GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Master's Thesis

Bridging Citation Analysis and Language Models: A Hybrid Recommender System for Computer Science Papers

Joel Beck

Institute of Computer Science

Chair for Scientific Information Analytics

26. September 2023

Georg-August-Universität Göttingen
Institute of Computer Science

Goldschmidtstraße 7
37077 Göttingen
Germany

☎ +49 (551) 39-172000
FAX +49 (551) 39-14403
✉ office@informatik.uni-goettingen.de
🌍 www.informatik.uni-goettingen.de

Thesis Advisor:	Dr. Corinna Breitinger
Primary Thesis Examiner:	Prof. Dr. Thomas Kneib
Secondary Thesis Examiner:	Dr. Terry Ruas

I hereby declare that I have written this thesis independently without any help from others and without the use of documents or aids other than those stated. I have mentioned all used sources and cited them correctly according to established academic citation rules.

Göttingen, 26. September 2023

Abstract

With the continuous growth of scientific literature, researchers face the challenge of keeping up with the latest research. Paper recommender systems aim to address this challenge by offering automated and personalized recommendations. However, many existing approaches fall short in terms of accessibility, transparency, and customization, often relying too heavily on either citation-driven or content-driven strategies.

We present a hybrid recommender system for Computer Science papers that blends citation-based and content-based approaches, termed the Citation Recommender and the Language Recommender. The Citation Recommender incorporates global document characteristics, such as publication date, paper citation count, and author citation count, combined with citation analysis techniques like co-citation analysis and bibliographic coupling. By assigning weights to these five elements, users can tailor the system to align with their preferences.

The Language Recommender employs pretrained language models to embed paper abstracts, generating recommendations based on semantic similarity. Users can choose from eight language models encompassing keyword-based sparse embedding models (TF-IDF and BM25), static embedding models (Word2Vec, FastText, GloVe), and contextualized embedding models (BERT, SciBERT, Longformer). In the hybrid approach, one recommender first identifies candidates that are subsequently re-ranked by the other recommender.

We assess our hybrid recommender system through a comprehensive three-stage evaluation. Our results highlight the bibliographic coupling score as the most influential factor for the Citation Recommender, while the standalone paper citation count performs at par with randomly sampled recommendations. For the Language Recommender, SciBERT stands out as the most effective language model. Overall, the Language Recommender dominates the Citation Recommender, leading to more relevant and interdisciplinary recommendations. Initiating the hybrid system with the Language Recommender for candidate selection followed by the Citation Recommender for re-ranking yields better and more stable results than the reverse ordering. However, the best overall performance is achieved using a solely content-driven approach with SciBERT, rather than employing a hybrid model.

This thesis offers insights into paper recommendation systems that combine citation and content strategies. While established techniques like co-citation analysis and bibliographic coupling remain valuable, we advocate for a greater emphasis on content-focused methods, especially those utilizing domain-specific models like SciBERT.

Contents

List of Figures

List of Tables

List of Abbreviations

LLMs	Large Language Models	3
RT	Research Task	5
C $\rightarrow$ **L**	Citation to Language	6
L $\rightarrow$ **C**	Language to Citation	6
CF	Collaborative Filtering	9
CBF	Content-Based Filtering	9
$P@k$	Precision at k	14
$R@k$	Recall at k	14
AP	Average Precision	14
MAP	Mean Average Precision	14
MRR	Mean Reciprocal Rank	14
DCG	Discounted Cumulative Gain	14
NDCG	Normalized Discounted Cumulative Gain	14
CPA	Citation Proximity Analysis	20
CPI	Citation Proximity Index	20
NLP	Natural Language Processing	23
OOV	Out-of-Vocabulary	25
BoW	Bag-of-Words	25
CBOW	Continuous Bag-of-Words	29
WSD	Word Sense Disambiguation	32
MLM	Masked Language Modeling	35
NSP	Next Sentence Prediction	35

Chapter 1

Introduction

This thesis addresses the need for a paper recommender system that can reliably recommend relevant research papers customized to the user's preferences. The paper recommender system is designed to be used by academics, who seek to stay up-to-date with the latest research in their field. The system is based on a hybrid architecture, which combines the strengths of citation-based and content-based methods.

Section 1.1 introduces the problem of finding relevant research papers in an ever-growing body of scientific literature and identifies the research gap of hybrid paper recommender systems employing citation-based and content-based approaches. Section 1.2 presents the research objectives and the specific research tasks this thesis aims to address. Lastly, Section 1.3 outlines the structure of the thesis.

1.1 Problem Statement & Motivation

Identifying relevant research papers is an ongoing and time-consuming task for academics. This challenge is particularly pronounced for those new to a field, as they must familiarize themselves with its current research landscape. However, even seasoned researchers encounter difficulties in staying updated with the latest findings due to the rapidly increasing volume of publications. Various sources indicate that between 2.5 million [1] and 3 million [2] new research papers are published each year. Khabsa et al.'s 2014 study [3] suggests there are nearly 25 million research papers available online, with an annual growth rate ranging from 4% [2] to as high as 9% [4]. This substantial growth complicates the task of pinpointing relevant articles.

To mitigate this issue, recommender systems have been developed to suggest pertinent papers to scholars. Yet, many researchers still gravitate towards manual search techniques, such as keyword search in digital libraries, traversing paper citation graphs, ranking papers by non-personalized metrics like citation count or recency, and even relying on recommendations from colleagues. The

subsequent sections detail the shortcomings of traditional approaches to academic paper search, underscoring the need for a more effective solution.

Search Engines

Search engines are a popular method to find research papers. In this approach, keywords provided by the user are matched to papers in the search engine's database. General-purpose search engines like Google[1] or Bing[2] offer several advantages: they are fast, widely accessible, user-friendly, free, and multilingual. However, they aren't tailored for academic paper searches, leading to a mix of academic and non-academic content in the results. Furthermore, users lack control over the presentation order of search results and less popular papers often remain unlisted.

Specialized academic search engines like Google Scholar[3], Semantic Scholar[4], IEEE Xplore[5], and ACM Digital Library[6] cater more to researchers' needs. They index only academic papers and offer enhanced search capabilities, such as filters for publication date, author, or venue. Yet, even these engines have shortcomings. Their ranking customization options are limited. For instance, Google Scholar primarily offers broad categories like *date* and *relevance*. Although Semantic Scholar provides a more granular set of filters and sorting options, including *citation count* and *most influential*, the mechanics of their ranking algorithms including the features that impact the relevance metric remain opaque.

Additionally, these platforms typically present rankings without a quantifiable relevance score, preventing users from gauging the relative pertinence of papers or determining how closely search results align with their queries in absolute terms.

Following Citations

Another strategy for identifying relevant papers is to trace the citations of a paper that the user considers valuable. This method is based on the premise that papers that cite each other are thematically connected. Figure 1.1 depicts this process with an example. The referenced paragraph contains links to two other citations, suggesting that the cited papers are related to the citing paper's topic and might be of interest to the reader.

Navigating the citation graph offers several advantages over search engines. It's a straightforward, intuitive method and personalizes recommendations based on the user's existing interests. Moreover, the citation context provides clues about the scope and relevance of the cited papers.

Yet, this approach is not without drawbacks. First, it is time-intensive, as users must manually select

[1] https://www.google.com/
[2] https://www.bing.com/
[3] https://scholar.google.com/
[4] https://www.semanticscholar.org/
[5] https://ieeexplore.ieee.org/
[6] https://dl.acm.org/

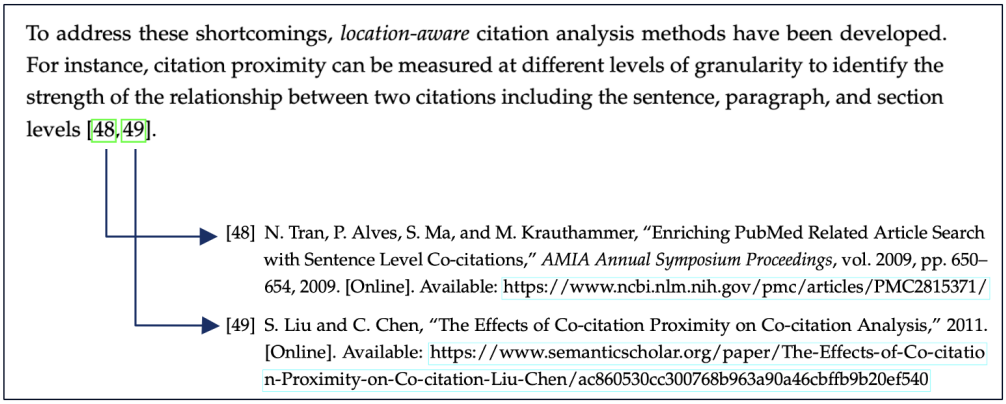


Figure 1.1: Following citations of one paper to find related papers. The cited papers are assumed to be related to the topic of the original paper and might therefore be relevant recommendations for further reading. The citation context provides clues about the scope and relevance of the cited papers.

citations, evaluate their relevance, and acquire access to the cited papers. Second, it necessitates a starting point, implying users must already be somewhat familiar with the research area. Third, it is not scalable, as users can only follow a limited number of citations before losing track of the original paper's topic. Fourth, the technique does not guarantee comprehensive coverage of the research landscape and relevant papers might be missed. Finally, users strongly depend on the authors' citation practices, as they might omit relevant citations or include citations that are not thematically related to the paper.

AI Tools

With OpenAI's launch of ChatGPT in November 2022[7], several AI tools emerged that are capable of recommending research papers. Built upon Large Language Models (LLMs) like GPT-3 [5] or GPT-4 [6], they offer a chat interface, allowing users to query the model in natural language. In theory, these chatbots can offer highly tailored recommendations, letting users articulate their preferences and iteratively refine results. Further, the model can provide immediate context to each suggestion, such as a paper summary.

Despite their potential, AI-based recommendation systems have significant limitations. Their primary issue is reliability - recommendations are not reproducible, users have no control over the model's responses, and there is a risk of missing relevant papers. Furthermore, without web access, these tools can only recommend papers up to their last training data update, potentially leading to outdated suggestions. Lastly, many current offerings are proprietary to some extent and might require a paid subscription to access their full functionality.

[7] https://openai.com/blog/chatgpt

Replicating Research Approaches

An alternate approach to finding relevant papers is to emulate research methodologies from the academic literature. This involves studying papers on existing recommender systems, pinpointing approaches suitable for the user's domain, and replicating the described methodologies.

This method combines the advantages and disadvantages of prior strategies. Recommendations are personalized, as researchers choose techniques aligned with their objectives. The ranking algorithm is transparent, given that methodologies are often thoroughly outlined in the paper. Finally, interacting with the latest research in the field gives insight into state-of-the-art recommendation techniques that might provide more accurate recommendations than solutions accessible to a broader audience.

However, this approach also poses challenges. First, reading and understanding the papers demand a significant time investment, while replicating the methodologies - including model training - often require substantial, or even infeasible, computational resources. Second, proposed techniques might be inaccessible to readers if the code or data is not public, if the paper omits crucial implementation details, or if the method is tailored for a specific scenario and cannot be generalized.

Shortage of Hybrid Paper Recommenders

Much of the current research on paper recommendation focuses on singular techniques, such as either citation-based or content-based methods, and lacks the flexibility to combine different approaches.

Citation-based methods [7–11] rely on the citation network to determine a paper's importance. Such methods provide insights into a paper's perceived relevance within its community. However, a limitation of this approach is its potential bias towards well-cited papers. This can lead to overlooking newer or less cited research that might offer innovative perspectives. This bias indicates that relying solely on citation metrics can miss out on recognizing emerging areas of study.

Conversely, content-based methods [12–17] focus on the textual content of documents to determine their relevance. These systems align recommendations with distinct research themes. However, like citation-based techniques, they pose challenges. A primary concern is their tendency to offer a narrow spectrum of recommendations. While adept at identifying papers that match a user's specific query, they might overlook related works from other disciplines that could provide valuable insights or additional context.

Considering these strengths and shortcomings, there's a compelling case for hybrid recommender systems that synergize citation-based and content-based approaches.

Contribution of this Thesis

This thesis presents a hybrid recommender system designed for Computer Science research that seeks to address many limitations of existing systems. Available as free open-source software with an easy-to-use interface and detailed documentation, the system delivers automated and reproducible recommendations with transparent ranking criteria. The hybrid structure combines global document characteristics, such as popularity and recency, with both citation and content-driven methods. It offers users the flexibility to adjust the influence of each method for customized recommendations. Additionally, the system enriches rankings with a relevance score, offering researchers insights into the relative importance of each recommendation.

1.2 Research Objectives

Driven by the increasing need for an automated, fast, and reliable academic paper recommendation system with high coverage, and recognizing the potential of both citation-based and content-based methods, we have set the following research objective:

Design, implement, and evaluate a hybrid recommender system that - given a query paper - integrates citation-based and content-based techniques to suggest relevant and personalized academic papers to researchers.

To achieve this objective Research Task (RT) 1 - Research Task 4 are derived:

RT 1: Review and analyze the existing literature and methodologies related to citation-based and content-based paper recommendation systems, and identify effective ways to merge them into a hybrid system.

RT 2: Develop and implement a performant, accessible and extensible framework that offers personalized paper recommendations using a blend of citation-based and content-based methods.

RT 3: Identify the best-performing configuration for each individual recommender including the most effective feature weighting for the citation-based recommender and the top-performing language model for the content-based recommender.

RT 4: Determine whether the hybrid system gives more accurate recommendations than the citation-based or content-based recommenders alone. Examine how the order of combining the methods influences the hybrid recommender's precision and diversity.

1.3 Thesis Outline

This thesis is structured into five chapters.

Chapter 1 introduces and motivates the problem of finding relevant research papers in a growing body of scientific literature, highlighting the need for an automated, reliable and precise paper recommender system (Section 1.1). It then presents the research objective and its derived research tasks (Section 1.2) and outlines the structure of the thesis (Section 1.3).

Chapter 2 reviews the existing literature on paper recommender systems and provides foundational knowledge for the subsequent chapters. It begins with a broad overview of general recommender systems, detailing their different types and common evaluation strategies (Section 2.1). Following this, Section 2.2 explores how citation-based methods are used in recommender systems. Section 2.3 then discusses content-based approaches, with a particular focus on language models that use the semantic information of documents. Lastly, Section 2.4 summarizes the current work on hybrid recommender systems for academic papers and highlights this thesis's contributions.

Chapter 3 describes the methodology used to develop the hybrid recommender system. First, a conceptual overview of the system is given (Section 3.1). Then, the data sources for this thesis and the dataset construction process are presented (Section 3.2). Next, detailed information on the training or precomputation phase (Section 3.3) and the inference phase (Section 3.4) is provided.

Chapter 4 evaluates the hybrid recommender system. First, Section 4.1 introduces the systematic and data-driven evaluation strategy of this thesis. Then, the results of the evaluation are presented by identifying the bibliographic coupling score as the most influential feature for the Citation Recommender, SciBERT as the most effective language model for the Language Recommender, and the Language to Citation (L $\rightarrow$ C) hybridization order as superior to the Citation to Language (C $\rightarrow$ L) order (Section 4.2).

