproduce results for most of the works in this survey.

According to [8] there is a generic assumption that the goal for the operator of a recommender system is to satisfy the needs of the users. On the contrary, operators may also want to keep down resource costs and as a result, an effective system for an operator may be the one that can be operated at low costs [8]. Operators may also prefer to generate a profit from the recommendations to users even if user satisfaction is not optimal [8]. ”The operator’s perspective” as mentioned by [8] has been widely ignored in our survey.

Another challenge that is quite similar to the previous one is the systems’ scalability and complexity. As the most recent systems tend to use more complex models, they also tend to require more resources. Very few approaches reported computational complexity or costs. Almost no approach considered scalabilty issues. For example apart from [3] who considered the system’s scalability, no work mentioned the capacity of the implemented system or if the performance would be affected for many users.

User characteristics such as registration status of users are already mentioned by [8, 16] as a factor which is disregarded in evaluations. The targeted audience of a paper recommendation system should influence its suggestions. In [8], the authors highlighted different needs of junior researchers and for senior researchers. In our study, target audiences in general were rarely defined and a recommendation scenario was mostly not described.

Last but not least, like [8], we found it impossible to determine the most effective recommendation among the approaches. If we were asked which recommendation approach to apply in practice or to use as baseline, there is no definite answer. This problem mainly relates to poor experimental design and information scarcity in the domain. It can also be because it is very hard to measure what a user considers relevant.

To conclude, there are many open challenges which current approaches did not manage to overcome. ”They define the requirements for future works in the area of paper recommendation systems” [16].

Chapter 4

Paper Recommender Implementation

In the previous chapter, we presented the short literature review survey, we have conducted, about the problem of scientific paper recommendation. Moreover, we discussed about the various paper recommendation approaches, the pros and cons of each one as well as related open challenges. In this chapter, we address the problem of finding related scientific papers of interest. In particular, we study the problem of suggesting a set of papers to researchers based on their interests. We define the researchers' interest, as a set of one or more papers the researchers have already found interesting. Therefore, the end users of our system are typically academic researchers or students who, regardless of seniority, are seeking relevant citations or related work.

To this end, we introduce *PaperVeTo* (section 4.1), a graph-based approach to deal with the problem of paper recommendation. *PaperVeTo* exploits latent relationships in scholarly information networks, to identify similarities between papers based on their authors and topics. Then, we present an extended version of *PaperVeTo*, namely *ExtendedPaperVeTo* (section 4.2), that improves upon its predecessor by also incorporating keyword similarity of paper titles and abstracts to generate recommendations. Additionally, we present an overview of the code and technologies used (section 4.3). Finally, we conclude with a short chapter summary (section

4.4)

PaperVeTo is based on *VeTo+* [1] [45], an already existing solution which tries to deal with the problem of expert set expansion [45, 71, 72]. The main difference between *PaperVeTo* and *VeTo+* is that different metapaths are used. Similar to the previous authors, we advocate an 'open source culture'. Therefore, the code for both *PaperVeTo* and *ExtendedPaperVeTo* is publicly available on a github repo [2].

4.1 PaperVeTo Recommender

In this section, we describe *PaperVeTo*; the initial algorithmic solution to deal with the problem of paper recommendation. Since *PaperVeTo* is based on *VeTo+* [45] the concept is quite similar.

The key idea behind it is that it considers the metapath-based similarity (see section 2.4) of papers to papers of interest (POIs) to form the list of the recommended papers. In particular, two metapaths are utilised: *Paper- Author - Paper* (`PAP`) and *Paper - Topic - Paper* (`PTP`) that capture two distinctly different paper similarities. Specifically, the former takes into consideration papers that have common authors, while the latter papers with common topics. *PaperVeTo* essentially combines these similarities of papers according to these two metapaths to create the list of suggested papers.

In the following, we present *PaperVeTo* in detail (section 4.1.1). Moreover, as multiple ranked lists need to be merged in various steps, we outline the considered rank aggregation approaches (section 4.1.2). Finally, we present the configuration parameters of the algorithm (section 4.1.3).

4.1.1 Method Description

Given a set of papers of interest (POIs) $POIs$, a set of candidate papers CP, n the number of recommendations to be made and the weighting parameters α, β such that $\alpha, \beta \in [0, 1]$ and $\alpha + \beta = 1$, *PaperVeTo* performs the following steps similar to

[1] `https://github.com/schatzopoulos/VeTo-workflows`

[2] `https://github.com/gbouzioto/VeTo-workflows/tree/master/paper_recommender`

[1]:

Step 1. For each paper $p \in POIs$, the top-k most similar papers based on the `PAP` metapath are identified as candidates, with k being a method parameter. The *JoinSim* [42, 45] (see equation 2.4 for more details) similarity measure is used to calculate these metapath-based similarities.

This step produces a ranked list $R^{p}_{PAP} = \{p_1, p_2, ..., p_k | p_i \in CP\}$ for each paper p, that contains candidate papers in descending order based on their similarity score with p according to the `PAP` metapath.

Step 2. A rank aggregation algorithm is performed on the set $\{R^{p}_{PAP} | p \in POIs\}$, containing the ranked lists for all papers, weighted by the parameter $\alpha \in [0, 1]$, based on metapath `PAP` from Step 1. This produces the weighted aggregated ranked list WR_{PAP} that ranks all paper candidates in descending order based on their similarities to all POIs according to `PAP`.

Step 3. Similarly, the weighted ranked list WR_{PTP} is produced, repeating Steps 1 & 2 using similarities based on the `PTP` metapath, weighted by the parameter $\beta \in [0, 1]$. It contains all paper candidates according to their "aggregated" similarity score based on `PTP` (in descending order).

Step 4. Ranked weighted lists WR_{PAP} and WR_{PTP} are merged into the final aggregated list R_{fin} using a rank aggregation algorithm. R_{fin} takes into account similarities between papers of interest $POIs$ and candidate papers CP based on both metapaths.

Step 5. The final result consists of the top-n papers of R_{fin}, when sorted in descending order based on their overall aggregated score.

4.1.2 Rank Aggregation

The notion of a rank aggregation algorithm is outlined below [1]:

Given a set of k items to be ranked $I = \{i_1, \ldots, i_k\}$, and a set of m alternative rankings of these items $\{R_1, \ldots, R_m\}$, a rank aggregation algorithm combines these rank orderings to obtain an "aggregated" ranking list. Each ranking R_j is considered as a set of k pairs (one for each $i \in I$) of the form $\langle i, s_j^i \rangle$, with $i \in I$ and s_j^i being i's ranking score according to R_j.

PaperVeTo utilises the same rank aggregation algorithms as *VeTo+* [45], namely:

- *Borda Count (BC)* [73] is essentially a positional voting algorithm: intuitively, each item aggregates points based on the number of items ranked lower than it in all given rankings. Therefore, the new ranked list is computed as follows:

$$R_{BC} = \left\{ \langle i, s_{BC}^i \rangle : i \in I,\ s_{BC}^i = \sum_{j \in [1,m]} k - rank(i, R_j) + 1 \right\}$$

  where $rank(i, R_j)$ denotes the rank of item i (i.e., its order) in ranking R_j.

- *Sum* [1]: This is a simple algorithm that aggregates the score of each item by summing its individual scores across all considered input ranking lists:

$$R_S = \left\{ \langle i, s_S^i \rangle : i \in I, s_S^i = \sum_{j \in [1,m]} s_j^i \right\}$$

- *Reciprocal Rank Fusion (RRF)* [1, 74]: Despite the fact that highly ranked items are important, the importance of lower ranked ones should be preserved. To this end, RRF introduces a configuration parameter λ to adjust the importance of items based on their ranking in the input ranking lists. Therefore, it computes the aggregated ranking list as follows:

$$R_R = \left\{ \langle i, s_R^i \rangle :\ i \in I,\ s_R^i = \sum_{j \in [1,m]} \frac{1}{\lambda + rank(i, R_j)} \right\}$$

  Intuitively, small values of λ promote highly ranked items, while large values mitigate their importance compared to those in lower ranks.

- *CombMNZ* [1, 75]: This is a simple yet effective algorithm that serves as a baseline method when comparing different rank aggregation and data fusion approaches [1, 76–78]. Essentially, it assigns a weighting factor to each item based on its presence in all considered input ranking lists:

$$R_C = \left\{ \langle i, s_C^i \rangle : i \in I, s_C^i = (\sum_{j \in [1,m]} s_j^i) * q_i \right\}$$

where q_i is the number of ranking lists that contribute to the aggregated score of i. Note that CombMNZ requires normalisation of scores before aggregation.

4.1.3 Configuration Parameters

In the following section, we briefly describe the parameters for the *PaperVeTo* algorithm. Note that these parameters are only for the algorithm presented in the section 4.1.1 and not for the python scripts provided in the github repo [3], some of them are present there as well but the naming convention is different. However, there is sufficient documentation [4] on the repo itself. Table 4.1 indicates the parameters for *PaperVeTo*:

- α & β: the weights for the two considered similarity types (see section 4.1).
- k: used in Steps 1 & 3 (see section 4.1.1); it dictates the number of top items of each list to be considered as candidates.
- n: used in Step 5 (see section 4.1.1); indicates the number of $top - n$ final recommendations.
- $rank$: the rank aggregation algorithm to be used; it takes one of the following values: 'BC', 'Sum', 'RRF' or 'CombMNZ'.
- λ: this is a parameter of the the RRF rank aggregation algorithm (see section 4.1.2), hence it is used in case that $rank = RRF$.