Finally, Chapter 5 concludes the thesis by summarizing the methodology and findings (Section 5.1). Section 5.2 highlights the contributions of this thesis to the broader field of paper recommender systems, while Section 5.3 points out current limitations and starting points for future work.

Figure 1.2 illustrates the thesis structure graphically.

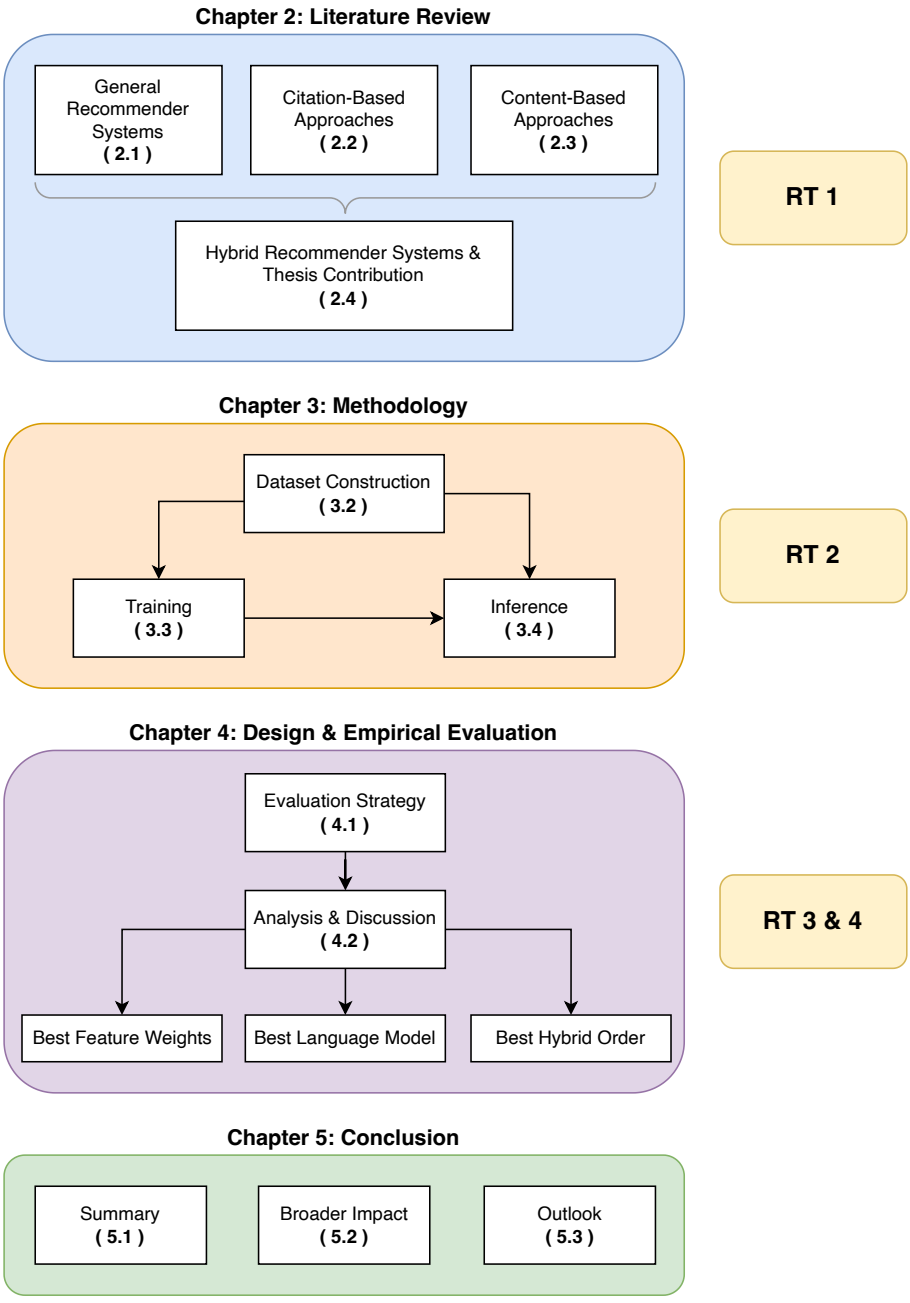


Figure 1.2: Thesis structure visualized. The diagram is adapted from [18]. **RT 1** - **RT 4** derived in Section 1.2 are added to their respective chapters.

Chapter 2

Background & Related Work

This chapter addresses **RT 1** as defined in Section 1.2. It presents a detailed overview of the existing literature on recommender systems for academic papers. Thereby, it introduces key concepts such as citation-based and content-based methods for recommendation, as well as hybrid recommender systems.

Section 2.1 begins with a broad overview of general recommender systems, detailing their different types and how they are evaluated. Following this, Section 2.2 explores how citation-based methods are used in recommender systems. Section 2.3 then discusses content-based approaches, with a particular focus on language models that use the semantic information of documents. Lastly, Section 2.4 summarizes the current work on hybrid recommender systems for academic papers, and highlights how this thesis contributes to the field.

2.1 Recommender Systems

Recommender systems are the subject of extensive research and have found wide-ranging applications in various domains, from e-commerce to entertainment and beyond [19,20]. The fundamental objective of these systems is to predict preferences by learning from the past behavior of a user and the behavioral patterns of similar users. In comparison to traditional keyword-centric search engines, recommender systems have proven to be more effective and reliable in drawing meaningful knowledge from large datasets [21].

Types of Recommender Systems

Recommender systems can be classified into several categories based on their underlying algorithms and methodologies. The primary types encompass Collaborative Filtering (CF), Content-Based Filtering (CBF), and hybrid systems. Content-based techniques propose items that are similar to the ones a user has previously expressed interest in, while collaborative techniques

recommend items that are favored by users with similar preferences [22].

According to a comprehensive survey conducted by Beel et al. [23], CBF is the most commonly applied approach in research paper recommender systems, used in 55% of the surveyed articles, while CF was utilized in only 18% of the articles reviewed. The following sections introduce CF, CBF, and hybrid systems in more detail.

2.1.1 Collaborative Filtering

The underlying assumption of Collaborative Filtering [24,25] is that those who agreed in the past will agree in the future, and that preferences for one user can be inferred from the preferences of similar users [21]. The conventional approach of CF is user-oriented, whereby the system identifies users whose rating patterns are similar to those of the active user and recommends items that these like-minded users have shown interest in [19]. Despite its efficacy, this method encounters scalability issues as the user base expands. Additionally, it is vulnerable to fluctuations in user behavior over time [26].

Sarwar et al. [26] suggest item-based CF to address the challenge of scalability. This method operates by comparing items as opposed to users. It identifies correlations between different items based on the co-occurrence of ratings across users. Once these relationships have been established, the system can generate recommendations for a user based on their previous ratings of items. This method offers better scalability as the growth rate of items typically lags behind that of users, and it is also more robust to change since items do not exhibit changes in their behavior over time as it is the case with users.

Agarwal et al. [16] propose a clustering approach to CF to handle sparse, high-dimensional data. Their method, called Subspace Clustering Based Analysis (SCuBA), maps users to specific subspaces where similar browsing behavior is identified. Each subspace represents a specific user behavior, and users are assigned to the subspace that best matches their behavior. Recommendations are made based on popular items in the user's assigned subspace.

CF has been effectively applied in many practical scenarios and fundamentally contributes to the success of many high-profile companies. For instance, the Netflix Prize competition was designed to enhance the company's CF algorithm [27]. CF is also employed by YouTube to recommend videos [28,29] and by Amazon to suggest products [30].

2.1.2 Content-Based Filtering

Content-Based Filtering methods utilize item features to generate recommendations. These methods propose items that are similar to the ones the user has previously shown interest in [19]. The primary assumption in CBF is that users will be attracted to items that resemble those they have previously liked [21].

Similar to CF, CBF also adopts two main forms: user-based and item-based [22]. User-based CBF recommends items based on the user's profile and their past behavior. Conversely, item-based CBF generates recommendations based on similarities between items [31]. For example, a user-based CBF system for academic papers may recommend a paper based on the user's previous ratings of similar papers. An item-based CBF system, on the other hand, may recommend a paper based on the paper's similarity in topics, authors, or citations to other papers the user has previously rated.

Comparison of Collaborative Filtering and Content-Based Filtering

Both CF and CBF have their strengths and weaknesses. By focusing on user-similarity rather than item-similarity, CF can offer serendipitous recommendations [18]. For instance, a user may be recommended a paper outside his research scope but which is highly rated by other users with similar interests. However, CF is hindered by the cold-start problem, where new users or new items have no ratings and thus cannot be recommended [32]. One approach to addressing the cold-start problem is to use *implicit* feedback, such as the number of times a user has viewed an item [33] or the number of pages a user has read from a document as a proxy for the user's preference [34].

Conversely, CBF allows for user-based personalization since it relies on the user's profile and past behavior. It is also not affected by the cold-start problem since it can recommend items based on their features alone [32]. Further, CBF requires less preliminary categorization work [18] since user models can be created automatically. However, CBF is highly dependent on the quality of the item features. For instance, if the item features are not sufficiently descriptive, the system may not be able to generate accurate recommendations. Further, CBF tends to propose items that are similar to those the user has already rated, potentially limiting the diversity of the recommendations [22, 32].

Ricci et al. [19] suggest that CBF is preferable when there's sufficient information about the items but less information about the users, whereas CF is more useful in situations with abundant user information but limited item details.

2.1.3 Hybridization Techniques

Hybrid recommender systems that combine CF and CBF can offer more accurate and diverse recommendations than either CF or CBF alone by addressing the limitations of each approach. They can better address the cold-start problem than pure CF systems and offer more diverse recommendations than pure CBF systems [22].

Hybrid recommenders can be designed in several ways. Burke [22] presents a classification of hybrid recommendation techniques, categorizing them into seven types: weighted, mixed, switching, feature combination, feature augmentation, cascade, and meta-level. These techniques can be further classified into two categories: In *weak* hybridization techniques, the primary technique remains dominant whereas in *true* hybridization techniques, the two techniques equally contribute

to the final recommendations [18,22,35]. CBF approaches commonly rely on the feature augmentation technique by using global relevance attributes such as citation counts, the article recency or the number of affiliations to rank candidate recommendations [18].

This thesis focuses on weak form of hybridization: the *cascade* pattern. The cascade strategy uses a sequential approach: The first recommender generates a candidate list of recommendations, while the second recommender refines and re-ranks these recommendations [22, 36] as illustrated in Figure 2.1. This method contrasts with parallel approaches, such as the weighted hybrid strategy, where both recommenders contribute to the final recommendation ranking and the impact of each recommender is controlled by a weighting factor [22].

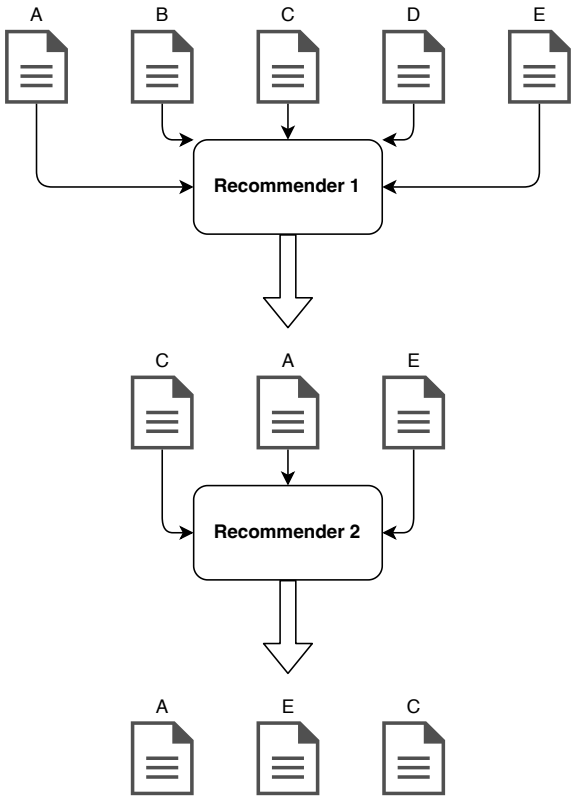


Figure 2.1: The cascade hybridization strategy. Papers A - E represent candidate papers from the dataset. Recommender 1 generates a sorted candidate list of recommendations with Paper C as the top recommendation. The second recommender then refines and re-ranks the candidate list, resulting in Paper A as the top recommendation.

The primary motivation for using the cascade strategy in this thesis is its effect on the computational complexity of the hybrid system. In contrast to parallel techniques where both recommenders process the entire dataset, the cascade approach can significantly reduce the computational requirements by only requiring the first recommender to process the whole dataset. The second

recommender only needs to process the candidate list generated by the first recommender, which is typically much smaller in size. Therefore, the size of the candidate list can be seen as a hyperparameter with two responsibilities:

1. It determines the computational effort for the second recommender. The larger the candidate list, the more documents have to be processed and re-ranked by the second recommender.

2. It controls the impact of each recommender on the final recommendation ranking. If the candidate list is very large, the first recommender only excluded a small number of documents and the hybrid system heavily relies on the second recommender. Conversely, if the candidate list is very small, the first recommender already significantly reduced the set of potential recommendations and the second recommender has less impact on the final recommendation ranking.

2.1.4 Academic Paper Recommendation

In the context of academic papers, *users* can be conceptualized as researchers and *items* as papers. However, the direct application of CF to academic papers presents challenges due to the sparsity of the user-item matrix and the dynamic nature of user interests.

For academic papers, the user-item matrix which denotes interactions between users and items is frequently sparse. Unlike in broader fields such as movies or books, where a large number of individuals may interact with the same item, the audience for academic papers is typically more specific and smaller [16]. This leads to fewer overlaps in the papers that researchers read, thereby making it difficult to form user groups based on shared interests - a fundamental element of CF. Alluding to the cold-start problem, new papers that have not been rated by any researchers cannot be recommended [18]. Further, the continual change of academics in their research focus and reading patterns complicates the use of CF in academic paper recommendation [37].

When comparing the effectiveness of CF and CBF in the context of academic paper recommendation, the results are ambiguous [23]. Several studies have reported CBF to outperform CF for paper recommendation, while others have found the opposite to be true. Beel et al. [23] suggest three potential reasons for these inconsistencies: limitations in the evaluations, a lack of detailed descriptions of the algorithms used, and different implementations of the same recommendation approaches leading to varying results.

This is where hybrid recommender systems have shown promise. By merging CF and CBF, hybrid systems can deliver more robust and flexible recommendations for academic papers [8, 21]. For instance, Kanakia et al. [38] combine co-citation analysis and content-based approaches in their system, leveraging the strengths of both methods. In the co-citation approach, papers that are often cited together are assumed to be similar. In the content-based approach, the text content of papers is analyzed to find similar papers. The hybridization enables the system to balance paper novelty

(discovering new and lesser-known papers) against authority (recommending highly-cited and influential papers).

Section 2.2 and Section 2.3 delve deeper into the two primary approaches employed in academic paper recommendation: citation analysis and semantic analysis. Section 2.4 then explores the application of hybrid recommender systems to paper recommendation.