[3]`https://github.com/gbouzioto/VeTo-workflows/tree/master/paper_recommender`

[4]`https://github.com/gbouzioto/VeTo-workflows/blob/master/paper_recommender/README.md`

4.2 ExtendedPaperVeTo Recommender

In this section, we describe *ExtendedPaperVeTo*, which is an extended version of *PaperVeTo* algorithm. The main difference between the two approaches is that, apart from the metapath-based similarity (see section 2.4), the extended version also considers keyword similarity to generate recommendations.

In particular, the whole dataset is inserted into a MongoDB [5] database, upon which full text search indexes are built (see section 2.5) for two columns, namely the paper's title and abstract. The relevance score is then retrieved through a db query. The query contains the keywords from the paper's of interest (POI) title. These are retrieved by removing all the stop words and punctuation from the original title. That was achieved using the NLTK [6] python library.

In the following, we present *ExtendedPaperVeTo* in detail (section 4.2.1). Rank aggregation approaches (section 4.2.2) are presented briefly, since they are quite similar with the previous implementation (section 4.1.2). Finally, we present the configuration parameters of the extended algorithm (section 4.2.3).

4.2.1 Method Description

Given a set of papers of interest (POIs) $POIs$, a set of candidate papers CP, n the number of recommendations to be made, the weighting parameters α, β, γ such that $\alpha, \beta, \gamma \in [0, 1]$ and $\alpha+\beta+\gamma = 1$ and a database DB containing the candidate papers

[5]`https://www.mongodb.com/`
[6]`https://www.nltk.org/`

Table 4.1: Configuration parameters for *PaperVeTo*.

Parameter	Values
$\boldsymbol{\alpha}$ & $\boldsymbol{\beta}$	$\alpha, \beta \in [0, 1], \alpha + \beta = 1$
$\boldsymbol{k}$	$k \in \mathbb{N}$
$\boldsymbol{n}$	$n \in \mathbb{N}$
$\boldsymbol{rank}$	{ BC, Sum, RRF, CombMNZ }
$\boldsymbol{\lambda}$	$\lambda \in \mathbb{N}$

indexed by their title p_t and abstract p_{abs}, using a full text search index and two weighting factors $w_1, w_2 \in \mathbb{N}$, which denote the relative significance of the indexed fields to each other[7], *ExtendedPaperVeTo* performs the following steps:

Step 1. For each paper $p \in POIs$, the top-k most similar papers based on the `PAP` metapath are identified as candidates, with k being a method parameter. The *JoinSim* [42, 45] (see equation 2.4 for more details) similarity measure is used to calculate these metapath-based similarities.

This step produces a ranked list $R^p_{PAP} = \{p_1, p_2, ..., p_k | p_i \in CP\}$ for each paper p, that contains candidate papers in descending order based on their similarity score with p according to the `PAP` metapath.

Step 2. A rank aggregation algorithm is performed on the set $\{R^p_{PAP} | p \in POIs\}$, containing the ranked lists for all papers, weighted by the parameter $\alpha \in [0, 1]$, based on metapath `PAP` from Step 1. This produces the weighted aggregated ranked list WR_{PAP} that ranks all paper candidates in descending order based on their similarities to all POIs according to `PAP`.

Step 3. Similarly, the weighted ranked list WR_{PTP} is produced, repeating Steps 1 & 2 using similarities based on the `PTP` metapath, weighted by the parameter $\beta \in [0, 1]$. It contains all paper candidates according to their "aggregated" similarity score based on `PTP` (in descending order).

Step 4. For each paper title $p \in POIs$, the top-k most similar papers based on the keyword similarity are identified as candidates, with k being a method parameter. The similarity is measured by performing a query q to the DB and then retrieving the k papers based on the database relevance paper score, for each p. The query consists of the p_{wt}, namely the p_t with all of its stopwords, *stop_w*, and punctuation, *punc*, removed. p_{wt} is given by:

$p_w t^p = \{word_1, word_2, ..., word_k | word_i \in p_t, \notin stop\_w, punc\}$

This step produces a ranked list $R^p_{keyword} = \{p_1, p_2, ..., p_k | p_i \in CP\}$ for each

[7]For instance if the *title* field has a weight of 5 and the *abstract* field a weight of 20, a term match in the *abstract* field has 4 times (i.e. 20:5) the impact as a term match in the *title* field and vice versa.

paper p, that contains candidate papers in descending order based on their keyword relevance score to p.

Step 5. A rank aggregation algorithm is performed on the set $\{R^p_{keyword} | p \in POIs\}$, containing the ranked lists for all papers, weighted by the parameter $\gamma \in [0, 1]$, based on the queries per p from Step 4. This produces the weighted aggregated ranked list $WR_{keyword}$ that ranks all paper candidates in descending order based on their similarities to the POIs according to their keyword similarity score.

Step 6. Ranked weighted lists WR_{PAP}, WR_{PTP} and $WR_{keyword}$ are merged into the final aggregated list R_{fin} using a rank aggregation algorithm. R_{fin} takes into account similarities between papers of interest $POIs$ and candidate papers CP based on both metapaths as well as their keyword similarity score to POIs.

Step 7. The final result consists of the top-n papers of R_{fin}, when sorted in descending order based on their overall aggregated score.

4.2.2 Rank Aggregation

ExtendedPaperVeTo utilises the same rank aggregation algorithms as *PaperVeTo* (see section 4.1.2) with the exception that for Step 5, the *Borda Count (BC)* aggregation algorithm is the only option and therefore hardcoded.

4.2.3 Configuration Parameters

In the following section, we briefly describe the parameters for the *ExtendedPaperVeTo* algorithm. Note that, as with *PaperVeTo* (see section 4.1.3), these parameters are only for the algorithm presented in the section 4.2.1 and not for the python scripts provided in the github repo [8]. Table 4.2 indicates the parameters for *PaperVeTo*:

- α & β & γ: the weights for the three considered similarity types (see section 4.2).

[8] `https://github.com/gbouzioto/VeTo-workflows/tree/master/paper_recommender`

- w_1 & w_2: the full text search weights for the papers' titles and abstracts DB indexes (see section 4.2.1).
- k: used in Steps 1 & 3 & 5 (see section 4.2.1); it dictates the number of top items of each list to be considered as candidates.
- n: used in Step 7 (see section 4.2.1); indicates the number of $top-n$ final recommendations.
- $rank$: the rank aggregation algorithm to be used; it takes one of the following values: 'BC', 'Sum', 'RRF' or 'CombMNZ'.
- λ: this is a parameter of the the RRF rank aggregation algorithm (see section 4.1.2), hence it is used in case that $rank = RRF$.

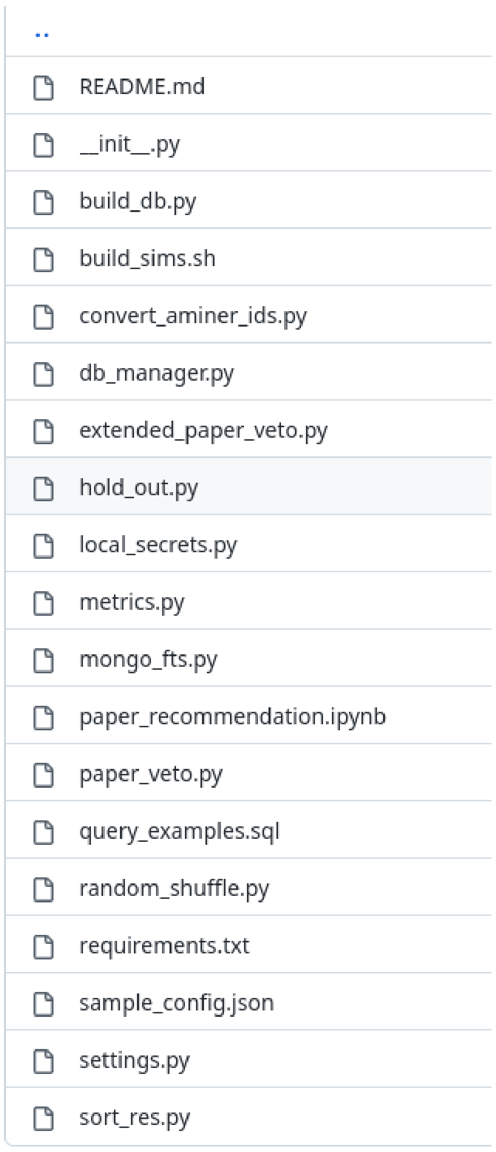
Table 4.2: Configuration parameters for *ExtendedPaperVeTo*.

Parameter	**Values**
$\boldsymbol{\alpha}$ **&** $\boldsymbol{\beta}$ **&** $\boldsymbol{\gamma}$	$\alpha, \beta, \gamma \in [0,1], \alpha+\beta+\gamma = 1$
$\boldsymbol{w_1}$ **&** $\boldsymbol{w_2}$	$w_1, w_2 \in \mathbb{N}$
$\boldsymbol{k}$	$k \in \mathbb{N}$
$\boldsymbol{n}$	$n \in \mathbb{N}$
$\boldsymbol{rank}$	{ BC, Sum, RRF, CombMNZ }
$\boldsymbol{\lambda}$	$\lambda \in \mathbb{N}$

4.3 Code Overview

In this section, we discuss the code and technologies used to build *PaperVeTo*, *MongoFTS* and *ExtendedPaperVeTo* recommendation approaches. The code is located in a public github repo[9] with sufficient documentation[10] on how to run each method. Since this implementation extends the previous $VeTo+$[11] implementation the contents are in a new directory called **paper_recommender**. Figure 4.1 illustrates the contents of the aforementioned directory.

[9]`https://github.com/gbouzioto/VeTo-workflows/tree/master/paper_recommender`
[10]`https://github.com/gbouzioto/VeTo-workflows/blob/master/paper_recommender/README.md`
[11]`https://github.com/gbouzioto/VeTo-workflows`



..
README.md
__init__.py
build_db.py
build_sims.sh
convert_aminer_ids.py
db_manager.py
extended_paper_veto.py
hold_out.py
local_secrets.py
metrics.py
mongo_fts.py
paper_recommendation.ipynb
paper_veto.py
query_examples.sql
random_shuffle.py
requirements.txt
sample_config.json
settings.py
sort_res.py

Figure 4.1: Contents of paper_recommender directory.

Python 3.8.9[12] was used to run the recommendation scripts, along with some extra packages listed in the **requirements.txt**[13] file, such as NLTK[14] [15]. MongoDB version v4.4.6[16] was also used to store the data.

In order to build the similarities needed for the VeTo algorithms, files from the DBLP Article Similarities (DBLP-ArtSim) dataset[17] were used and more specifically

[12]https://www.python.org/downloads/release/python-389/
[13]https://github.com/gbouzioto/VeTo-workflows/tree/master/paper_recommender/requirements.txt
[14]https://www.nltk.org/data.html
[15]Used to remove stopwords and punctuation.
[16]https://www.mongodb.com/docs/manual/installation/
[17]https://zenodo.org/record/4567527#.Y6XVpNJBzWl

the following files:

- **PAP.csv.gz**[18], contains the paper-author-paper metapath sim scores
- **PTP.csv.gz**[19], contains the paper-topic-paper metapath sim scores
- **aminer_ids.csv.gz**[20], contains the aminer to veto id mapping

To build the similarities for the recommender scripts, a bash script is provided, **build_sims.sh**. It accepts a configuration file as a parameter. The configuration file is in JSON format. a sample file named **sample_config.json**[21] is provided. Its main parameters are described on table 4.3.

Table 4.3: Configuration for sample_config.json file.

Parameter Name	**Description**
pap_hin	a tsv file encoding the metapath-based HIN view according to the PAP metapath with columns: source, destination, number of paths.
ptp_hin	a tsv file encoding the metapath-based HIN view according to the PTP metapath with columns: source, destination, number of paths.
pap_sims_dir	the folder to store the PAP based similarities
ptp_sims_dir	the folder to store the PTP based similarities

Before building the mongo database, a credential file, named local_secrets.py[22], is provided. Table 4.4 shows the variables of the credential file, along with some dummy values as an example.

Given the above file with valid credentials, MongoDB can be built by utilising the **build_db.py**[23] script. The script arguments are outlined on table 4.5.

To produce recommendations using *MongoDB full text search (MongoFTS)* the script **mongo_fts.py**[24] can be used. The script arguments are outlined on table 4.6.

[18]`https://zenodo.org/record/4567527/files/PAP.csv.gz?download=1`
[19]`https://zenodo.org/record/4567527/files/PTP.csv.gz?download=1`
[20]`https://zenodo.org/record/4567527/files/aminer\_ids.csv.gz?download=1`
[21]`https://github.com/gbouzioto/VeTo-workflows/tree/master/paper_recommender/sample\_config.json`
[22]`https://github.com/gbouzioto/VeTo-workflows/tree/master/paper_recommender/local_secrets.py`
[23]`(https://github.com/gbouzioto/VeTo-workflows/tree/master/paper_recommender/build\_db.py`
[24]`(https://github.com/gbouzioto/VeTo-workflows/tree/master/paper_recommender/mongo_fts.py`

Table 4.4: MongoDB credentials - local_secrets.py file.

Variable Name	Description
DB_NAME	foobar
DB_HOST	dummy_db
DB_USER	postgres
DB_PWD	12345
DB_PORT	5910

Table 4.5: Script build_db.py arguments.

Argument Name	Description
'-f' / '–files'	A string containing the dataset filepaths, can be comma separated to included multiple filepaths
'-af' / '–aminer_ids_file'	The csv file containing the aminer to veto id mapping
'-tw' / '–title_weight'	The search weight for the paper title
'-aw' / '–abstract_weight'	The search weight for the paper abstract

Table 4.6: Script mongo_fts.py arguments.

Argument Name	Description
'-vf', '–veto_id_file'	The filepath containing the veto paper ids (**newline separated**)
'-mp', '–max_papers'	max similar paper number per query
'-mr', '–max_results'	max number of results

To produce recommendations using *PaperVeto* the script **paper_veto.py**[25] can be used. The script arguments are outlined on table 4.7.

Table 4.7: Script paper_veto.py arguments.

Argument Name	Description
'-pf' / '–paper_file'	the filepath containing the veto paper ids (**newline separated**)
'-vo' / '–veto_output'	filepath where the results will be written
'-pap' / '–pap_sims'	directory containing the PAP similarity scores
'-ptp' / '–ptp_sims'	directory containing the PTP similarity scores
'-spe' / '–sims_per_paper'	how many similarities per paper should be considered
'-papw' / '–pap_weight'	score weight for the PAP similarities
'-ptpw' / '–ptp_weight'	score weight for the PTP similarities
'-algo' / '–algorithm'	the scoring algorithm to be used
'-rrfk' / '–rrf_k'	rrf k algorithm ranking parameter (**used only with rrf algorithm**)
'-outs' / '–output_size'	the size of the output

To produce recommendations using *ExtendedPaperVeto* the script **extended_paper_veto.py**[26] can be used. The script arguments are outlined on table 4.8.

Table 4.8: Script extended_paper_veto.py arguments.

Argument Name	Description
'-pf' / '–paper_file'	the filepath containing the veto paper ids (**newline separated**)
'-vo' / '–veto_output'	filepath where the results will be written
'-pap' / '–pap_sims'	directory containing the PAP similarity scores
'-ptp' / '–ptp_sims'	directory containing the PTP similarity scores
'-spe' / '–sims_per_paper'	how many similarities per paper should be considered
'-papw' / '–pap_weight'	score weight for the PAP similarities
'-ptpw' / '–ptp_weight'	score weight for the PTP similarities
'-kw' / '–keyword_weight'	score weight for the keyword similarities
'-algo' / '–algorithm'	the scoring algorithm to be used
'-rrfk' / '–rrf_k'	rrf k algorithm ranking parameter (**used only with rrf algorithm**)
'-outs' / '–output_size'	the size of the output

Some other helper scripts are also included in the repository, that were used to facilitate the evaluation experiments and are not worth delving into detail here.

[25] `https://github.com/gbouzioto/VeTo-workflows/tree/master/paper_recommender/paper\_veto.py`

[26] `https://github.com/gbouzioto/VeTo-workflows/tree/master/paper_recommender/extended_paper_veto.py`

4.4 Summary

In this chapter, we studied the problem of recommending scientific papers in academia; given a set of papers of interest (POIs), the goal is to suggest similar papers of interest to the researchers. In this context, we proposed, *PaperVeTo* and *ExtendedPaperVeTo*, one graph-based and one hybrid scientific paper recommendation approach, that exploit paper similarities to generate recommendations. In particular, *PaperVeTo* and *ExtendedPaperVeTo*, identify similar papers with the ones in the given paper set, considering latent relationships among papers' authors and topics. Furthermore, *ExtendedPaperVeTo* extends its predecessor, *PaperVeTo*, by also exploiting the keyword similarity of paper titles and abstracts. Last but not least, we should note that our work does not take into consideration that scholarly information networks evolve over time. It is interesting to investigate the effect of temporal changes on the performance of *PaperVeTo* and *ExtendedPaperVeTo*; for instance, considering the most recent publications as more important.

Chapter 5

Evaluation

The evaluation of a recommendation system is a crucial step in understanding its effectiveness and identifying areas for improvement. To this end, we present the setup of our experiments (section 5.1). Afterwards, we outline our evaluation methodology (section 5.2) . Finally, we perform a discussion about the evaluation results (section 5.3).

5.1 Experimental setup

In this section we discuss the experimental setup of our evaluation. Initially, we present the datasets that were used (section 5.1.1). Afterwards, we show the proposed recommendation approaches, including the baseline ones, (section 5.1.2), as well as the configuration for each approach (section 5.1.3). Finally, we mention the evaluation metrics that were utilised to measure the performance of our proposed approach (section 5.1.4).

5.1.1 Datasets

For the scope of this work we have utilised two datasets:

1. Aminer's **DBLP-Citation-network V10** [1] [41]: consists of 3.079.007 papers

[1] `https://www.aminer.org/citation`

from DBLP [2], that range from the year 1936 to 2018. Table 5.1 illustrates the dataset's schema. 530.475 papers do not have an abstract. No preprocessing has been made on the dataset.

2. **DBLP Article Similarities (DBLP-ArtSim) dataset** [3] [79]: contains similarity scores among articles in AMiner's DBLP v10 dataset. Similarities are calculated using the JoinSim [42] similarity measure on the derived citation network using the following metapaths: *Paper- Author - Paper* (`PAP`), *Paper - Topic - Paper* (`PTP`) and *Paper - Venue - Paper* (`PVP`). There is also a file which contains the paper to venue (PV) relationships and a file which contains a mapping from AMiner's ids to the internal numeric ids used in the similarities files. Out of all the available files we utilised three: the `PAP` file, the `PTP` file and the `internal-numeric-ids-to-Aminer-ids` file.