2.1.5 Evaluation of Recommender Systems

Types of Evaluation

Evaluating the performance of recommender systems is an essential step in their development and deployment. The three prevalent types of evaluations include offline evaluations, online evaluations, and user studies [39]. Offline evaluations involve assessing the system's performance using pre-gathered data, while online evaluations test the system in a live setting with actual users capturing real-time user behavior. User studies, on the other hand, involve detailed analyses of user interactions with the system, often involving user feedback and surveys. In this way, they can provide insights into user preferences and experiences.

However, the outcomes from these evaluation methodologies may not always align. For instance, a system that performs well in offline evaluations may not deliver the same performance in online evaluations or user studies [39]. Offline evaluations are based on historical data and may not accurately reflect current or future user behavior. Moreover, offline evaluations often rely on implicit feedback (e.g., click-through rates), which may not fully capture user satisfaction or user preferences.

Despite these limitations, offline evaluations remain popular as they can be used to compare different approaches quickly and inexpensively. Yet, for a more accurate and comprehensive understanding of a recommender system's performance, Beel and Langer [39] suggest that offline evaluations should be complemented with online evaluations and user studies.

Evaluation Metrics

Accuracy-based metrics are popular due to their straightforward quantification and interpretation. These include the Root Mean Squared Error, Precision, Recall, F1 score, and AUC [19, 21, 40].

However, there are variations of these metrics that are more suited to evaluating recommender systems, including Precision at k ($P@k$), Recall at k ($R@k$), Average Precision (AP), Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), Discounted Cumulative Gain (DCG), and Normalized Discounted Cumulative Gain (NDCG) [40].

$P@k$ is computed as the number of relevant items in the top-k recommendations divided by k [21]. Mathematically, this is expressed as:

$$P@k = \frac{1}{k}\sum_{i=1}^{k} \mathbb{1}(i \text{ is relevant}) = \frac{\#\text{ of recommended relevant items}}{k} \tag{2.1}$$

where $\mathbb{1}(i \text{ is relevant})$ is an indicator function that equals 1 if the i^{th} recommendation is relevant and 0 otherwise. $P@k$ is a suitable metric when the aim is to recommend a few high-quality items to the user, regardless of their ranking in the recommendation list.

On the other hand, $R@k$ is the proportion of the relevant items that are included in the top-k recommendations [21]. This is calculated as:

$$R@k = \frac{1}{m}\sum_{i=1}^{k} \mathbb{1}(i \text{ is relevant}) = \frac{\#\text{ of recommended relevant items}}{m} \tag{2.2}$$

where m is the total number of relevant items. $R@k$ is a more suited metric when the aim is to identify as many relevant items as possible in the top-k recommendations, regardless of their position and the total number of recommendations.

MAP is an extension of the Precision metric that takes into account the order of the recommendations and is more suited for systems where ranking is crucial [21, 41, 42]. Like $P@k$, MAP requires a binary classification of the recommendations as relevant or irrelevant.

For a set of queries, the MAP is the mean of the AP. The AP for a single query is calculated as:

$$AP = \frac{1}{m}\sum_{k=1}^{n} P@k \cdot \mathbb{1}(k \text{ is relevant}) \tag{2.3}$$

where n is the total number of recommendations and m is the total number of relevant items. Then, MAP is computed as:

$$MAP = \frac{1}{Q}\sum_{q=1}^{Q} AP_q \tag{2.4}$$

where Q is the total number of queries and AP_q is the AP for the q^{th} query. AP and MAP are particularly useful when the recommender system is expected to retrieve several relevant items and their ranks are important.

MRR is another metric that considers the position of the relevant items, but only takes the first relevant item into account [21, 42, 43]. MRR is defined as:

$$MRR = \frac{1}{Q} \sum_{q=1}^{Q} \frac{1}{\text{rank}_q} \tag{2.5}$$

where rank_q is the rank of the first relevant item for the q^{th} query. MRR is a suitable metric when the goal of the recommender system is to recommend the most relevant item as early as possible.

DCG and its normalized form NDCG consider the relevance of each item as well as their positions [21, 42, 44]. DCG is calculated as:

$$DCG@k = \sum_{i=1}^{k} \frac{\text{rel}_i}{\log_2(i+1)} \tag{2.6}$$

where rel_i is the relevance of the i^{th} recommendation. The DCG is then normalized by the ideal DCG (IDCG), which is the DCG value for the optimal recommendation list where all items are sorted by their relevance:

$$NDCG@k = \frac{DCG@k}{IDCG@k} \tag{2.7}$$

NDCG is particularly useful when different items have different levels of relevance and they cannot be classified into binary categories (e.g., relevant or irrelevant).

Metrics Beyond Accuracy

While accuracy-based metrics have dominated the evaluation of recommender systems, they overlook other important aspects that influence the quality of a recommender system. These include diversity, serendipity, novelty, and coverage [21, 40, 45]. Diversity refers to the variety in the list of recommendations, serendipity to the element of surprise, novelty to the newness or unfamiliarity of the recommended items, and coverage to the proportion of items or users for which the system can generate recommendations. Despite their significance, there is a scarcity of comprehensive research in these areas, which underscores the need for further exploration how alternative metrics can enhance the effectiveness and user experience of recommender systems [40].

Moreover, Bai et al. [21] identify a growing demand for personalized, context-aware, and cross-domain recommendation systems to cater more effectively to diverse user needs. Roy and Dutta

[37] underscore that recommendations can be significantly improved by considering the context in which they are made. Contextual factors may include the time of day, the user's location, or even the user's mood. Incorporating these factors into recommender systems can help to tailor recommendations to the user's specific circumstances, thus improving the user experience.

In conclusion, there are several challenges in the field of recommender systems that need to be addressed in future research. The most prominent ones include dealing with the cold-start problem, handling the high dimensionality of data, managing the dynamic nature of user interests, and protecting user privacy in recommendation systems [19,21,37,38].

2.2 Citation Analysis

A citation establishes a connection from one document to another, creating a relationship between the document that cites and the one that is cited. The study of these relationships constitutes citation analysis, a central element of bibliometrics [46].

Citation analysis offers a valuable method for assessing the relevance of scientific papers. It can be used to identify related works, track the evolution of research trends, and understand the structure of a knowledge field [47]. This section provides an overview of the different approaches to citation analysis, with a particular focus on bibliographic coupling and co-citation analysis. The section also explores how citation analysis can enhance paper recommender systems.

Citation Count as a Measure of Relevance

A simple approach to utilizing citation information for relevance is to take into account the paper's citation count, that is, the number of times a paper has been cited, as a marker of its relevance. However, Smith [46] points out that this method can be short-sighted as the influence of a paper may not necessarily align with its citation count, since not all citations carry equal significance. Some citations are given out of respect or convention, without the cited papers significantly affecting the citing one. Thus, while citation count offers an insight into a work's impact, it shouldn't stand as the sole metric for determining relevance [46].

Additionally, Marshakova-Shaikevich [47] emphasizes the need for a comprehensive review of all references in documents related to that field, which goes beyond a mere citation count. Finally, citation counts lack sensitivity to specific reference or query documents, rendering them as static for all researchers, with no option for personalization.

Using Connections Between Papers

A refined approach beyond simply counting citations involves considering connections between papers. Related papers can be identified by examining the direct connections between *citing* and

cited papers. Cited papers, or *outgoing links*, are all references found in a paper's bibliography. Conversely, citing papers, or *incoming links*, are all papers that cite the paper of interest.

However, direct connections aren't always the most indicative of relatedness. Very popular papers are commonly cited out of convention, without a strong link to the citing paper's topic. Additionally, some researchers might over-cite their own works or those of close colleagues to boost their citation count, creating a false impression of relatedness.

To address these problems, researchers have turned to methods like bibliographic coupling, co-citation analysis, and citation context analysis, which use indirect connections between papers to reveal associations between works that might not be apparent from direct citations [46]. These methods help create a citation network, a set of documents related through citations, which can be represented as a directed graph. As noted by Marshakova-Shaikevich [47], this graph can be used to identify clusters of closely related documents, track the progression of ideas and research trends, and understand the structure of a knowledge field.

2.2.1 Bibliographic Coupling & Co-Citation Analysis

Throughout this thesis, we distinguish between the terms *citation* and *reference*. Echoing Breitinger [18], we define *citations* as the in-text markers pointing to other research papers, while *references* refer to the list of cited papers at the end of a research paper, commonly termed the bibliography.

Bibliographic coupling assesses the similarity between two articles by counting the references they share. The intuition is that if two articles cite many of the same sources, they likely address similar topics. The degree of similarity between two articles is directly proportional to the number of shared references. Moreover, this method can be employed for grouping papers, revealing the structure and evolution of scientific fields [10, 48].

Conversely, *co-citation analysis* counts how often two articles are jointly cited in other papers. If two articles are frequently co-cited, it's assumed that they relate to each other. The strength of their relationship increases with the frequency of co-citations [10, 49].

The difference between bibliographic coupling and co-citation analysis is visualized in Figure 2.2. Numerically, the co-citation analysis score of an article is bounded by its total number of citations, while the bibliographic coupling score is bounded by its total number of references. For instance, if paper A has 10 citations, the co-citation analysis score with any other paper B can be at most 10. The maximum score is reached when all 10 citing papers of A also cite paper B. Conversely, the maximum bibliographic coupling score is reached when the bibliography of paper A is a subset of the bibliography of paper B.

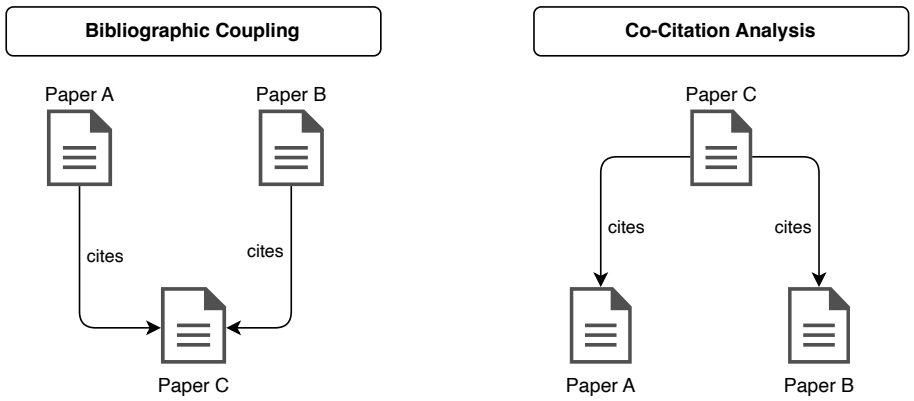


Figure 2.2: Bibliographic Coupling vs. Co-Citation Analysis, adapted from [10]. Left: Bibliographic coupling counts the number of shared references. Paper A and Paper B are connected by bibliographic coupling since they both cite the same Paper C. Right: Co-citation analysis counts the number of shared citing papers. Here, Paper A and Paper B are connected by co-citation analysis since they are both cited by Paper C.

Comparison of Bibliographic Coupling and Co-Citation Analysis

When comparing the two approaches, Small [49] claims that co-citation analysis provides a more dynamic view of a field's intellectual structure, showing not only the relationships between works but also their relative importance and influence. Marshakova-Shaikevich [47] agrees, noting that bibliographic coupling is a static measure that doesn't consider the evolution of a document's relevance over time. She argues that co-citation analysis considers the future of a document through subsequent citations, whereas bibliographic coupling focuses on the document's past through its references.

However, both co-citation analysis and bibliographic coupling have their limitations. By definition, bibliographic coupling might miss relevant papers that don't share many common references with the paper of interest, while co-citation analysis may overlook papers that aren't frequently cited together with other papers. Additionally, bibliographic coupling tends to favor long documents that simply have more space to include references, while co-citation analysis favors older documents that have had more time to collect citations. Notably, both methods can be applied without full-text access, as they only require citation information or a bibliography. However, this implies that they ignore the *location* of the citations within the text, which could provide valuable information about the relationship between the cited documents.

To address these shortcomings, *location-aware* citation analysis methods have been developed. To identify the strength of the relationship between two citations, citation proximity can be measured at different levels of granularity including sentence, paragraph, and section levels [50, 51].

Two popular approaches of location-aware citation analysis are Citation Proximity Analysis and

section-based bibliographic coupling.

2.2.2 Citation Proximity Analysis

Citation Proximity Analysis (CPA) [10] is an improvement over traditional co-citation analysis, remedying some of its limitations. Whereas traditional co-citation analysis only takes into account the frequency of co-citations without considering their context or position within the text, CPA introduces the concept of citation proximity, which refers to the textual distance between citations. It assumes that the closer two citations are to each other, the stronger their relationship is.

One practical application of CPA is in paper recommendation, where it identifies related papers based on their CPA score. Gipp et al. [10] suggest that CPA could be combined with text mining algorithms to analyze the relationship between two references, thereby identifying agreeing or contradicting papers.

Citation Proximity Index

The Citation Proximity Index (CPI) quantifies the proximity score between two citations. Citations that are close to each other are assigned higher CPI scores, while those further apart are assigned lower scores. The distance between citations is not determined by an absolute measure, like the number of characters between them, but is instead based on the document's structure. The highest CPI is assigned to a citation pair within the same sentence (CPI=1), followed by the same paragraph (CPI=$\frac{1}{2}$), the same chapter (CPI=$\frac{1}{4}$), the same journal or book (CPI=$\frac{1}{8}$) and the same journal but different issues (CPI=$\frac{1}{16}$). The scoring is illustrated in Figure 2.3 from Gipp et al. [10].

In cases of multiple citation occurrences, e.g., once within the same sentence and once within the same paragraph, the highest CPI is assigned to that citation pair. The CPI between two *papers* (instead of individual citations) is calculated by aggregating their individual CPI scores across the entire corpus. Gipp et al. [10] found that using a weighted average yields the best results for aggregation.

Limitations of CPA

The authors assert that CPA significantly outperforms traditional co-citation analysis for documents with either many references or few citations. However, CPA has its limitations. First, the authors note that the optimal weighting of the CPI values might depend on the research field or research type. Second, the CPI scores heavily depend on the document's structure. Documents with longer sentences and sections contribute to higher CPI values than those with shorter sentences and sections, as there is a higher probability of citations being within the same segment. Additionally, citation pairs with a large *absolute* distance, e.g., between the first and last sentence of the same chapter, might be assigned a higher CPI value than citation pairs with a small absolute distance, e.g., between the last and first sentence of consecutive chapters.

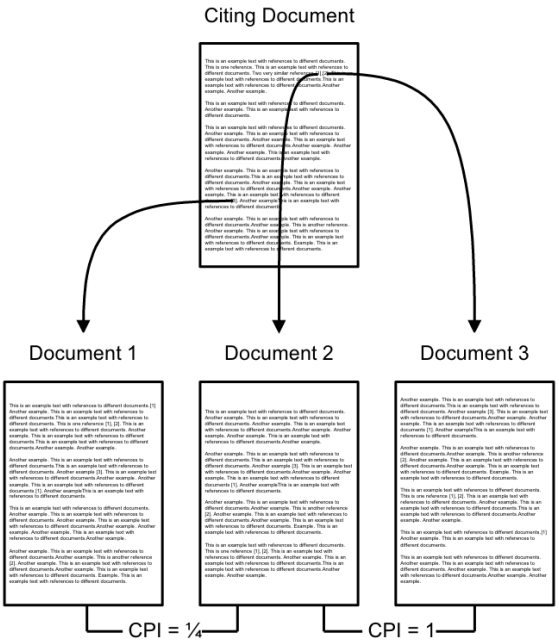


Figure 2.3: Citation Proximity Analysis, illustration from [10]. CPA assigns positive CPI values to document pairs with shared citing papers. The CPI score depends on the proximity of the citations within the citing document. Documents 1 and 2 are co-cited within the same chapter but in different paragraphs of the citing document. Their CPI score is $\frac{1}{4}$. Documents 2 and 3 are co-cited within the same sentence of the citing document; therefore, their CPI score is 1.