Field Name	**Field Type**	**Description**	**Example**
id	string	paper ID	013ea675-bb58-42f8-a423-f5534546b2b1
title	string	paper title	Prediction of consensus binding mode geometries for related chemical series of positive allosteric modulators of adenosine and muscarinic acetylcholine receptors
authors	list of strings	paper authors	["Leon A. Sakkal", "Kyle Z. Rajkowski", "Roger S. Armen"]
venue	string	paper venue	Journal of Computational Chemistry
year	int	published year	2017
n_citation	int	citation number	0
references	list of strings	citing papers' ID	["4f4f200c-0764-4fef-9718-b8bccf303dba", "aa699fbf-fabe-40e4-bd68-46eaf333f7b1"]
abtract	string	abstract	This paper studies ...

Table 5.1: DBLP-Citation-network V10 schema + example.

5.1.2 Recommendation Approaches

In the following, we briefly describe the three approaches that were applied on the paper recommendation problem in our experiments:

[2]https://dblp.uni-trier.de/xml/
[3]https://doi.org/10.5281/zenodo.3778915

- *PaperVeTo*, presented in section 4.1
- *ExtendedPaperVeTo*, presented in section 4.2
- *MongoDB full text search (MongoFTS)*, a full search text approach based on keyword similarity. It is essentially the **Step 4** and **Step 5** of the *Extended-PaperVeTo* algorithm, presented in section 4.2.1.

PaperVeTo and *MongoFTS* were used as baseline approaches while *Extended-PaperVeTo* as the suggested approach. All aforementioned approaches were implemented in Python and are publicly available [4]. The metapath-based similarities required by *PaperVeTo* and *ExtendedPaperVeTo* were taken by the DBLP-ArtSim dataset (see section 5.1.1).

5.1.3 Configuration

The configuration for the above three methods is outlined on table 5.2. All parameters have been set empirically. It is also worth adding, that we have considered that paper topics have more relevance impact to users than authors and that papers' abstracts and titles are equally important in terms of keyword matching. As a result, for *PaperVeTo* we have set $\boldsymbol{\alpha}$, PAP metapath weight, to 0.2, $\boldsymbol{\beta}$, PTP metapath weight, to 0.8, k, the similarities considered per paper, to 50, and n, the number of the output results, to 20 and the *rank*, the ranking algorithm, to *Bord Count (BC)*. For *ExtendedPaperVeTo* we have set $\boldsymbol{\alpha}$, PAP metapath weight, to 0.2, $\boldsymbol{\beta}$, PTP metapath weight, to 0.3, $\boldsymbol{\gamma}$, keyword relevance weight, to 0.5, k, the similarities considered per paper, to 50, n, the number of the output results, to 20, both w_1 and w_2, the title and abstract full text search index weights, to 1, and the *rank*, the ranking algorithm, to *Bord Count (BC)*. Finally, for *MongoFTS* we have set k, the similarities considered per paper, to 50, n, the number of the output results, to 20, both w_1 and w_2, the title and abstract full text search index weights, to 1.

[4]`https://github.com/gbouzioto/VeTo-workflows/tree/master/paper_recommender`

Parameter	**PaperVeTo**	**ExtendedPaperVeTo**	**MongoFTS**
$\boldsymbol{\alpha}$	0.2	0.2	N/A
$\boldsymbol{\beta}$	0.8	0.3	N/A
$\boldsymbol{\gamma}$	N/A	0.5	N/A
$\boldsymbol{w_1}$	N/A	1	1
$\boldsymbol{w_2}$	N/A	1	1
$\boldsymbol{k}$	50	50	50
$\boldsymbol{n}$	20	20	20
$\boldsymbol{rank}$	BC	BC	N/A

Table 5.2: Configuration parameter values for *PaperVeTo*, *ExtendedPaperVeTo* and *MongoFTS*.

5.1.4 Evaluation Metrics

The average rating (AR) [14] score and the normalised discounted cumulative gain (NDCG@p) [80] at a position p were selected as performance metrics. For both metrics, the positions 1, 5, 10 and 20 were picked. Since when making recommendations the order of the results typically matters, for example a user may miss accurate results if they are on the bottom of the suggestions, we consider that NDCG is more impactful than AR. However we have chosen to keep AR as a generic indication of the system's relevance performance. Below we present the equations for both metrics:

- **NDCG@p**: is a measure of ranking quality at a position p, based on the assumptions that highly relevant documents are more useful when appearing earlier in a result list and than marginally relevant documents, which are in turn more useful than non-relevant documents. It is given by:

$$nDCG@p = \frac{\sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)}}{\sum_{i=1}^{p} \frac{rel_{ideal,i}}{\log_2(i+1)}}$$

  where rel_i is the relevance score of the item at position i, and $rel_{ideal,i}$ is the relevance score of the item at position i in the ideal ranking. The value of p represents the position at which the NDCG is calculated.

- **Average Rating (AR) Score**: is a measure of ranking quality at a position

p without considering the position of the document in the result list. It is given by:

$$AR@p = \frac{\sum_{i=1}^{p} rating_i}{p}$$

where $rating_i$ is the rating of the item at position i, and p is the position at which the AR is calculated.

5.2 Evaluation Methodology

In this section, we present our evaluation methodology for the proposed system. The methodology is comprised of three main components: a description of the evaluation method and the intuition behind our framework (section 5.2.1), of the user study we have conducted to assess the system's performance (section 5.2.2), and a series of experiments along with the results to provide further insight into the system's behavior (section 5.2.3). In the following, we will delve into each of these components in detail to provide a thorough understanding of our evaluation process.

5.2.1 Method Description

The intuition behind our framework is to propose similar papers to users according to their interests and based on data from real-life applications. For instance, consider a user who has some available reading lists in a website such as *BIP! Finder* [5] and would like to find similar papers to expand his research. To evaluate our system's suggestions, for multiple user paper lists P_{ln}, which represent the users' interests, and given an available corpus of papers C_p, which that website would have, we perform the following steps:

Step 1. The proposed recommendation approach along with other two baseline approaches are applied on each user list P_l. For each P_l, three new recommendation lists are generated, P_{l1}, P_{l2}, P_{l3}, of equal size n, containing papers from C_p excluding all papers present in P_l. The papers in each list are ordered based on their relevance score from highest to lowest.

[5] `https://bip.imsi.athenarc.gr/`

Step 2. Each P_l, the lists from the previous step, as well as their items are shuffled producing three new ones, P_{rl1}, P_{rl2}, P_{rl3}. The papers are no longer ordered by their relevance score. In order to avoid bias, the shuffling process is randomized, meaning that it is different for each user.

Step 3. Three distinct lists are provided to each user. Users do not know which of the three approaches used, produced each list. They are then asked to rate each paper on each result-list. Each rating, $r \in [0, 2], \mathbb{N}$ represents how much the user finds the paper relevant:

- $r = 0$ means that the paper is not relevant
- $r = 1$ means that the paper is somewhat relevant
- $r = 2$ means that the paper is very relevant

Step 4. Finally, based on the results from the previous step, for each P_l, proper information retrieval measures are applied on the three original lists P_{l1}, P_{l2}, P_{l3}.

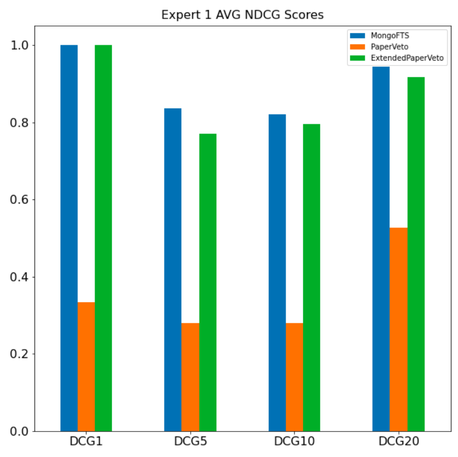
5.2.2 User Study

As mentioned in the previous section, the intention of our system is to be used by real users. Therefore, we have chosen to conduct a lab-based preliminary user study to evaluate the initial recommendation results. In this study, two domain experts E_1 and E_2 have participated and rated the produced paper lists. Regarding the academic degree, both experts have a Ph.D. Each expert chose thirteen papers of interest to provide as input to the system. Table 5.3 illustrates the choice of each paper list per expert. The papers are represented by their *Aminer id* and title, namely the *id* and *title* fields found in the *DBLP-Citation-network V10* dataset (section 5.1.1).

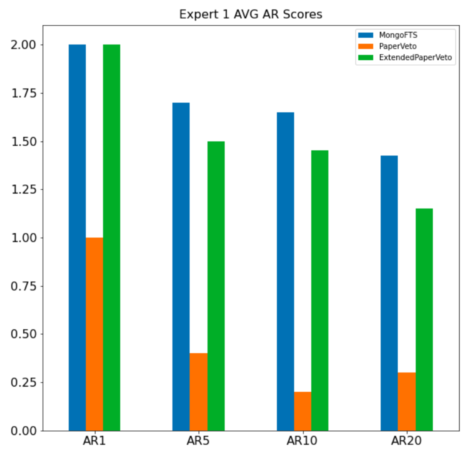
5.2.3 Experimental Evaluation

Following the process described in section 5.2.1 we have conducted four experiments, two for each expert. Two of the experiments contained the full input list of each expert (see table 5.3). For the other two experiments, we have randomly removed 30% of the input for each expert list, to evaluate results with smaller input.

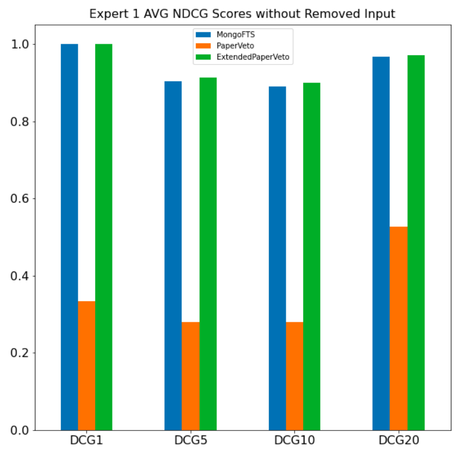
The metrics of average rating (AR) score and the normalised discounted cumulative gain (NDCG@p) at the positions 1, 5, 10 and 20 were selected, as mentioned in section 5.1.4. Table 5.4 illustrates the results per expert, table 5.5 the total results based on the removed input and table 5.6 the total results. In a similar notion, figure 5.1 illustrates the results for E_1, figure 5.2 the results for E_2 and figure 5.3 the total results for both experts. In these figures blue color stands for *MongoFTS* approach, orange for *PaperVeTo* and green for *ExtendedPaperVeTo* respectively.



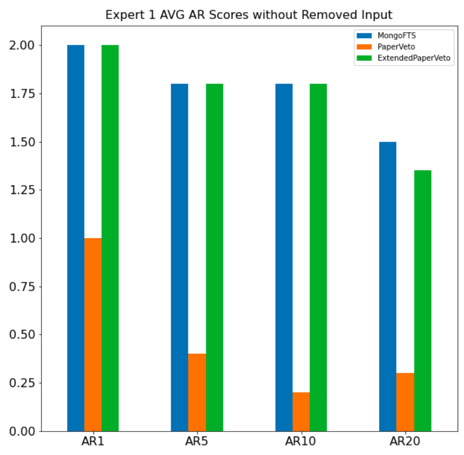
(a) AVG NDCG



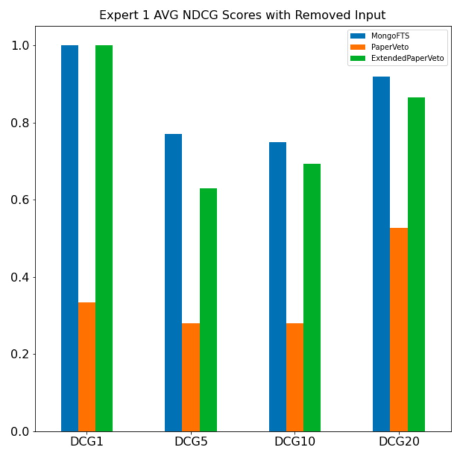
(b) AVG AR



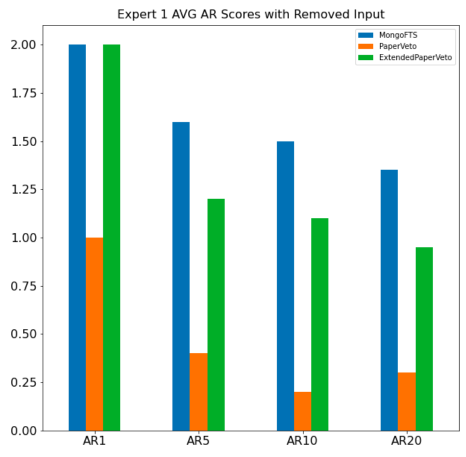
(c) AVG NDCG FULL IN



(d) AVG AR FULL IN



(e) AVG NDCG REMOVED IN



(f) AVG NDCG REMOVED IN

Figure 5.1: NDCG and AR results for E_1.

Table 5.3: Paper input lists per expert, papers are represented by their *DBLP-Citation-network V10* id and title fields.

E_1		E_2	
ID	**Title**	**ID**	**Title**
6cbd879b-32e4-4f75-bddf-d53c6efb3237	P-Rank: An indicator measuring prestige in heterogeneous scholarly networks	5ad4e6e8-0caa-4477-afe3-f8aff29c155e	Constrained-meta-path-based ranking in heterogeneous information network
d194966e-870e-49df-893f-1b3ed313fbbe	Weighted citation: An indicator of an article's prestige	772d63fd-2a4a-4132-a6f8-ad29cf216bc4	Top-k Similarity Join in Heterogeneous Information Networks
4735fa71-649c-483d-9925-61bd24b612b0	FutureRank: Ranking Scientific Articles by Predicting their Future PageRank	f4ea412e-d335-4805-b255-63e978d27c10	HRank: A Path based Ranking Framework in Heterogeneous Information Network
e6a6a2c7-a829-4ca0-9d97-052526cc8826	Ranking scientific publications with similarity-preferential mechanism	eea965fa-c41b-4d6c-bdc8-f5d6f906d71c	Collective Prediction of Multiple Types of Links in Heterogeneous Information Networks
b86c4747-9e2e-4411-ab32-6c77b41a5e8a	Time-Aware Ranking in Dynamic Citation Networks.	087f5698-5bf0-4a04-baa3-c13547b9e9f1	HeteSim: A General Framework for Relevance Measure in Heterogeneous Networks
b8f947e5-03a3-468c-b418-944436c0f5ab	Comparing paper ranking algorithms	2bbe0264-a33b-4069-86eb-d178698da2c6	Ranking-based Clustering on General Heterogeneous Information Networks by Network Projection
c726a2d1-050a-4989-8fa8-8643600fb752	Yet another paper ranking algorithm advocating recent publications	773ea39f-d8c8-44fd-a9d0-b1b6e9519158	Social influence based clustering of heterogeneous information networks
bfbbc4d4-2de7-4fba-b2da-47aac8af8ca8	Tri-Rank: An Authority Ranking Framework in Heterogeneous Academic Networks by Mutual Reinforce	a5fc8aed-b007-4ed1-92ee-81fad0dc67d2	PathSelClus: Integrating Meta-Path Selection with User-Guided Object Clustering in Heterogeneous Information Networks
96905049-f689-4ea2-a8a3-772c24735df1	Identification of milestone papers through time-balanced network centrality	fdcb9ca3-87ad-441b-96ea-5eb7a78d890b	PathRank: a novel node ranking measure on a heterogeneous graph for recommender systems
9c1dc3ca-6b97-401d-a84e-d929e49ff93c	Adding the Temporal Dimension to Search - A Case Study in Publication Search	c11852d6-2165-4135-9d66-6b1208fc9403	MultiRank: co-ranking for objects and relations in multi-relational data
eb3dfc11-52eb-461f-bcf9-e49ab7e2bb42	PrestigeRank: A new evaluation method for papers and journals	51cfdd10-1566-4380-adae-3889630ebaf4	Pathsim: Meta path-based top-k similarity search in heterogeneous information networks
4a47b5d9-2479-4618-9261-e8ec78477963	Focused Page Rank in Scientific Papers Ranking	b336dab8-0c89-4100-896b-c01ada791b11	Ranking-based clustering of heterogeneous information networks with star network schema
b671963e-f9d3-4f8c-ba81-ebb4831e1e44	Towards an effective and unbiased ranking of scientific literature through mutual reinforcement	f23a5b3a-8ac9-4820-abcb-9ad0b73e528e	RankClus: integrating clustering with ranking for heterogeneous information network analysis

Approach	DCG1	DCG5	DCG10	DCG20	AR1	AR5	AR10	AR20	Expert	Removed Input
MongoFTS	1.0	0.903	0.891	0.968	2.0	1.8	1.8	1.5	1	no
PaperVeTo	0.333	0.28	0.28	0.527	1.0	0.4	0.2	0.3	1	no
ExtendedPaperVeTo	1.0	0.913	0.899	0.972	2.0	1.8	1.8	1.35	1	no
MongoFTS	1.0	0.77	0.75	0.919	2.0	1.6	1.5	1.35	1	yes
PaperVeTo	0.333	0.28	0.28	0.527	1.0	0.4	0.2	0.3	1	yes
ExtendedPaperVeTo	1.0	0.629	0.694	0.864	2.0	1.2	1.1	0.95	1	yes
MongoFTS	1.0	1.0	1.0	0.995	2.0	2.0	2.0	1.75	2	no
PaperVeTo	1.0	0.744	0.601	0.846	2.0	1.6	1.0	1.05	2	no
ExtendedPaperVeTo	1.0	1.0	1.0	0.999	2.0	2.0	2.0	1.9	2	no
MongoFTS	1.0	1.0	1.0	0.996	2.0	2.0	2.0	1.9	2	yes
PaperVeTo	1.0	0.41	0.619	0.802	2.0	0.6	1.2	0.95	2	yes
ExtendedPaperVeTo	1.0	1.0	1.0	0.997	2.0	2.0	2.0	1.85	2	yes

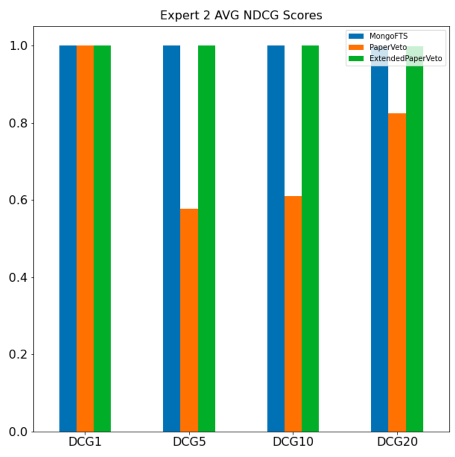
Table 5.4: Results per Expert.