Alternative Proximity Measures

To address these shortcomings, Knoth and Khadka [11] introduced alternative proximity measures: normalized CPA (NCPA), inverse CPA (ICPA), and squared inverse CPA (SICPA). NCPA normalizes the proximity values based on the total number of co-citations within a document, whereas ICPA and SICPA assign higher weights to closer co-citations and decrease the weights continuously, rather than discretely, with increasing distance between co-citations. However, even these alternative proximity measures overlook the citation context.

2.2.3 Section-based Bibliographic Coupling

Section-based bibliographic coupling [9] extends the concept of bibliographic coupling by considering the section of a document where a reference is cited. This approach recognizes that a citation in different sections of a paper might have different meanings or implications. For instance, a reference cited in the introduction might provide background information, while the same reference cited in the conclusion could support the author's argument. Habib et al. [9] suggest that references from the *Results* and *Method* sections are more likely to be related to the paper's topic than references from the *Introduction*, *Conclusion*, and *Related Work* sections, which might be more

generic.

Similar to CPA, section-based bibliographic coupling requires access to the full text of the document as it considers the location of citations. The methodology involves first identifying common references between a target paper and a candidate paper. Then, the citing patterns of the common references in different sections of the target and candidate papers are analyzed. Lastly, a section-based bibliographic coupling score is calculated, assigning different weights to citations based on their location within the paper. The Results and Method sections are assigned the largest weights, followed by the Introduction and Conclusion. The Related Work section is assigned the lowest weight [9].

2.2.4 Applications in Paper Recommendation

Citation analysis has found its way into various areas of research. For instance, Gipp et al. [52] use citation analysis to spot plagiarism in academic papers. Unlike conventional character-based similarity measures, they analyze the citation placement within a document to identify semantically similar content, even without textual overlap. According to the authors, even when plagiarists disguise their work through translation, paraphrasing, or idea theft, they tend to retain the original citation pattern, making it a reliable indicator of plagiarism

However, the most prominent application of citation-based approaches is in scientific paper recommendation. Sugiyama and Kan [8] propose to identify "potential citation papers" that a researcher may not have cited in their work, but which are relevant to their research interests, to improve paper recommendation. They use CF techniques to identify these potential citation papers. The authors argue that different logical sections of a paper, such as the introduction, methodology, results, and conclusion, have different significance and could therefore be used to represent papers for recommendation in different ways. However, in contrast to section-based bibliographic coupling [9], their experiments show that assigning the largest weight to the conclusion section rather than the results and methodology sections leads to the best results.

Incorporating citation knowledge into recommender systems can also address the cold-start problem, as shown by Khadka et al. [2]. They use citation knowledge from direct and indirect citations such as co-citation analysis and bibliographic coupling to compute similarity scores between users and papers to generate personalized recommendations. By mixing the citation-based approach with CF, they emphasize that the citation component is essential for recommending *recent* papers that lack user interaction and thus cannot be recommended by CF alone.

Related to paper recommendation, Schwarzer et al. [53] apply CPA to the domain of Wikipedia articles to generate article recommendations by using links. The authors argue that the links between Wikipedia articles play a similar role to citations in academic literature, providing a basis for similarity and thus for recommendations. They demonstrate that citation-based methods are not only successful in the academic literature domain, but can also be effective in the distinct realm

of Wikipedia.

On a meta-level, Ali et al. [54] present a comprehensive taxonomy to categorize citation recommendation models based on their underlying graph structures. The authors argue that graph structures are an appropriate basis for categorization because most citation recommendation models can be viewed as graph-based models. More specifically, these models often operate on graphs where nodes represent documents and edges represent relationships (e.g., citations or co-authorships). The taxonomy distinguishes between four main categories: bibliographic coupling, co-citation, citation context, and hybrid models. This broader perspective on citation methods can facilitate a better understanding of different models and help with selecting suitable models for specific tasks.

2.3 Semantic Analysis

Besides paper metadata and citation information, the text content of a document is another important source of information for recommender systems [17, 21]. This section introduces the concept of embeddings to represent the document's content in numerical form that can be processed by language models. Based on these embeddings, the semantic similarity between documents can be computed, which is a key component of content-based recommender systems.

A *token* in Natural Language Processing (NLP) refers to a piece of a whole, so a token could correspond to a word, a subword, or even punctuation depending on how tokenization is performed. Although *token* is more precise than *word* from a technical perspective, the latter is still used in established terminology such as *Word Embedding*, *Bag-of-Words*, and *Word2Vec*. Therefore, the terms are used interchangeably in this thesis.

2.3.1 Embeddings

A core challenge in NLP is converting the high-dimensional and non-numeric textual data into a format that computers can handle. To address this, various techniques have been developed for transforming, or *embedding*, textual data into numeric vectors. These vectors capture the meaning of the language, making it possible for computational models to process it [55].

Embeddings turn the separate and categorical aspects of text into a continuous, high-dimensional space. This space, known as the *embedding space*, is organized to reflect the semantic and syntactic relationships between words [56]. In the embedding space, words with similar meanings or syntactic features are positioned close together, while dissimilar words are farther apart. This organization allows computational models to understand and process the meaning of text.

Figure 2.4 illustrates an exemplary embedding space projected onto two dimensions. Words with similar meanings are placed closer together. Since the embedding task is inherently unsupervised, the word categories indicated by the different colors are not known in advance. Instead, they

are discovered by the projection of the embedding space itself. Similarly, cluster labels, such as *Recommender Systems*, *Language Models*, or *Metrics* can be added after the fact for visualization purposes, but they are not available to the model during training.

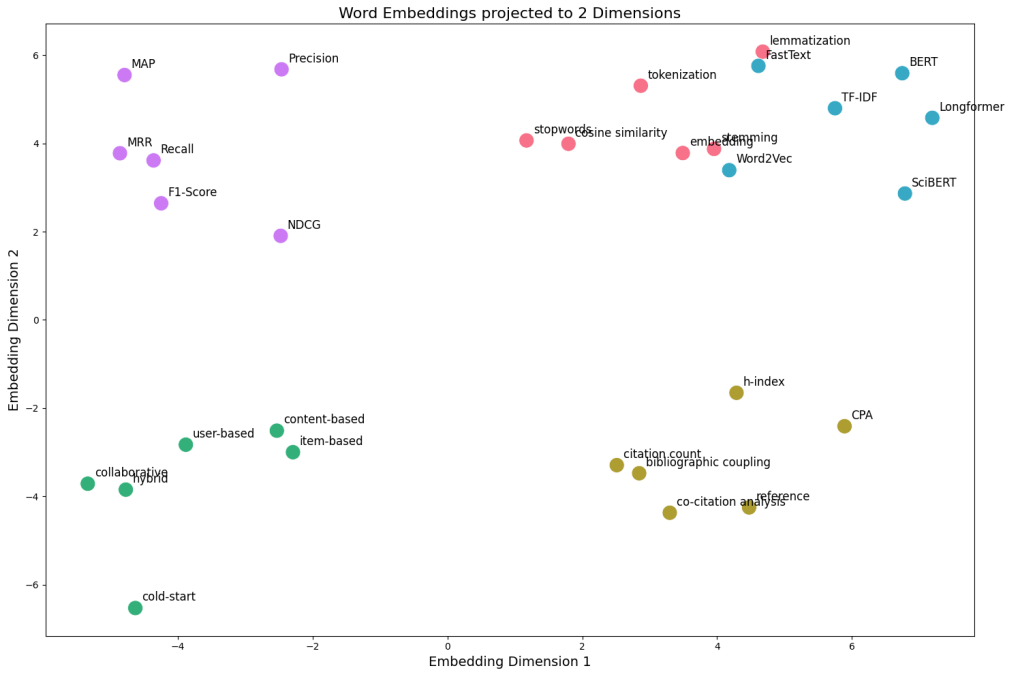


Figure 2.4: Conceptual visualization of the Embedding Space projected onto two dimensions. Words with similar meanings are placed closer together in the embedding space. The colors indicating word categories are added after the fact for visualization purposes and are not available to the model during training. The data for this figure was generated for illustration purposes and does not represent the embedding space of any specific model.

Word embedding models typically rely on the distributional hypothesis [57] as their theoretical foundation. This hypothesis suggests that words used in similar contexts tend to have similar meanings [55]. The term *context* can be defined in various ways. One popular approach uses a sliding window to define the context. The words that surround the target word within this window are considered its context. The window size is a hyperparameter of the model that can be tuned to achieve optimal results.

Learning word embeddings is an unsupervised task, meaning the model learns from raw text without needing explicit semantic or syntactic annotations. This makes them highly scalable and suitable for handling large bodies of text. On the flip side, this implies that embedding models generally cannot capture metadata not contained in the raw text, like the part-of-speech or grammatical role of a word.

After training, an embedding model can map any word in its vocabulary into the embedding

space. However, words not in its vocabulary, known as Out-of-Vocabulary (OOV) words, can pose a challenge. This scenario regularly occurs when pretrained models are used on a different text corpus during inference. To address this, two common approaches are used. The first is to simply ignore OOV words. In this case the unknown tokens do not impact the document embedding. The second approach is to map all OOV words to a special token, such as `<UNK>`, and train the model to learn a representation for this token in the same way as for any other word [55].

Numerous methods exist for generating numerical embedding vectors from text tokens. These methods can be broadly classified into two categories: *sparse embeddings* and *dense embeddings*.

2.3.2 Sparse Embeddings

Sparse vector embeddings belong to a class of text representation models that employ a count-based vector space model. A key advantage of sparse embeddings lies in their interpretability: each dimension corresponds to a unique word from the vocabulary, and the value along that dimension indicates the significance of that word within the document: The higher the frequency of the word, the higher the relevance of that word to the document. This makes sparse vector embeddings a popular choice for tasks like document classification and information retrieval [55, 58].

The Bag-of-Words (BoW) model represents the most basic form of sparse vector embeddings. Within the BoW model, each document is depicted as a vector within an N-dimensional space, where N denotes the size of the vocabulary. Each dimension is associated with a unique word from the vocabulary, and the value along that dimension reflects the count of that word within the document. Although this model is simple and efficient, it treats each word independently and fails to capture the order of the words within the document.

TF-IDF

A more advanced form of sparse vector embeddings is the Term Frequency - Inverse Document Frequency (TF-IDF) model [59]. TF-IDF is a statistical measure that assesses the importance of a word to a document within a collection of documents. Unlike the simple count-based measure used in the BoW model, TF-IDF considers not just the frequency of the word within the document (the term frequency), but also how common the word is across all documents in the corpus (the document frequency).

The TF-IDF model offers the benefit of down-weighting tokens common across all documents, such as stop words, while highlighting tokens unique to a specific document. This makes it more effective than the BoW model at capturing the semantics of the document. The underlying assumption is that a term occuring in many documents is a less accurate discriminator than a term occuring in a small subset of documents and thus should be given less weight [18].

The TF-IDF score for a token t within a document d from a corpus D can be calculated using the formula:

$$\text{tf-idf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D) \tag{2.8}$$

where $\text{tf}(t, d)$ is the term frequency of t in d, and $\text{idf}(t, D)$ is the inverse document frequency of t in D. The term frequency $\text{tf}(t, d)$ is typically the raw count of t in d, although other measures such as the relative frequency are also employed. The inverse document frequency $\text{idf}(t, D)$ quantifies how common or rare t is across all documents in D.

The precise formulas for the term frequency and inverse document frequency can vary. For example, Jurafsky and Martin [55] compute the term frequency using a logarithm with base 10:

$$\text{tf}(t, d) = \log_{10}(1 + \text{count}(t, d)) \tag{2.9}$$

where $\text{count}(t, d)$ is the number of occurrences of t in d.

The rationale for incorporating the logarithm is that "a word appearing 100 times in a document doesn't make that word 100 times more significant to the meaning of the document" [55]. A constant 1 is added to compute the logarithm for words that do not appear in the document (i.e., with a count of 0).

The inverse document frequency is computed by Jurafsky and Martin as

$$\text{idf}(t, D) = \log_{10}\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right) \tag{2.10}$$

where $|D|$ is the total number of documents in the corpus, and $|\{d \in D : t \in d\}|$ is the number of documents in which t appears.

While the impact of slight variations in Equation (2.9) and Equation (2.10) for term frequency and inverse document frequency is often negligible, it is worth noting that the widely used *scikit-learn* [60] implementation of the TF-IDF model defines the term frequency as

$$\text{tf}(t, d) = \text{count}(t, d) \tag{2.11}$$

and the inverse document frequency as

$$\text{idf}(t, D) = \log\left(\frac{1 + |D|}{1 + |\{d \in D : t \in d\}|}\right) + 1 \tag{2.12}$$

where $\log$ is the natural logarithm with base e. A constant 1 is added to the denominator of the inverse document frequency to prevent division by zero.

BM25 and BM25+

BM25 is a ranking function employed in information retrieval systems that builds upon the TF-IDF model with several refinements [61]. The primary improvements include the addition of a document length normalization factor and a term frequency saturation factor. Both are designed to improve the robustness of the scoring mechanism, preventing overly frequent words from dominating the score and accounting for variations in document length.

The BM25 score for a token t within a document d from a corpus D is computed using the formula:

$$\text{BM25}(t, d, D) = \text{BM25}_{tf}(t, d) \cdot \text{BM25}_{idf}(t, D) \tag{2.13}$$

where $\text{BM25}_{tf}(t, d)$ denotes the term frequency component of the BM25 score, and $\text{BM25}_{idf}(t, D)$ represents the inverse document frequency component of the BM25 score.

In the BM25 model, Robertson et al. [61] define the term frequency component as

$$\text{BM25}_{tf}(t, d) = \frac{(k_1 + 1) \cdot \text{tf}(t, d)}{k_1 \cdot ((1 - b) + b \cdot \frac{|d|}{avgdl}) + \text{tf}(t, d)} \tag{2.14}$$

and the inverse document frequency component as

$$\text{BM25}_{idf}(t, D) = \log\left(\frac{|D| - |\{d \in D : t \in d\}| + 0.5}{|\{d \in D : t \in d\}| + 0.5}\right) \tag{2.15}$$

where $k_1 > 0$ and $b \in [0, 1]$ are free parameters, $|d|$ indicates the length of the document d, and $avgdl$ is the average document length in the corpus [61].

The b parameter manages the length normalization. If $b = 0$,

$$(1-b)+b\cdot\frac{|d|}{avgdl}=1 \tag{2.16}$$

resulting in no length normalization. If $b=1$,

$$(1-b)+b\cdot\frac{|d|}{avgdl}=\frac{|d|}{avgdl} \tag{2.17}$$

decreasing BM25$_{tf}$ for long document with $|d|>avgdl$ in Equation (2.14). The length normalization ensures that long documents do not disproportionately benefit due to their size.

The k_1 parameter regulates the saturation of BM25$_{tf}(t,d)$. When k_1 approaches zero, BM25$_{tf}(t,d)$ gravitates towards one, which means that both BM25$_{tf}(t,d)$ and $tf(t,d)$ have minimal impact on the BM25 score as seen in Equation (2.13). Conversely, a large value of k_1 causes BM25$_{tf}(t,d)$ to be acutely sensitive to fluctuations in $tf(t,d)$ when $tf(t,d)$ is low. This is because the contribution of $tf(t,d)$ to the denominator in Equation (2.14) is relatively minor in this range. As $tf(t,d)$ increases, the influence of any additional rise in term frequency begins to diminish, leading to a saturation effect for BM25$_{tf}(t,d)$.

BM25+ [62], an extension of BM25, sets a lower bound for the contribution of infrequent terms to the BM25 score by adding a constant $\delta>0$:

$$\text{BM25+}_{tf}(t,d)=\text{BM25}_{tf}(t,d)+\delta \tag{2.18}$$

This adjustment aims to amplify the influence of rare terms on the BM25 score, ensuring that "even a single occurrence of a search term contributes at least a constant amount" to this score [63]. A frequently used value for δ is 1.0 [62].

2.3.3 Dense Embeddings

While sparse word embeddings are effective, they are inherently high-dimensional. Their dimensionality is equivalent to the size of the vocabulary, which results in a large number of dimensions, most of which have a value of zero. This makes the representation rather inefficient, especially for extensive vocabularies [55, 58].

In contrast, dense word embeddings provide a more condensed representation. They map words into a lower-dimensional space, often in the range of hundreds or thousands of dimensions, where all dimensions carry information and have non-zero values [56, 64, 65]. This compactness not only

enhances computational efficiency but also improves the model's capacity to capture semantic and syntactic relationships between words [56, 64]. In contrast to keyword-based sparse word embeddings, dense word embeddings are able to detect semantic similarity even in the absence of lexical overlap.

A key distinction between dense and sparse word embeddings lies in their learning mechanism. Sparse word embeddings are typically derived from explicit counts or statistics of the text data (e.g., term frequency in TF-IDF). In contrast, dense word embeddings are learned in an unsupervised fashion from large-scale text data [55, 58]. The unsupervised nature of this learning process enables dense word embeddings to leverage large quantities of unlabeled data such as web pages or book corpora.

The subsequent sections provide an overview of two categories of dense embeddings: static and contextual embeddings.

2.3.4 Static Embeddings

Static embeddings provide a word representation technique where each word is depicted by a single, constant vector, independent of its usage context. Despite this approach's limitation in capturing context-dependent semantic nuances, static word embeddings have proven to be highly effective in numerous NLP tasks. Three commonly used models that generate static vector embeddings include Word2Vec, FastText, and GloVe.

Word2Vec

Mikolov et al. [56] introduced the Word2Vec model, which brought a transformative change to the NLP community through its innovative approach to learning word embeddings. Word2Vec includes two related predictive models: Continuous Bag-of-Words (CBOW) and Skip-Gram [66, 67]. Both models derive dense vector representations of words from raw textual data.

In the CBOW model, the goal is to predict a target word from its local context words within a fixed window size. Given a sequence of words $w_1, w_2, \ldots, w_T$ in the corpus, the training objective is to maximize the likelihood of a target word given the context words:

$$L_{\text{CBOW}} = \frac{1}{T} \sum_{t=1}^{T} \log p(w_t | w_{t-\text{window}:t+\text{window}}) \tag{2.19}$$

where $w_{t-\text{window}:t+\text{window}}$ are the context words for the target word w_t.

In contrast, the Skip-Gram model operates in the opposite direction - it aims to predict the context words from a target word. For each target word, the model maximizes the likelihood of the context words given the target word:

$$L_{\text{Skip-Gram}} = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} \log p(w_{t+j}|w_t) \tag{2.20}$$

where w_{t+j} are the context words for the target word w_t, and c is the size of the context window.

Both CBOW and Skip-Gram use a sliding window approach, moving over the entire corpus one word at a time. This results in overlapping context windows, ensuring comprehensive training examples for each word in relation to its local context. The CBOW and Skip-Gram models are compared in Figure 2.5 from Mikolov et al. [56].

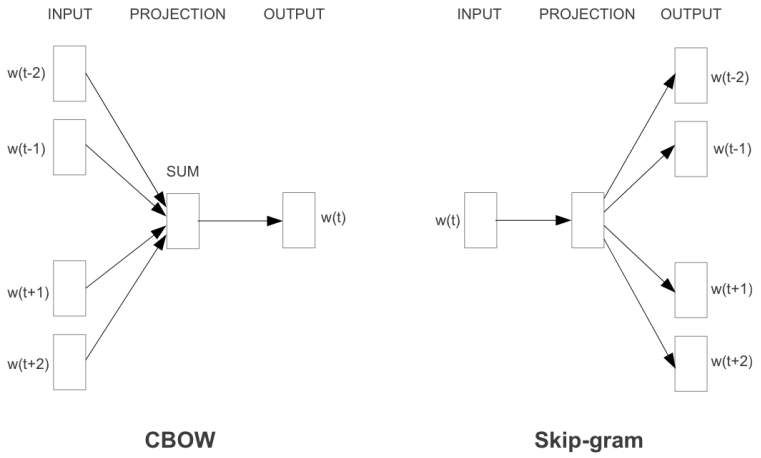


Figure 2.5: Learning Word Embeddings with CBOW and Skip-Gram, illustration from [56]. Both models are predictive models: Whereas CBOW predicts the target word w_t from its context words $w_{t-2}, \ldots, w_{t+2}$, the Skip-Gram model predicts the context words from the target word.

FastText

FastText, an extension of Word2Vec introduced by Bojanowski et al. [65], takes a more granular approach to word representation by treating each word as a bag of character n-grams. This allows FastText to capture the internal structure of words and the relationships between words sharing similar subword structures.

Although FastText employs the same Skip-Gram training objective as Word2Vec, it differs in how it computes the vectors for context or target words. Instead of using a single vector to represent a word, FastText uses the sum of the vectors of the character n-grams that compose the word. This can be mathematically represented as:

$$v_w = \sum_{g \in G_w} z_g \tag{2.21}$$

where v_w is the vector representation of word w, G_w is the set of n-grams for word w, and z_g is the vector representation of n-gram g.

As an example of word decomposition into character n-grams, the word *where* is decomposed into the following character 3-grams: `<wh`, `whe`, `her`, `ere`, `re>`. All n-grams for $3 \leq n \leq 6$ are considered by the default FastText model [65]. The decomposition of words into character n-grams allows FastText to handle rare and OOV words more effectively than Word2Vec. Embeddings of unknown words are generated by summing the embeddings of their constituent subwords.

GloVe

GloVe (Global Vectors for Word Representation), developed by Pennington et al. [64], offers an alternative approach to word embedding distinct from methods like Word2Vec and FastText. While the latter models use predictive strategies that focus on individual local context windows during each training iteration, GloVe takes a count-based approach. It constructs a global word-word co-occurrence matrix by accumulating local context information across the entire corpus.

The unique strength of GloVe is its ability to combine both global corpus statistics and local semantic relationships into one unified model. To achieve this, it uses a weighted least squares regression objective function that operates directly on the co-occurrence counts matrix:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2 \tag{2.22}$$

In Equation (2.22), V denotes the vocabulary size, w_i and $\tilde{w}_j$ represent the word vectors of word i and context word j respectively. b_i and $\tilde{b}_j$ denote their respective biases. X_{ij} quantifies the number of times word j occurs in the context of word i within a defined window size. Lastly, f is a weighting function that diminishes the impact of both very rare and overly frequent co-occurrences.

Hence, the goal is to align the dot product of the word vectors ($w_i^T \tilde{w}_j$) and their biases ($b_i + \tilde{b}_j$) with the logarithm of the co-occurrence count ($\log X_{ij}$). The word embeddings are learned by minimizing the sum of squared differences between these values, weighted by the factor $f(X_{ij})$.

2.3.5 Contextual Embeddings

Unlike traditional static embeddings, which generate a single, fixed representation for each word regardless of its context, contextual embeddings produce a unique representation for each word based on its specific context within a sentence or a document.

As pointed out by Ruas et al. [68], this advantage over static embeddings is particularly pronounced for polysemous words - words with multiple meanings. The author's methodology is rooted in integrating Word Sense Disambiguation (WSD) into the word embedding process. WSD is a task in NLP that identifies which sense of a word is used in a particular context. By incorporating WSD into the word embedding generation process, the authors ensure that different senses of a word are accurately captured and represented.

Transformer-based models [69] such as BERT [70], SciBERT [71], and Longformer [72] generate rich contextually-aware word embeddings by leveraging the Transformer architecture's self-attention mechanism.

Attention in Language Models

The *attention mechanism*, first introduced by Bahdanau et al. [73], enables models to focus specifically on certain parts of an input sequence while generating each element of the output. This selective focus enables the model to detect relationships over long distances, irrespective of the actual distances between words. This represents a notable improvement over recurrent neural network (RNN) architectures, which process input sequences sequentially and struggle with long sequences due to the vanishing gradient problem. Further, in contrast to recurrent neural networks, the attention mechanism allows the model to process the input sequence in parallel, leading to significant performance improvements [69, 74].

In achieving this, the attention mechanism uses a process known as soft alignment between the input and output sequences. This process is particularly crucial for machine translation. In this scenario, the model learns to align the source and target sequences, focusing on pertinent parts of the source sequence as it generates each word in the target sequence. The alignment is realized by computing an attention score for each word in the source sequence. This score indicates the degree to which the model concentrates on that word while generating the target sequence.

The *attention weight* a_{ij} for a source word j when generating a target word i is determined as follows [73]:

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \tag{2.23}$$

In Equation (2.23), e_{ij} stands for the *attention score* between target word i and source word j, and

T_x represents the length of the source sequence. The attention score is typically computed as the dot product of the embeddings of the target and source words, although other scoring methods can also be applied. Once the attention scores are computed, they are passed to a softmax operation to ensure their sum equals one across the source sequence. The resulting attention weights a_{ij} indicate the importance of the source word j when generating the target word i. The attention weights are then used to compute a weighted sum of the source word embeddings, resulting in the final output of the attention mechanism.

The attention mechanism is illustrated in Figure 2.6 by Vig [75] on the example of BERT [70].

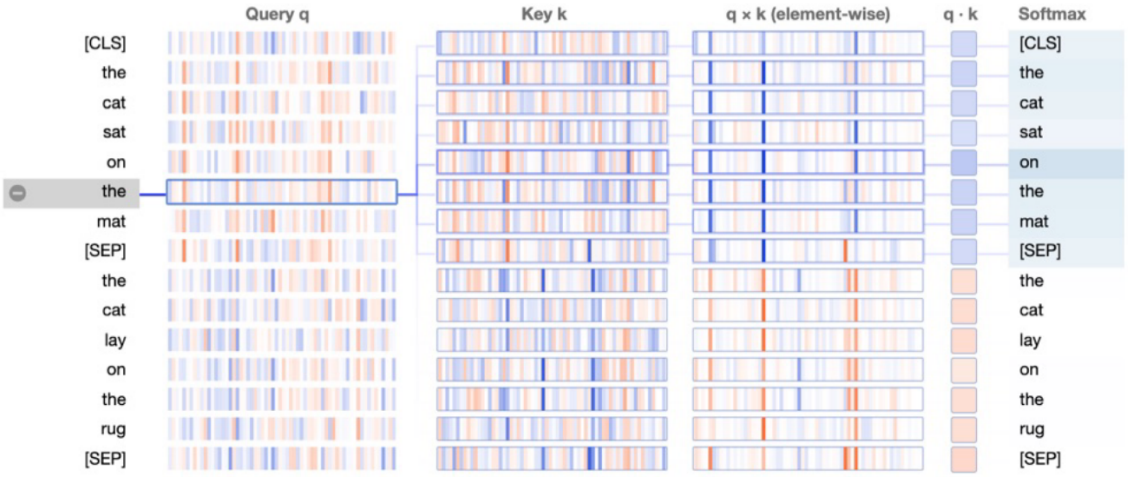


Figure 2.6: Visualization of the (self-) attention mechanism using BertViz [75]. The text tokens on the left represent the input sequence. Values of the query and key vectors are depicted as vertical bands of color. Blue represents positive values, while orange indicates negative ones. The color saturation is based on the value's magnitude. Lines connecting the query and key vectors represent attention weights. The line width corresponds to the magnitude of the attention weight.

The Transformer

The Transformer model, introduced by Vaswani et al. [69], is the foundation of many of the recent advancements in NLP. It was designed to handle sequence data efficiently and is thus well-suited for NLP tasks. The cornerstone of the Transformer model is the self-attention mechanism. This is a particular implementation of the attention mechanism that is extended and modified to be applied not only between the source and target sequences but also within a single sequence. This allows the model to assign different weights to the words in a sentence when generating the embedding for a particular word, much like the attention mechanism described in the preceding section.

The self-attention mechanism associates each word in a sentence with a query, key, and value. These are derived from the input vector of the word and are instrumental in computing the

attention score. The attention score between a query Q_i and a key K_j is calculated as the dot product of Q_i and K_j, scaled by the inverse square root of the dimension of the key vectors to counteract the potential of the dot product growing large for high-dimensional vectors.

Formally, the scaled dot-product attention is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.24}$$

where d_k is the dimensionality of the key vectors, Q is the matrix of query vectors, K is the matrix of key vectors, and V is the matrix of value vectors. The element-wise softmax function ensures the attention scores are normalized to sum to one.

The result of the softmax operation represents the attention weights in matrix form, showing the importance of each word in the context of the others. These attention weights are subsequently used to compute a weighted sum of the value vectors, encapsulating the relevant information from the sequence based on the learned attention scores.

Extending this mechanism, the Transformer model employs multi-head attention, wherein the model has multiple parallel attention layers referred to as *heads*. Each head independently computes a different learned linear transformation of the input. As illustrated in Figure 2.7, the final output of the multi-head attention layer is the concatenation of the outputs of each head, followed by a linear transformation. This approach allows the model to capture various types of information and dependencies from the input sequence [76].

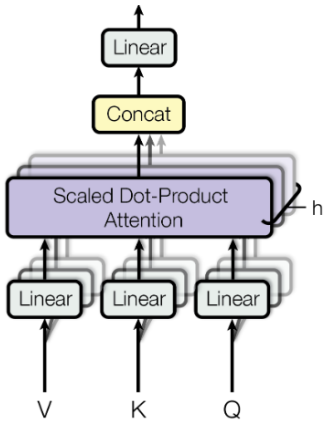


Figure 2.7: Multi-head attention in the Transformer, illustration from [69]. The input is transformed into query, key, and value vectors. Each of the h attention heads transforms the input independently. The outputs of the heads are concatenated and linearly transformed to produce the final output.