Approach	DCG1	DCG5	DCG10	DCG20	AR1	AR5	AR10	AR20	Removed Input
MongoFTS	1.0	0.885	0.875	0.958	2.0	1.8	1.75	1.625	no
PaperVeTo	0.666	0.345	0.45	0.665	1.5	0.5	0.7	0.625	no
ExtendedPaperVeTo	1.0	0.814	0.847	0.93	2.0	1.6	1.55	1.4	no
MongoFTS	1.0	0.885	0.875	0.958	2.0	1.8	1.75	1.625	yes
PaperVeTo	0.666	0.345	0.45	0.665	1.5	0.5	0.7	0.625	yes
ExtendedPaperVeTo	1.0	0.814	0.847	0.93	2.0	1.6	1.55	1.4	yes

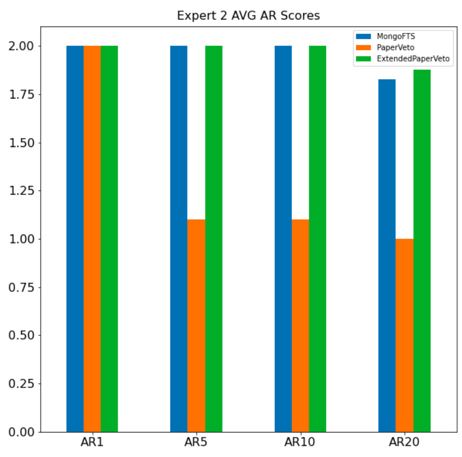
Table 5.5: Total Results per removed input.

Approach	DCG1	DCG5	DCG10	DCG20	AR1	AR5	AR10	AR20
MongoFTS	1.0	0.918	0.91	0.97	2.0	1.85	1.825	1.625
PaperVeTo	0.666	0.428	0.445	0.676	1.5	0.75	0.65	0.65
ExtendedPaperVeTo	1.0	0.886	0.898	0.958	2.0	1.75	1.725	1.512

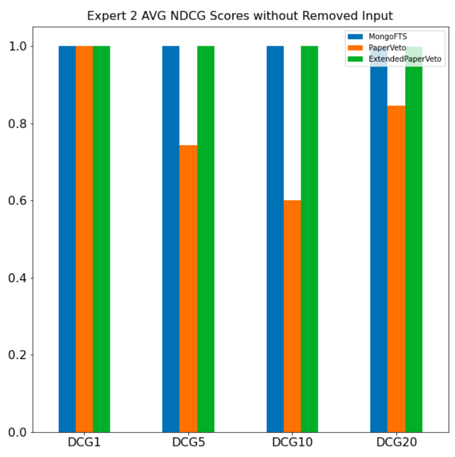
Table 5.6: Total Results.



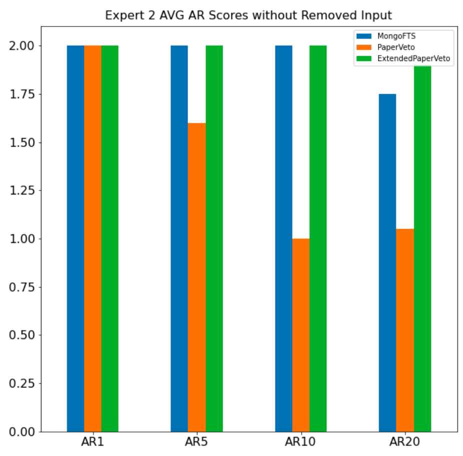
(a) AVG NDCG



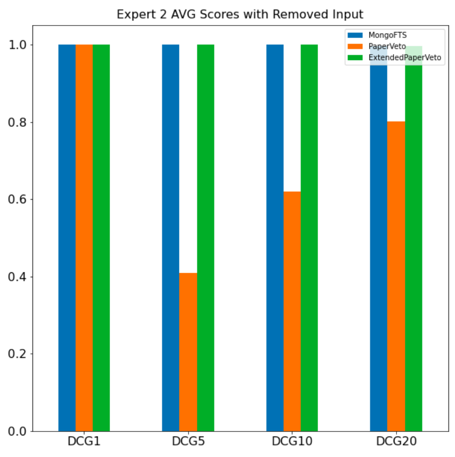
(b) AVG AR



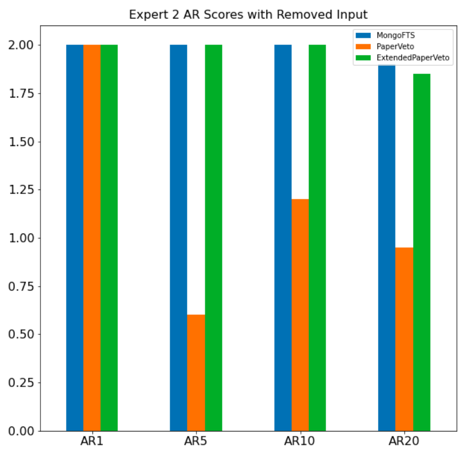
(c) AVG NDCG FULL IN



(d) AVG AR FULL IN

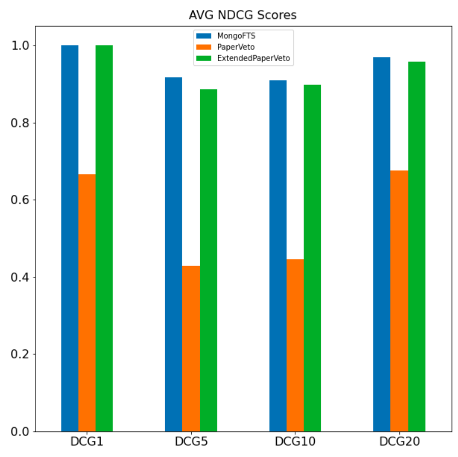


(e) AVG NDCG REMOVED IN

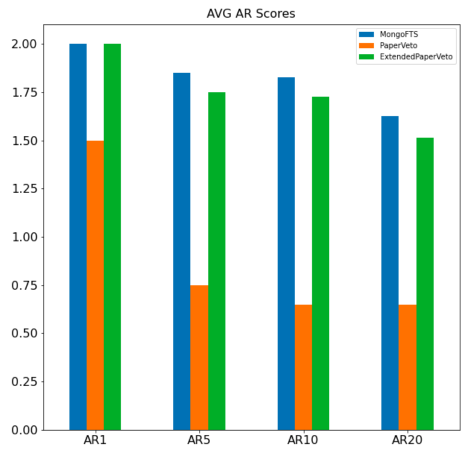


(f) AVG NDCG REMOVED IN

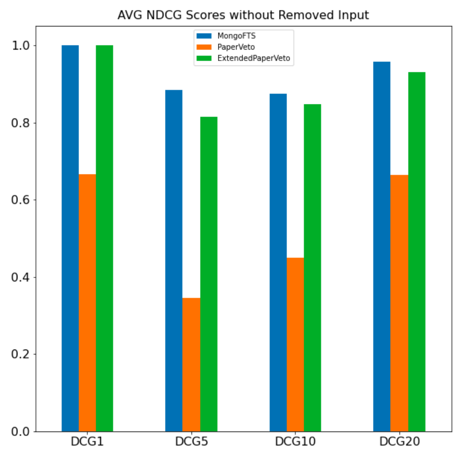
Figure 5.2: NDCG and AR results for E_2.



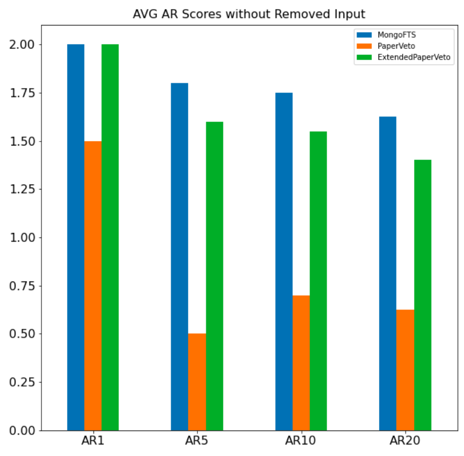
(a) AVG NDCG



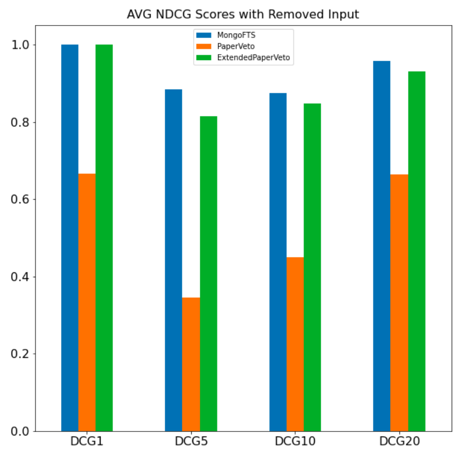
(b) AVG AR



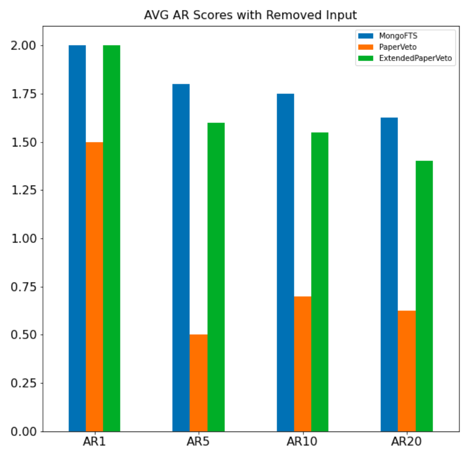
(c) AVG NDCG FULL IN



(d) AVG AR FULL IN



(e) AVG NDCG REMOVED IN



(f) AVG NDCG REMOVED IN

Figure 5.3: NDCG and AR results for both experts.

5.3 Discussion

The results of our preliminary experiments revealed, as illustrated in the previous figures and tables (section 5.2.3), that the suggested approach, *ExtendedPaperVeTo*, is marginally outperformed by the baseline approach, *MongoFTS* in terms of NDCG and AR scores. Both approaches significantly outperformed the *PaperVeTo* approach. We can also observe that results of *ExtendedPaperVeTo* approach, for E_2 are slightly better than *MongoFTS*. The opposite applies for E_1. That is expected however, since *ExtendedPaperVeTo* is essentially the combination of *MongoFTS* and *PaperVeTo* with the later yielding better results for E_2.

It is also worth mentioning the limitations of our evaluation. To begin with, it was very resource-costly to conduct a proper user study with a few dozens of participants. As a result, our user study was limited to two people and results may not be as meaningful as they would be in a larger study. Another limitation due to time constrains, is that we did not make exhaustive parameter configuration trials but rather tried some configurations empirically and used the most promising one. With a different set of configurations, results could have been different. Furthermore, rather than returning a combined list of the results to the end users, we returned three lists, one for each approach. Although the lists and results were scrambled and the end users did not know which approach produced each list, one could argue that returned a merged result list would reduce bias even further. Additionally, we built our approach on top of *VeTo+* [45], a tool which deals with the expert set expansion in academia, and we used pre-calculated metapath-based similarities based on common paper authors and topics, to generate recommendations. Maybe different metapaths, such as keywords provided by the authors metapaths or popularity metapaths based on the number of citations etc., could be taken into consideration. Last but not least, we did not explicitly utilised user profile information in our approach. For example, users could be clustered into groups, based on the context of their paper library.

To conclude, although the preliminary results may not be so encouraging, given the aforementioned limitations, we believe that there is definitely room for improvement. Researchers could expand on the current work to surpass some of the limitations mentioned above or even use the suggested approach as a baseline for other recommendation approaches.

Chapter 6

Conclusions and Future Work

In this study, we conducted a short survey of paper recommendation approaches and evaluated the performance of our own implementation, the *ExtendedPaperVeTo* recommender. Our findings indicate that hybrid approaches are the most common type of paper recommendation approach (PRA) used in the reviewed studies, followed by content-based filtering, graph-based approaches, and collaborative filtering. However, we also identified several challenges and limitations in the current state of the field, including the lack of reproducibility and scalability of many approaches, the limited consideration of the operator's perspective and user characteristics, and the reliance on offline evaluations.

In our implementation evaluation, we observed that the *ExtendedPaperVeTo* approach, which combines the *MongoFTS* and *PaperVeTo* approaches, was slightly outperformed by the *MongoFTS* approach in terms of NDCG and AR scores, but performed better than the *PaperVeTo* approach. However, our evaluation was limited in scope and may not fully capture the performance of the approach in real-world scenarios. Our user study was limited to two participants and more exhaustive parameter configuration trials were not conducted, which may have impacted the results. Additionally, the results of the three approaches were presented separately, rather than combined, which may have introduced bias.

Overall, our findings suggest that there is still room for improvement and innovation in the field of paper recommendation systems. While certain approaches have demonstrated effectiveness in certain contexts, there is a need for more comprehensive and realistic evaluations that consider the needs and characteristics of both the operator and the users.

Moving forward, there are several directions for future research that could address the challenges and limitations identified in our paper recommendation survey and implementation evaluation. One promising direction is the development of more reproducible and scalable recommendation approaches that consider the needs of the operator and the characteristics of the users. This could involve the use of machine learning and natural language processing techniques to better understand the content and context of papers, or the incorporation of user feedback and interaction into the recommendation process.

Another area of focus could be the incorporation of user profile information and alternative metapaths into the recommendation process. For example, researchers could explore the potential of clustering users into groups based on the context of their paper libraries, or using keywords provided by the authors or popularity metrics based on the number of citations as metapaths. In addition, the use of online and user study evaluations could provide a more complete picture of the performance of recommendation systems in real-world scenarios, and help to better understand the impact of recommendations on user behavior.

Finally, it would be interesting to explore the potential applications of recommender systems in different domains and contexts. For example, researchers could investigate the use of recommender systems to recommend articles or research funding opportunities, or to integrate recommendation functionality into existing scholarly information networks or heterogeneous information networks. This could help to improve the effectiveness and efficiency of information discovery and dissemination processes in a variety of settings.

References

[1] Serafim Chatzopoulos. *Data mining for scholarly information networks.* PhD thesis, Πανεπιστήμιο Πελοποννήσου. Σχολή Οικονομίας και Τεχνολογίας. Τμήμα ..., 2022.

[2] Xiaomei Bai, Mengyang Wang, Ivan Lee, Zhuo Yang, Xiangjie Kong, and Feng Xia. Scientific paper recommendation: A survey. *IEEE Access*, 7:9324–9339, 2019.

[3] Tsung Teng Chen and Maria Lee. Research paper recommender systems on big scholarly data. In *Knowledge Management and Acquisition for Intelligent Systems*, pages 251–260. Springer International Publishing, 2018.

[4] Bangchao Wang, Ziyang Weng, and Yanping Wang. A novel paper recommendation method empowered by knowledge graph: for research beginners. *arXiv preprint arXiv:2103.08819*, 2021.

[5] Yi Li, Ronghui Wang, Guofang Nan, Dahui Li, and Minqiang Li. A personalized paper recommendation method considering diverse user preferences. *Decision Support Systems*, 146:113546, 2021.

[6] Khalid Haruna, Maizatul Akmar Ismail, Atika Qazi, Habeebah Adamu Kakudi, Mohammed Hassan, Sanah Abdullahi Muaz, and Haruna Chiroma. Research paper recommender system based on public contextual metadata. *Scientometrics*, 125(1):101–114, aug 2020.

[7] Hao Tang, Baisong Liu, and Jiangbo Qian. Content-based and knowledge graph-based paper recommendation: Exploring user preferences with the knowledge graphs for scientific paper recommendation. *Concurrency and Computation: Practice and Experience*, 33(13):e6227, 2021.

[8] Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. Paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17(4):305–338, 2016.

[9] Zhi Li and Xiaozhu Zou. A review on personalized academic paper recommendation. *Comput. Inf. Sci.*, 12(1):33–43, 2019.

[10] Travis Ebesu and Yi Fang. Neural citation network for context-aware citation recommendation. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, aug 2017.

[11] Chandra Bhagavatula, Sergey Feldman, Russell Power, and Waleed Ammar. Content-based citation recommendation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, 2018.

[12] Guibing Guo, Bowei Chen, Xiaoyan Zhang, Zhirong Liu, Zhenhua Dong, and Xiuqiang He. Leveraging title-abstract attentive semantics for paper recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):67–74, Apr. 2020.

[13] Haofeng Jia and Erik Saule. Local is good: A fast citation recommendation approach. In *Lecture Notes in Computer Science*, pages 758–764. Springer International Publishing, 2018.

[14] Jianshan Sun, Yuanchun Jiang, Xusen Cheng, Wei Du, Yezheng Liu, and Jian Ma. A hybrid approach for article recommendation in research social networks. *Journal of Information Science*, 44(5):696–711, sep 2017.

[15] Ritu Sharma, Dinesh Gopalani, and Yogesh Meena. Concept-based approach for research paper recommendation. In *Lecture Notes in Computer Science*, pages 687–692. Springer International Publishing, 2017.

[16] Christin Katharina Kreutz and Ralf Schenkel. Scientific paper recommendation systems: a literature review of recent publications. *arXiv preprint arXiv:2201.00682*, 2022.

[17] Maya Hristakeva, Daniel Kershaw, Marco Rossetti, Petr Knoth, Benjamin Pettit, Saúl Vargas, and Kris Jack. Building recommender systems for scholarly information. In *Proceedings of the 1st Workshop on Scholarly Web Mining - SWM '17*. ACM Press, 2017.

[18] Frederick Ayala-Gómez, Bálint Daróczy, András Benczúr, Michael Mathioudakis, and Aristides Gionis. Global citation recommendation using knowledge graphs. *Journal of Intelligent Fuzzy Systems*, 34(5):3089–3100, may 2018.

[19] Zafar Ali, Guilin Qi, Khan Muhammad, Bahadar Ali, and Waheed Ahmed Abro. Paper recommendation based on heterogeneous network embedding. *Knowledge-Based Systems*, 210:106438, dec 2020.

[20] Maha Amami, Rim Faiz, Fabio Stella, and Gabriella Pasi. A graph based approach to scientific paper recommendation. In *Proceedings of the International Conference on Web Intelligence*. ACM, aug 2017.

[21] Nazmus Sakib, Rodina Binti Ahmad, and Khalid Haruna. A collaborative approach toward scientific paper recommendation using citation context. *IEEE Access*, 8:51246–51255, 2020.

[22] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

[23] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[24] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. *arXiv preprint arXiv:1301.7363*, 2013.

[25] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, 2008.

[26] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46:109–132, 2013.

[27] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment*, 4(11):992–1003, 2011.

[28] Yixiang Fang, Yixing Yang, Wenjie Zhang, Xuemin Lin, and Xin Cao. Effective and efficient community search over large heterogeneous information networks. *Proceedings of the VLDB Endowment*, 13(6):854–867, 2020.

[29] Yixiang Fang, Kai Wang, Xuemin Lin, and Wenjie Zhang. Cohesive subgraph search over big heterogeneous information networks: Applications, challenges, and solutions. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2829–2838, 2021.

[30] Yizhou Sun and Jiawei Han. Mining heterogeneous information networks: a structural analysis approach. *Acm Sigkdd Explorations Newsletter*, 14(2):20–28, 2013.