BERT

BERT (Bidirectional Encoder Representations from Transformers) is a Transformer-based model proposed by Devlin et al. [70]. It builds upon the self-attention mechanism of the Transformer model to pre-train deep bidirectional representations from unlabeled text, conditioning on both left and right context in all layers. This bidirectionality allows BERT to learn context-rich word representations, a significant advancement over models which only consider unidirectional context [77,78].

The pretraining process of BERT leverages two unsupervised tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). In the MLM task, about 15% of the input tokens are randomly masked and the model is trained to predict these masked tokens based on the context provided by the non-masked tokens [70]. This task encourages BERT to understand the context and relationships between words.

The NSP task involves predicting whether a given sentence is the subsequent sentence of another sentence in the original text. During training, half of the inputs are pairs of sentences in which the second sentence is the subsequent sentence in the original document, while for the other half, a random sentence from the corpus is chosen as the second sentence. This task helps BERT understand relationships between sentences, a useful skill for many downstream NLP tasks.

Once BERT is pretrained using a large corpus of text, it can be fine-tuned on downstream tasks using a relatively small amount of task-specific training data. The fine-tuning process involves minor adjustments to the pretrained parameters, allowing BERT to apply the general language understanding acquired during pretraining to the downstream task. This process is more efficient and requires less data than training a model from scratch, making BERT a highly effective model for a wide range of NLP tasks.

SciBERT

SciBERT, developed by Beltagy et al. [71], is a specialized variant of BERT that is designed to handle the unique language patterns and technical terminology found in scientific literature. This variant is especially effective for tasks related to scientific document classification, information extraction from scientific papers, and other tasks requiring comprehension of scientific text. Like BERT, SciBERT is pretrained using the MLM and NSP tasks. However, the training corpus is constructed from scientific papers, providing SciBERT with a stronger understanding of the language used in scientific literature.

The pretraining of SciBERT is drawn from the Semantic Scholar corpus, which includes approximately 1.14 million scientific papers spanning a wide range of disciplines. WordPiece tokenization [79] that breaks down words into subwords is used to handle OOV words, which are quite common in the scientific domain due to the prevalence of unique terminologies, abbreviations, and symbols. This corpus and vocabulary construction strategy allows SciBERT to effectively handle

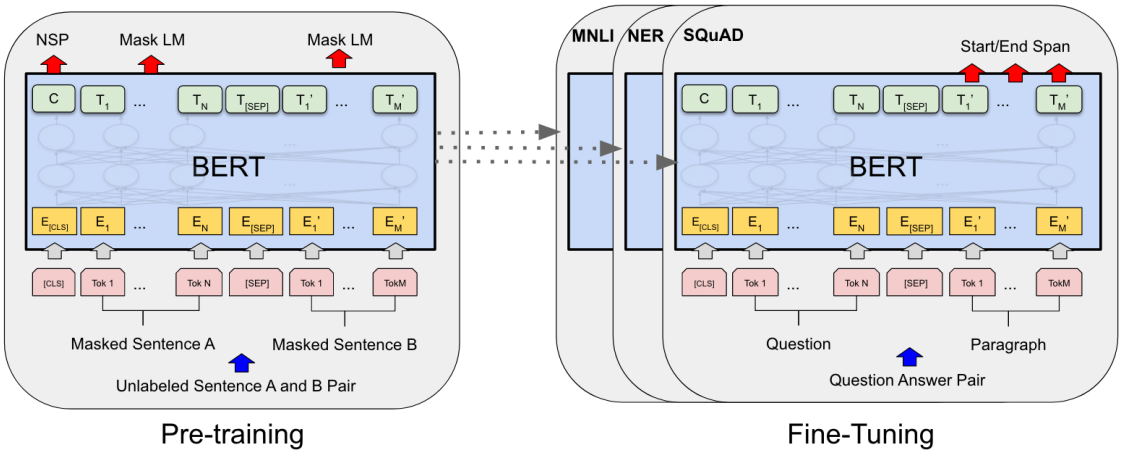


Figure 2.8: Pretraining and fine-tuning of BERT, illustration from [70]. Left: BERT is pretrained on two unsupervised tasks: Masked Language Modeling and Next Sentence Prediction. Right: The pretrained model can be fine-tuned for specific tasks, such as question answering, sentiment analysis, or named entity recognition, enhancing its downstream performance.

the specific language patterns and technical jargon found in scientific literature.

The performance of SciBERT is tested on a range of NLP tasks including sequence tagging (NER), sentence classification (textual entailment, sentiment analysis), and dependency parsing. The authors found that SciBERT outperformed BERT and achieved new state-of-the-art results on several tasks, particularly those involving scientific text [71].

Longformer

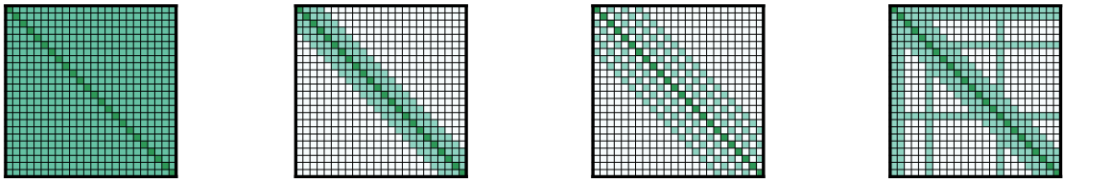
The Longformer model, developed by Beltagy et al. [72], aims to overcome the computational challenges of Transformer-based models like BERT when dealing with long documents. The self-attention mechanism in Transformer models has a computational complexity that grows quadratically with the length of the input sequence, making it inefficient for processing large-scale documents. Longformer addresses this issue by modifying BERT's self-attention mechanism to scale linearly with the input sequence length. By integrating local and global attention mechanisms, Longformer offers an efficient solution for processing long documents, capturing their global context.

This novel self-attention mechanism, termed *sliding window attention*, utilizes a fixed-size window that moves across the input sequence. Each token can only attend to other tokens within this window, effectively limiting the contextual scope of each token. This approach allows the model to manage computational resources efficiently, even when dealing with long sequences.

To enable the model to capture global context, Longformer integrates a limited number of globally

attentive tokens into its architecture. These tokens can attend to the entire sequence, providing a mechanism for the model to gather information across the full length of the input. This feature is important for tasks that require a comprehensive understanding of the entire document, such as document classification and summarization.

The sliding window attention mechanism is contrasted from the traditional attention mechanism in Figure 2.9 from Beltagy et al. [72]. Table 3.1 compares the BERT and Longformer models in more detail including their tokenization processes.



(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

Figure 2.9: Sliding window attention in the Longformer, as presented in [72]. Image a) demonstrates the standard self-attention mechanism where each token can attend to all other tokens. Image b) illustrates the sliding window attention mechanism where each token attends only to tokens within a fixed-size window. Image c) depicts the dilated sliding window attention where the tokens in the window are spaced with increasing gaps, allowing the model to attend to distant tokens without enlarging the window size. Image d) introduces global attention tokens that can interact with all other tokens in the sequence, capturing long-range dependencies.

2.3.6 Document Embeddings

While word embeddings can capture the semantic content of individual words, they are limited in their ability to represent larger textual units such as whole documents. Document embeddings address this limitation by mapping entire documents to a high-dimensional vector space [80].

A straightforward method to derive document embeddings is by aggregating the embeddings of all individual words in the document. Common approaches include averaging the word embeddings or calculating their dimension-wise maximum or sum. However, these techniques assume that all words contribute equally to the document's meaning and they do not consider the order of words. Further, the dimension-wise maximum is sensitive to outliers, whereas the sum is impacted by the document length.

To address these limitations, Kanakia et al. [38] propose a weighted average of word embeddings, computed as a normalized linear combination of the word vectors weighted by their TF-IDF scores [59]. This method considers the significance of each word within the document.

Le and Mikolov [81] extend the Word2Vec model by introducing *Paragraph Vectors*, which are shared across all words in the paragraph. The paragraph vector is trained simultaneously with the word

embeddings to predict a word from its context. After training, the paragraph vector is a semantic representation of the entire paragraph. This model, often referred to as *Doc2Vec*, has been used in diverse applications, including document classification [82] and citation recommendation [15, 83].

Ruas et al. [84] introduce two novel algorithms, which aim to enhance traditional word embeddings by leveraging multi-semantic representation through lexical chains. This method emphasizes the importance of the relationship between words in a sentence, capturing more about the underlying semantic content of a document than its individual words. By integrating semantic relations derived from lexical chains, document embeddings can be constructed from the most important correlated senses of the words in the document.

An alternative approach to document embeddings is proposed by Ostendorff et al. [85]. Instead of representing a document as a single generic embedding, they represent it as multiple specialized embeddings, each capturing a distinct aspect of the document. This is achieved by considering the task, method, and dataset of the respective research papers as their aspects. The specialized embeddings are generated using various models, including FastText [65], SciBERT [71], and SPECTER [13]. Compared to traditional document embeddings, this method offers a more nuanced understanding of document similarity by effectively capturing the distinct aspects that make research papers similar.

2.3.7 Document Similarity

Once document embeddings are generated, it is possible to compute the similarity between documents. There are numerous document similarity measures to choose from [86], and the choice of measure will depend on the specific task and the nature of the documents being compared. Two of the most popular measures are *cosine similarity* and *Jaccard similarity*.

Cosine Similarity

Cosine similarity is calculated as the cosine of the angle θ between two numerical document embedding vectors $\mathbf{a}$ and $\mathbf{b}$. In mathematical terms, this can be written as:

$$\text{cosine similarity}(\mathbf{a}, \mathbf{b}) = \cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|} \tag{2.25}$$

In Equation (2.25), $\mathbf{a} \cdot \mathbf{b}$ is the dot product of $\mathbf{a}$ and $\mathbf{b}$, $\|\mathbf{a}\|$ and $\|\mathbf{b}\|$ are the magnitudes (Euclidean norms) of $\mathbf{a}$ and $\mathbf{b}$ respectively and θ is the angle between $\mathbf{a}$ and $\mathbf{b}$ [87].

Cosine similarity values range from -1 to 1. As the cosine similarity between two vectors is closely linked to the correlation coefficient, the interpretation of these two measures is similar. A value of 1 indicates perfect similarity, 0 means no similarity, and -1 represents perfect dissimilarity. Cosine

similarity is computationally efficient and is not influenced by document length. However, it might not be the best fit when the distribution of words is important, as it does not take into account the frequency or order of words.

Jaccard Similarity

The Jaccard similarity coefficient [88] is another document similarity measure that works directly on the sets of words (tokens) in documents, not their document embedding vectors. It is the size of the intersection divided by the size of the union of the two word sets. This can be mathematically expressed as:

$$\text{Jaccard similarity}(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{2.26}$$

where $|A|$ and $|B|$ are the cardinalities of the sets A and B, respectively.

In Equation (2.26), A and B represent the sets of words in the two documents being compared. In contrast to cosine similarity, the Jaccard similarity coefficient remains unaffected by word repetitions due to its use of sets containing unique elements. This makes it a good choice in cases where the presence or absence of words is more critical than their frequency or order, e.g. for short documents like titles and abstracts [18].

Advanced Similarity Measures

Traditional document similarity measures such as cosine and Jaccard similarity may fall short in capturing the heterogeneous semantic relations between documents, as they do not consider the context in which similarity is evaluated. To address this, Ostendorff et al. [82] propose a multi-class classification model for document pairs to determine semantic relations between documents. Their results reveal that even complex relation classes yield promising performance, which is crucial for recommender systems and information retrieval applications. They find that models that attend to both documents simultaneously, such as Siamese BERT [70, 89] and Siamese XLNet [90], generally outperform traditional methods like AvgGloVe [64] and Doc2vec [81], which classify concatenated document vectors. However, these traditional methods still show reasonable performance, especially in scenarios where computational resources are limited.

2.3.8 Applications in Paper Recommendation

In academic paper recommendation, language models are often utilized to generate document embeddings for a given query paper and a set of potential candidate papers. Based on these embeddings, the similarity between the query paper and each candidate paper is computed with a similarity measure such as cosine similarity. The candidate papers are then ranked based on their

similarity score to the query paper with the highest-ranked papers being recommended to the user. The details of this process including the choice of language model and the document sections used for generating embeddings can vary significantly among different approaches.

Bhagavatula et al. [15] introduce a content-based strategy for citation recommendation. Existing citation recommendation systems often rely on metadata from the query documents, such as author names and publication venues. The authors note that during instances like the peer review process, this information might be missing. To address this issue, the authors propose a content-based method for citation recommendation that does not rely on metadata but instead uses the full text of the query document. First, they embed the given query document into a vector space using Doc2Vec [81]. Then, they identify papers with the highest cosine similarity as citation candidates. Finally, the authors train a discriminative model to predict the relevance of each candidate citation, which is then used to rank the candidates by predicted relevance.

Remaining in the context of paper citations, Cohan et al. [13] introduce *SPECTER*, a technique tailored for generating document embeddings within the scientific literature domain. SPECTER uses a transformer-based architecture that is pretrained with a citation prediction objective: given a citing paper and a list of candidate cited papers, it must identify the true cited paper. This objective allows the model to learn contextualized document representations from the citation graph. SPECTER uses titles and abstracts, concatenated and preprocessed with SciBERT's [71] vocabulary, as its input.

Nascimento et al. [17] propose a content-based approach to generate recommendations using only a single research paper as input. The authors propose a sequential strategy: First, a set of candidate papers is generated with potential queries using terms from the input paper. These queries are then submitted to existing web information sources that hold research papers. Once a set of candidate papers for recommendation is retrieved, the framework applies content-based recommender algorithms to rank these candidates by using the paper title and abstract.

Mass and Roitman [12] enhance ad-hoc document retrieval by leveraging the capabilities of pretrained models like BERT [70] and GPT2 [91]. Their method uses weak supervision, leveraging unsupervised rankings from traditional retrieval methods and collecting pseudo-relevance feedback to curate training data. This data then serves to fine-tune the aforementioned pretrained models, with the objective of achieving enhanced retrieval performance. The models leverage the titles and abstracts of the papers to generate the embeddings.

Finally, Mohamed et al. [14] conduct a comparative evaluation of five widely used pretrained sentence encoders: USE [92], BERT [70], InferSent [93], ELMo [94], and SciBERT [71], in the context of research paper recommendation using only the paper titles. For each of the five models, the sentence embeddings of the input paper title and the candidate paper titles are calculated and their cosine similarity is used as a measure of semantic similarity. The authors observe that the efficacy of all models in recommending research papers, when considering only titles, is not on par with

traditional techniques like BM25 [61]. However, when combined with BM25 in a hybrid approach, the performance improves. In the hybrid setup, a linear combination of BM25 scores and semantic similarity scores is used to generate the final ranked recommendations.

Despite the widespread adoption of Transformer models, Beel et al. [23] argue that models emphasizing lexical attributes over semantic ones remain the preferred choice for research paper recommendations. Their analysis discerned that 83% of all content-based methods from 1998 to 2013 and 67% of methods from 2014 to 2021 adopt TF-IDF as their principal weighting mechanism. Nonetheless, the rise of LLMs has sparked the interest in employing Transformer models for recommendation tasks. A discussion on the potential of LLMs in recommender systems can be found in the Appendix.