[31] Wei Shen, Jiawei Han, and Jianyong Wang. A probabilistic model for linking named entities in web text with heterogeneous information networks. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1199–1210, 2014.

[32] Xun Jian, Yue Wang, and Lei Chen. Effective and efficient relational community detection and search in large dynamic heterogeneous information networks. *Proceedings of the VLDB Endowment*, 13(10):1723–1736, 2020.

[33] Hsi-Wen Chen, Hong-Han Shuai, De-Nian Yang, Wang-Chien Lee, Chuan Shi, S Yu Philip, and Ming-Syan Chen. Structure-aware parameter-free group query via heterogeneous information network transformer. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2075–2080. IEEE, 2021.

[34] Xiang Li, Danhao Ding, Ben Kao, Yizhou Sun, and Nikos Mamoulis. Leveraging meta-path contexts for classification in heterogeneous information networks. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 912–923. IEEE, 2021.

[35] Mohamad Yaser Jaradeh, Allard Oelen, Kheir Eddine Farfar, Manuel Prinz, Jennifer D'Souza, Gábor Kismihók, Markus Stocker, and Sören Auer. Open research knowledge graph: next generation infrastructure for semantic scholarly knowledge. In *Proceedings of the 10th International Conference on Knowledge Capture*, pages 243–246, 2019.

[36] Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, et al. Construction of the literature graph in semantic scholar. *arXiv preprint arXiv:1805.02262*, 2018.

[37] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Hsu, and Kuansan Wang. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th international conference on world wide web*, pages 243–246, 2015.

[38] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1):396–413, 2020.

[39] Paolo Manghi, Claudio Atzori, Alessia Bardi, Jochen Schirrwagen, Harry Dimitropoulos, Sandro La Bruzzo, Ioannis Foufoulas, Aenne Löhden, Amelie Bäcker, Andrea Mannocci, et al. Openaire research graph dump. *Zenodo*, 2019.

[40] Paolo Manghi, Alessia Bardi, Claudio Atzori, Miriam Baglioni, Natalia Manola, Jochen Schirrwagen, Pedro Principe, Michele Artini, Amelie Becker, Michele De Bonis, et al. The openaire research graph data model. *Zenodo*, 2019.

[41] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 990–998, 2008.

[42] Yun Xiong, Yangyong Zhu, and S Yu Philip. Top-k similarity join in heterogeneous information networks. *IEEE Transactions on Knowledge and Data Engineering*, 27(6):1710–1723, 2014.

[43] Pijitra Jomsri, Siripun Sanguansintukul, and Worasit Choochaiwattana. A framework for tag-based research paper recommender system: an ir approach. In *2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*, pages 103–108. IEEE, 2010.

[44] Abdul Shahid, Muhammad Tanvir Afzal, Moloud Abdar, Mohammad Ehsan Basiri, Xujuan Zhou, Neil Y Yen, and Jia-Wei Chang. Insights into relevant knowledge extraction techniques: a comprehensive review. *The Journal of Supercomputing*, 76(3):1695–1733, 2020.

[45] Serafeim Chatzopoulos, Thanasis Vergoulis, Theodore Dalamagas, and Christos Tryfonopoulos. Veto+: improved expert set expansion in academia. *International Journal on Digital Libraries*, pages 1–19, 2021.

[46] Neal Robert Haddaway, Alexandra Mary Collins, Deborah Coughlin, and Stuart Kirk. The role of google scholar in evidence reviews and its applicability to grey literature searching. *PloS one*, 10(9):e0138237, 2015.

[47] Akhtar Rasool, Amrita Tiwari, Gunjan Singla, and Nilay Khare. String matching methodologies: A comparative analysis. *REM (Text)*, 234567(11):3, 2012.

[48] Andrzej Białecki, Robert Muir, Grant Ingersoll, and Lucid Imagination. Apache lucene 4. In *SIGIR 2012 workshop on open source information retrieval*, page 17, 2012.

[49] Peter D Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188, 2010.

[50] K Sparck Jones, Steve Walker, and Stephen E. Robertson. A probabilistic model of information retrieval: development and comparative experiments: Part 2. *Information processing & management*, 36(6):809–840, 2000.

[51] Ciprian-Octavian Truica, Alexandru Boicea, and Florin Radulescu. Building an inverted index at the dbms layer for fast full text search. *Journal of Control Engineering and Applied Informatics*, 19(1):94–101, 2017.

[52] Thanasis Vergoulis, Serafeim Chatzopoulos, Ilias Kanellos, Panagiotis Deligiannis, Christos Tryfonopoulos, and Theodore Dalamagas. Bip! finder: Facilitating

scientific literature search by exploiting impact-based ranking. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, pages 2937–2940, New York, NY, USA, 2019. ACM.

[53] Hebatallah A. Mohamed Hassan. Personalized research paper recommendation using deep learning. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*. ACM, jul 2017.

[54] Joeran Beel, Akiko Aizawa, Corinna Breitinger, and Bela Gipp. Mr. DLib: Recommendations-as-a-service (RaaS) for academia. In *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. IEEE, jun 2017.

[55] Chifumi Nishioka, Jörn Hauke, and Ansgar Scherp. Research paper recommender system with serendipity using tweets vs. diversification. In *Digital Libraries at the Crossroads of Digital Information for the Future*, pages 63–70. Springer International Publishing, 2019.

[56] Shahbaz Ahmad and Muhammad Tanveer Afzal. Combining co-citation and metadata for recommending more related papers. In *2017 International Conference on Frontiers of Information Technology (FIT)*. IEEE, dec 2017.

[57] Pablo Figueira, Fabiano Belem, Jussara M. Almeida, and Marcos A. Goncalves. Automatic generation of initial reading lists: Requirements and solutions. In *2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. IEEE, jun 2019.

[58] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[59] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

[60] André Vellino. Usage-based vs. citation-based methods for recommending scholarly research articles. *arXiv preprint arXiv:1303.7149*, 2013.

[61] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, pages 89–98, 1998.

[62] Jian Wu, Kunho Kim, and C Lee Giles. Citeseerx: 20 years of service to scholarly big data. In *Proceedings of the Conference on Artificial Intelligence for Data Discovery and Reuse*, pages 1–4, 2019.

[63] Kazunari Sugiyama and Min-Yen Kan. Exploiting potential citation papers in scholarly paper recommendation. In *Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries*, pages 153–162, 2013.

[64] Kazunari Sugiyama and Min-Yen Kan. A comprehensive evaluation of scholarly paper recommendation using potential citation papers. *International Journal on Digital Libraries*, 16(2):91–109, 2015.

[65] Daniel Hienert, Frank Sawitzki, and Philipp Mayr. Digital library research in action–supporting information retrieval in sowiport. *D-Lib Magazine*, 21(3/4):2015, 2015.

[66] Joeran Beel, Siddharth Dinesh, Philipp Mayr, Zeljko Carevic, and Jain Raghvendra. *Stereotype and most-popular recommendations in the digital library sowiport.* Humboldt-Universität zu Berlin, 2017.

[67] Michael Ley. Dblp: some lessons learned. *Proceedings of the VLDB Endowment*, 2(2):1493–1500, 2009.

[68] Toine Bogers and Antal Van den Bosch. Recommending scientific articles using citeulike. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 287–290, 2008.

[69] Hao Wang, Binyi Chen, and Wu-Jun Li. Collaborative topic regression with social regularization for tag recommendation. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[70] Joeran Beel and Stefan Langer. A comparison of offline evaluations, online evaluations, and user studies in the context of research-paper recommender systems. In *International conference on theory and practice of digital libraries*, pages 153–168. Springer, 2015.

[71] Thanasis Vergoulis, Serafeim Chatzopoulos, Theodore Dalamagas, and Christos Tryfonopoulos. Veto: Expert set expansion in academia. In *International*

Conference on Theory and Practice of Digital Libraries, pages 48–61. Springer, 2020.

[72] Serafeim Chatzopoulos, Thanasis Vergoulis, Theodore Dalamagas, and Christos Tryfonopoulos. Veto-web: A recommendation tool for the expansion of sets of scholars. In *2021 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 334–335. IEEE, 2021.

[73] Peter Emerson. The original borda count and partial voting. *Social Choice and Welfare*, 40(2):353–358, 2013.

[74] Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 758–759, 2009.

[75] Edward A Fox and Joseph A Shaw. Combination of multiple searches. *NIST special publication SP*, 243, 1994.

[76] Steven M Beitzel, Eric C Jensen, Abdur Chowdhury, David Grossman, Ophir Frieder, and Nazli Goharian. Fusion of effective retrieval strategies in the same information retrieval system. *Journal of the American Society for Information Science and Technology*, 55(10):859–868, 2004.

[77] David Lillis, Fergus Toolan, Rem Collier, and John Dunnion. Extending probabilistic data fusion using sliding windows. In *European Conference on Information Retrieval*, pages 358–369. Springer, 2008.

[78] David Lillis, Lusheng Zhang, Fergus Toolan, Rem W Collier, David Leonard, and John Dunnion. Estimating probabilities for effective data fusion. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 347–354, 2010.

[79] Serafeim Chatzopoulos, Thanasis Vergoulis, Ilias Kanellos, Theodore Dalamagas, and Christos Tryfonopoulos. Further improvements on estimating the popularity of recently published papers. *Quantitative Science Studies*, 2(4):1529–1550, 2022.

[80] Kalervo Järvelin and Jaana Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *ACM SIGIR Forum*, volume 51, pages 243–250. ACM New York, NY, USA, 2017.