2.4 Hybrid Paper Recommenders

Section 2.2 and Section 2.3 discussed various existing citation-based and content-based approaches to recommender systems. While literature on citation analysis has grown steadily over the past few decades, the recent surge in LLMs has spurred an explosion of research on content-based methods. However, in spite of the comprehensive literature in both of these areas, there is limited research on *hybrid* recommender systems that integrate both language models and citation-based approaches for recommending academic papers. This combination, though promising for more personalized and comprehensive recommendations, remains largely unexplored.

This section provides an overview of previous work on hybrid paper recommendation, identifies gaps in the existing literature, and emphasizes the contributions of this thesis to the field.

2.4.1 Literature Overview

Kanakia et al. [38] developed a hybrid recommender system that effectively integrates citation-based and content-based recommendation methods. The authors contend that while pure citation-based recommendations deliver high quality, they face challenges with coverage. Conversely, while content-based recommendations offer high coverage, they often lag in quality and demand significant computational resources. To address these challenges, they introduced a weighted mixed hybridization approach combining the two methods. The citation-based recommender employs co-citation analysis using citation data from the Microsoft Academic Graph[1], while the content-based recommender utilizes a Word2Vec model [56] to produce document embeddings from paper titles and abstracts. To bypass the quadratic runtime complexity of pairwise similarity computations, they adopt spherical K-Means clustering to minimize the number of pairwise comparisons.

Färber and Sampath [83] introduce a hybrid model for citation recommendation that incorpo-

[1] https://www.microsoft.com/en-us/research/project/microsoft-academic-graph/

rates multiple content-based approaches, encompassing embedding-based, topic modeling, and information retrieval techniques. The authors focus on local citation recommendation, wherein the objective is to suggest citations for specific sections of text within a document, termed as citation contexts. Their model utilizes embeddings to capture semantic similarity, topic modeling to identify shared themes, and information retrieval to pinpoint similar documents based on term frequency. By combining these methods into a hybrid system, they show that different similarity measures can provide complementary benefits in citation recommendation.

Khadka et al. [2] designed a hybrid recommender system that blends CBF, CF techniques, and citation knowledge. Their research stems from the realization that pure collaborative filtering methods are ineffective for recommending newly published papers, as they suffer from the cold-start problem. As is common in CBF, they initially build user profiles using *user-based* features, such as the user's publication history. Then, citation knowledge is used to extract *item-based* features of research papers, such as both direct and indirect citations, along with the citation context. Recommendations are generated by computing similarity scores between users and papers based on the CF nearest neighbor heuristic.

Finally, Ostendorff et al. [95] developed an aspect-based document similarity measure for academic papers by fusing content-based and citation-based approaches. The authors observe that conventional document similarity measures only provide a binary sense of similarity, without explaining in what aspects two documents are similar. To enhance the granularity of document similarity, they propose a pairwise document classification task using section titles in which citations occur as labels for the document pairs. By applying various Transformer models to this task, they succeed in capturing the specific aspects in which two documents align.

2.4.2 readnext: A Hybrid Recommender System for Computer Science Papers

This thesis introduces the *readnext* framework, a hybrid recommender system for Computer Science papers that incorporates both citation-based and content-based features, as well as global document characteristics.

On the shoulders of giants

Our approach derives inspiration from previous work in the field. In line with the content-based approach of Nascimento et al. [17], *readnext* also uses a single query paper to generate a list of recommended papers. Similar to the approach of Khadka et al. [2], we use co-citation analysis and bibliographic coupling to incorporate citation information into the recommendation process. We agree with the authors that many citation-based methods are biased towards established papers. Therefore, we include the publication date of papers as a feature to prioritize recently published papers in the recommendation process.

As noted by Beel et al. [23], TF-IDF remains a strong baseline for content-based recommendation,

competitive with modern Transformer models. Therefore, we use TF-IDF as a baseline for our content-based recommender. Due to the strong performance of BM25 for paper recommendation highlighted by Mohamed et al. [14], we include BM25 as an alternative. Moreover, BERT has repeatedly proven its capability within document retrieval tasks [12, 96], while SciBERT has been shown to be particularly effective for scientific papers [95]. Thus, we include these two language models as options for our framework. Finally, the work of Kanakia et al. [38] has influenced performance optimization strategies within *readnext* to address the quadratic runtime complexity of pairwise similarity computations.

Contributions of this thesis

However, the *readnext* recommender system sets itself apart from existing work in several ways.

First, it provides unique customizability, allowing users to select their preferred language model for the content-based recommender and define the weighting scheme for combining global document features and citation features for the citation-based recommender. Recommendations can be retrieved from each of the two individual recommenders as well as from the hybrid recommender for both hybrid orderings.

Second, this thesis presents a systematic and data-driven evaluation of recommendations generated by *readnext* with different input parameters and recommender orderings. The results allow for robust statements about the benefits of incorporating a content-based modeling step into traditional citation-based literature recommenders, offering insights into the interaction of these methods and suggesting directions for future research on hybrid recommender systems for academic papers.

Finally, *readnext* is open-source and freely accessible on GitHub[2]. The project not only provides the source code to replicate the results of this thesis, but also includes comprehensive documentation[3] on how researchers can install, extend and further customize the recommender system to their own needs. This transparency allows practitioners to use the *readnext* framework as a foundation for their future research, advancing the field of hybrid recommender systems for academic papers.

[2] `https://github.com/joel-beck/readnext`
[3] `https://joel-beck.github.io/readnext/`

Chapter 3

Methodology

The *readnext* framework provides a hybrid recommender system, utilizing both citation-based and content-based features, to recommend scientific papers within the Computer Science domain. This chapter addresses **RT 2** as defined in Section 1.2 by delving into the methodology behind the creation of this recommender system. This encompasses the dataset construction, the training and the inference process.

Section 3.1 begins with a general overview of the Hybrid Recommender architecture. It introduces both the Citation Recommender and the Language Recommender, and explains how they are combined to form the Hybrid Recommender. Additionally, it summarizes the structure of the evaluation and explains the choice of the evaluation metric.

Section 3.2 explains the process of dataset construction. It includes information about the different data sources and the rationale behind the dataset's design.

Section 3.3 discusses the training or precomputation stage. The section outlines the preprocessing and computation steps for both the Citation Recommender and the Language Recommender. Moreover, it describes how this information is stored for subsequent use during the inference stage.

Lastly, Section 3.4 elaborates on how users can harness the framework to generate recommendations based on a specific query paper. The section differentiates between query papers included in the training dataset and those that are not. It also emphasizes how the training stage accelerates the inference process.

3.1 Overview

3.1.1 Hybrid Recommender Architecture

From a high-level perspective, the *readnext* framework comprises three main components: the Citation Recommender, the Language Recommender, and the Hybrid Recommender. For consistent notation throughout this thesis, the term *Hybrid Recommender* is capitalized when referring to the specific hybrid system developed in this thesis. Conversely, the lowercase version is used when referring to the general concept of hybrid recommender systems.

Citation Recommender

The Citation Recommender generates recommendations based on five features of two different categories: three global document characteristics and two citation-based features.

Global Document Characteristics

The global document characteristics are derived from a paper's metadata. The Citation Recommender uses the following three features:

- The *publication date* of the paper serves as an indicator of *novelty*. More recent publications are given a higher score because they build on previous papers, compare and contrast their findings with existing results, and are thus perceived as more valuable. The publication date feature compensates for the lower citation count newer papers might encounter.
- The *paper citation count* acts as a measure of *document popularity*. Papers with higher citation counts are, on average and without any prior knowledge, considered more valuable and relevant.
- The *author citation count* across all publications functions as a measure of *author popularity*. Authors with a higher citation count are regarded as more influential in the research community. For papers with multiple authors, the citation count of the most-cited author is selected. This method is preferred over using the average citation count of all co-authors based on the assumption that the co-authorship of a single, well-established researcher has a more significant impact on the paper's reputation than the co-authorship of several less established researchers. Additionally, the value should be independent of the number of authors, which means aggregation measures like the sum of all co-author citation counts are not suitable.

These features are independent of the query paper for which recommendations are generated. For instance, the candidate paper with the highest citation count will always rank first among all candidates for that feature, irrespective of the specific query paper being considered.

Citation-based Features

In addition to global document features, the Citation Recommender uses *co-citation analysis* [47, 49] and *bibliographic coupling* [48] scores between the query paper and each candidate paper. These are static citation analysis measures that neither consider the citation location nor the context. While location-aware methods, such as CPA [10] and section-based bibliographic coupling [9], often yield superior results, they necessitate full-text access to discern the location of citations. As the full document texts are not available for this thesis, we stick to static citation analysis measures.

Unlike the global document characteristics, the citation-based features strongly depend on the query paper as they generate pairwise scores between the query paper and each candidate paper. As such, a particular candidate paper might have a high co-citation analysis score with one query paper but a low score with another.

Language Recommender

The Language Recommender employs the text content of the paper to generate recommendations. Due to the lack of full text access, the recommender utilizes paper abstracts, which are first tokenized and then embedded into a vector space using one of eight language models. Finally, the content similarity between query and candidate papers is calculated using the cosine similarity between their respective vector embeddings.

The eight language models can be categorized as follows: Keyword-based language models generating sparse vector embeddings and language models generating dense vector embeddings. Dense vector embedding models are further divided into static and contextual embedding models. Within each category, a popular foundational model (TF-IDF [59], Word2Vec [56], and BERT [70]) is employed, as well as one or two variations developed to improve upon this base model (BM25 [61], FastText [65] & GloVe [64], and SciBERT [71] & Longformer [72]).

Hybrid Recommender

The Hybrid Recommender merges the recommendations of the Citation Recommender and the Language Recommender using the sequential cascade strategy (Section 2.1.3). Each recommender produces a preliminary list of 20 recommendation candidates, which the other recommender then re-ranks to generate the final ranking. The value $n = 20$ for the size of the candidate list is configurable by the user, but indicated a good trade-off between performance and computational complexity in our experiments.

If the Citation Recommender operates first, it ranks all candidate papers based on a weighted score from global document characteristics and citation-based features. The top 20 papers with the best weighted score are subsequently passed to the Language Recommender, which then re-ranks the candidate list based on content similarity of the paper abstracts.

Figure 3.1 illustrates the interaction between the Citation Recommender and the Language Recommender within the hybrid system.

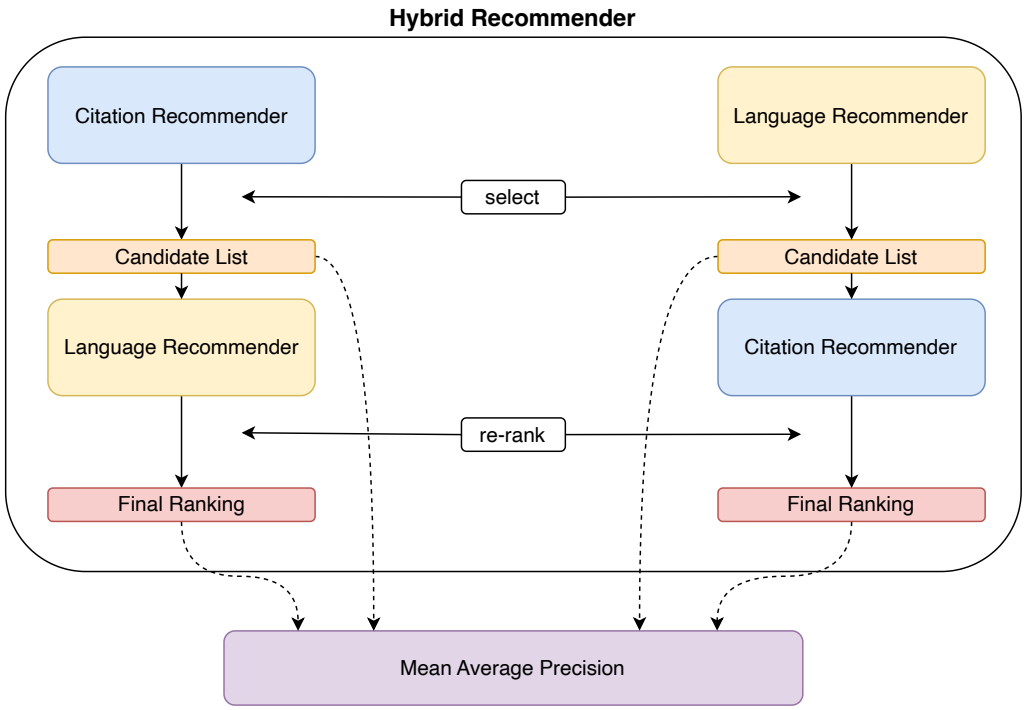


Figure 3.1: Hybrid Recommender Architecture. The Hybrid Recommender combines the recommendations of the Citation Recommender and the Language Recommender using a cascade strategy: Each recommender selects a candidate list of recommendations which is then re-ranked by the other recommender. The MAP is calculated and compared for both candidate and both final rankings.

Evaluation

Among the various metrics available for recommender systems (Section 2.1.5), the MAP is selected as the primary evaluation metric for this thesis for the following reasons:

1. MAP considers the order of recommendations, emphasizing not just the relevance of the items but also their positioning early in the list [41, 42]. This is not the case for the Precision, Recall, and F1-Score metrics [21].

2. All items on the recommendation list are considered, i.e. it's not only important to recommend relevant items but also to avoid irrelevant items. This is not the case for the MRR metric, which only considers the first relevant item on the recommendation list [40].

3. MAP works with binary 0/1 encoded labels, which in this thesis indicate irrelevant/relevant recommendations. This is not the case for the NDCG metric, which works better with graded

relevance labels like irrelevant, somewhat relevant, relevant, and highly relevant [40].

During the evaluation process, the MAP is calculated not only for the final rankings of the Hybrid Recommender for both the C $\rightarrow$ L and L $\rightarrow$ C orderings, but also for both intermediate candidate lists. Employing the same metric for both candidate and final rankings facilitates direct performance comparison. This includes: (i) contrasting the candidate lists of the Citation Recommender and the Language Recommender to identify the superior candidate selector, (ii) evaluating final recommendation rankings to pinpoint the optimal hybrid ordering, and (iii) comparing candidate and final rankings to assess if the re-ranking step enhances recommendation quality.

3.2 Dataset Construction

Recommender systems recommend items to users that are retrieved from some form of data storage. In the case of *readnext*, this data storage takes the form of a dataset assembled from multiple sources. This dataset will be referenced as *base* or *readnext* dataset throughout this thesis to distinguish it from the *training*, *validation*, and *test* splits used for evaluation Chapter 4.

Several factors must be considered during the construction of the *readnext* dataset:

1. The dataset must not be too *small* as this would limit the number of potential recommendations, increasing the likelihood that none of the recommendations is relevant or that the user has already read many of the recommended papers. On the other hand, if the dataset is too *large*, it could amplify the computational complexity of the recommender system and extend the waiting time for the user until recommendations are generated.

2. The dataset must not be too *narrow* in its coverage of research areas, as this would limit the number of users who could benefit from the recommender system. Conversely, the dataset must not be too *broad*, as this would increase the likelihood that the recommendations are irrelevant to the user's research interests.

3. The dataset should be reasonably *up-to-date*. As the relevance of papers diminishes over time, the recommendations need to reflect the user's evolving research interests.

4. All features requisite for the Hybrid Recommender must be present for each paper in the dataset. Documents with missing values that cannot be inferred from other observations, such as a missing paper abstract, are excluded from the dataset.

5. The *labels* for the evaluation must satisfy several conditions. First, they must be available for all papers in the dataset since they define which recommendations are considered relevant and which are deemed irrelevant. Second, they must be sufficiently granular to allow for a meaningful evaluation of the recommendations. For example, if the labeling scheme only covers two classes and each class is very broad, many recommended papers might be considered relevant even if they are only loosely related to the query paper. On the other

hand, if the number of classes is too large and each class is highly specific, papers that are of high interest to the user might be considered irrelevant. Having excessively narrow classes can also prevent serendipitous and interdisciplinary recommendations, harming research diversity.

Figure 3.2 illustrates the different data sources for this thesis. The subsequent sections describe each data source in detail.

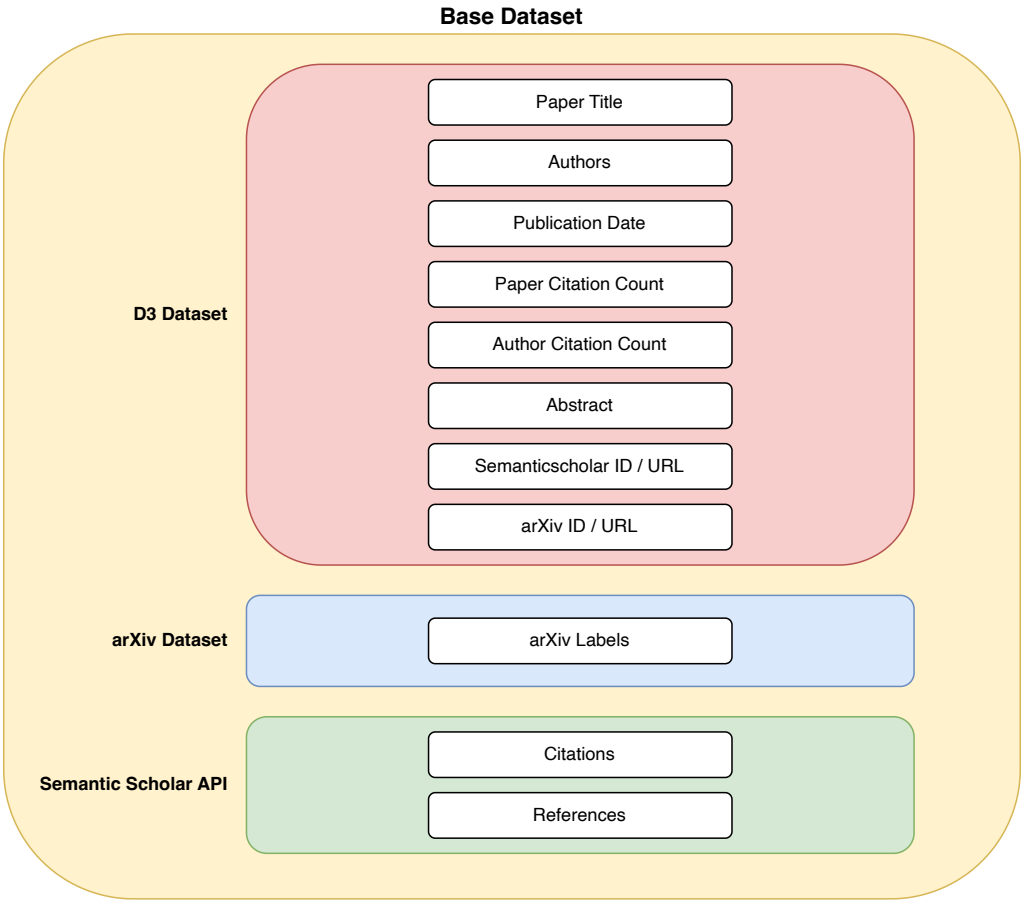


Figure 3.2: Data Sources used in this thesis. The D3 dataset provides paper metadata, including the publication date, citation counts and the paper abstract. ArXiv categories are added as evaluation labels from the arXiv Kaggle dataset. Lastly, the Semantic Scholar Academic Graph API is fetched to add lists of individual citations and references for each paper.

3.2.1 The D3 Dataset

The DBLP Discovery Dataset (D3) [97] is the primary data source for this thesis. Comprising over 6 million Computer Science papers, the D3 dataset is a curated compilation of research metadata

harvested from the DBLP open-access repository[1]. This metadata, which includes abstracts, author affiliations, and citation data, provides a comprehensive overview of the Computer Science research landscape. Due to its high data quality, the D3 dataset has found application in multiple publications [98, 99].

Besides basic metadata such as titles, authors, and publication years provided by DBLP, the D3 dataset is further enriched with additional information that is parsed from PDF documents [97]. As parsing PDF documents is notoriously prone to inaccuracies, the accuracy of the D3 dataset is not guaranteed. Additionally, the D3 dataset only includes publications listed on DBLP, which might not encompass the complete spectrum of Computer Science research.

This thesis uses Version 2.1[2] of the D3 dataset. Two files containing data about papers and authors can be downloaded in GZIP format from the Zenodo repository. Both datasets contain a unique author ID, which can be used to link the author information from the authors dataset to the paper information from the papers dataset.

The columns of both the "papers" and "authors" datasets are thoroughly documented on the D3 project's GitHub repository[3]. *readnext* uses the following features from D3:

- The *paper title* and *author names*. Although these variables are not directly used as features for the Hybrid Recommender, they are provided as additional context to the recommended papers.
- The *publication date*, the *paper citation count*, and the *author citation counts*. These features are used as global document features for the Citation Recommender.
- The *paper abstract*. This feature is used as input to the Language Recommender.
- The *Semantic Scholar ID* and *arXiv ID*. These features serve multiple purposes: First, they assist in merging labels (Section 3.2.2) as well as citation lists and reference lists (Section 3.2.3) for each paper with the D3 dataset. Second, they are used as (external) unique identifiers for each paper. During the inference stage, they determine if the query paper from the user's input is already included in the *readnext* corpus. Finally, the IDs are used to add the corresponding Semantic Scholar and arXiv URLs to the system's recommendations, facilitating immediate user access to the recommended papers.

3.2.2 ArXiv Labels

In order to determine whether the recommendations generated by the Hybrid Recommender are relevant or not, a ground-truth is required. This ground-truth is obtained by using arXiv categories

[1] https://dblp.org/
[2] https://zenodo.org/record/7071698#.ZA7jbC8w2Lc
[3] https://github.com/jpwahle/lrec22-d3-dataset

as labels. The primary reason for this choice is the mass availability in form of the arXiv Kaggle dataset[4], which contains data for over 1.7 million papers from the arXiv database.

The arXiv data is merged with the D3 dataset using the arXiv ID. Since many papers in the D3 dataset do not possess a corresponding arXiv ID, this process results in a significant reduction of the dataset size. Concretely, only $13,764$ papers out of the $100,000$ most cited papers in the D3 dataset have a corresponding arXiv ID. The remaining papers lack essential label information needed for the MAP computation during evaluation and are excluded from the dataset.

Unlike the *ACM Computing Classification System*[5], which contains narrow and nested categories, the arXiv categories are more coarse-grained with fewer and non-nested categories. According to the *arXiv category taxonomy*[6], there are 40 categories within the Computer Science domain. Importantly, one paper is not restricted to a single category but can be assigned to multiple categories. While arXiv distinguishes between primary and secondary classifications, the arXiv Kaggle dataset does not preserve this ordering. Thus, this thesis treats all categories equally.

One advantage of using arXiv categories as labels is that they are classified by the authors themselves. This often results in precise category assignments, as authors, being experts in their research fields, typically classify their papers more accurately than automated systems.

Definition of Relevance

The evaluation results crucially depend on the ground-truth definition for relevance. For this thesis, we define a recommendation to be *relevant* if the query paper and the recommended paper share at least one arXiv category. Conversely, it is classified as *irrelevant* if the query paper and the recommended paper do not share any category.

To illustrate this definition with an example, consider the query paper *Aspect-based Document Similarity for Research Papers* by Ostendorff et al. [95], which is assigned the arXiv categories *cs.CL* (Computation and Language) and *cs.IR* (Information Retrieval). Two candidate papers are considered, the first being *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* by Devlin et al. [70], which is assigned the arXiv label *cs.CL* (Computation and Language). Since there is an overlap with the query labels, this recommendation is considered relevant. The second candidate paper is *Deep Residual Learning for Image Recognition* by He et al. [100], which is assigned the arXiv label *cs.CV* (Computer Vision and Pattern Recognition). Since there is no overlap with the query labels, this recommendation is considered irrelevant.

The above definition is chosen over the alternative of classifying a recommendation relevant if query and recommended paper share *all* categories for the following two reasons: First, the number of papers that share all categories with the query paper is very small, leading to a highly

[4] https://www.kaggle.com/Cornell-University/arxiv
[5] https://dl.acm.org/ccs
[6] https://arxiv.org/category_taxonomy

unbalanced relevant/irrelevant ratio. Second, rewarding recommendations with at least one shared category allows for interdisciplinary recommendations since the query and candidate paper can still differ in multiple other categories.

Recommendation Diversity

As highlighted in Section 2.1.5, non-accuracy-based metrics like diversity, serendipity, novelty, and coverage are often more difficult to quantify than their accuracy-based counterparts. However, these metrics provide essential insights into recommendation quality. To capture some of these aspects, we propose a definition of *recommendation diversity* that is specific to the context of this thesis.

Recommendation diversity is defined as the number of unique arXiv categories in the entire recommendation list. For example, if the recommendation list contains three papers with the arXiv categories *cs.CL*, *cs.IR*, and *cs.CV*, the recommendation diversity is three. If the recommendation list contains three papers with the arXiv categories *cs.CL*, *cs.CL*, and *cs.IR*, the recommendation diversity is two. Since arXiv categories reflect research areas, recommendation diversity can be viewed as an indicator of interdisciplinarity within the recommendations. As recommendations from different research areas are often unexpected, the recommendation diversity is closely related to serendipity.

This definition of diversity yields two insights. First, the diversity of a recommendation list is independent of the ranking order. Thus, the re-ranking process of the Hybrid Recommender does not affect the initial diversity of the candidate recommendations and it is sufficient to evaluate the diversity of the candidate lists generated by the Citation and Language Recommenders.

Second, there is not a perfect inverse correlation between a recommendation list's diversity and its MAP, in which case the additional diversity metric would be redundant. This observation arises because a single paper can belong to multiple research domains. For instance, recommendations for a query paper tagged with *cs.CV* can display both high diversity and high MAP if the suggested papers primarily come from the Computer Vision domain, complemented by papers from various other research areas.

3.2.3 Citation Features

While the D3 dataset contains the number of cited and citing papers for each document, it does not contain lists of all *individual* citations and references. As these are required to compute co-citation analysis and bibliographic coupling scores, an additional source is needed to supplement the dataset. Therefore, data from the Semantic Scholar Academic Graph API[7] is fetched. To increase the rate limit, a private API key can be requested[8]. The API allows for retrieving citation and

[7] https://api.semanticscholar.org/api-docs/graph
[8] https://www.semanticscholar.org/product/api#api-key

reference lists for all papers in the dataset, which are subsequently added as new columns to the base dataset.

A significant limitation of the Semantic Scholar API is its restriction to request responses with a maximum of 1000 items. This leads to information loss for papers with more than 1000 citations or references. Since only a single paper in the corpus cites more than 1000 papers, namely *A Survey of Neuromorphic Computing and Neural Networks in Hardware* by Schuman et al. [101], the bibliographic coupling scores are barely affected by this limitation. However, the co-citation analysis scores are heavily affected since 868 papers are cited more than 1000 times. These frequently cited papers are thus represented with a too weak connection to the related literature.

While this constraint may introduce bias in the dataset, the paper citation count feature of the Citation Recommender might offset its impact. Larger weights can be assigned to paper citation counts for highly cited papers to counteract the truncated co-citation analysis scores.

3.3 Training

The training stage precomputes and stores data to make the inference and evaluation stages faster and more efficient. Since all static and contextual embedding models in this thesis are pretrained, the term *precomputation* stage is more accurate than *training* stage. However, both terms will be used interchangeably to distinguish this phase from the subsequent inference phase.

This section provides details on the training stages for the Citation Recommender (Section 3.3.1) and the Language Recommender (Section 3.3.2). The Hybrid Recommender, which merges data from both recommenders, is not part of the training stage. Instead, it integrates precomputed data from the Citation Recommender and the Language Recommender on-the-fly during inference (Section 3.4).

3.3.1 Citation Recommender

As elucidated in Section 3.1.1, the Citation Recommender uses global document characteristics and citation-based features to generate recommendations. Figure 3.3 illustrates how the publication date, paper citation count, and author citation count are immediately provided by the base dataset, while the co-citation analysis and bibliographic coupling features are derived from the citations and references columns, respectively.

Given two papers from the *readnext* dataset, the co-citation analysis score counts the number of shared citing papers in the citations column. Analogously, the bibliographic coupling score counts the number of shared cited papers in the references column. Figure 3.4 visualizes this process at the example of the bibliographic coupling score.

During training, each document in the *readnext* corpus is iteratively considered as the *query paper*, with all remaining documents acting as *candidate papers*. For each query paper, the pairwise co-

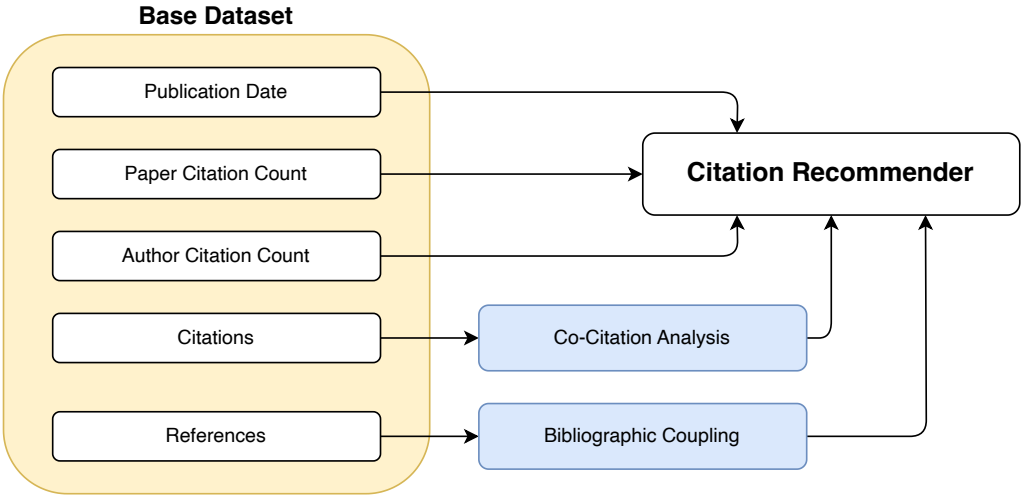


Figure 3.3: Inputs to the Citation Recommender. The publication date, paper citation count, and author citation count are immediately provided by the base dataset. The co-citation analysis and bibliographic coupling features are derived from the citations and references columns of the dataset, respectively.

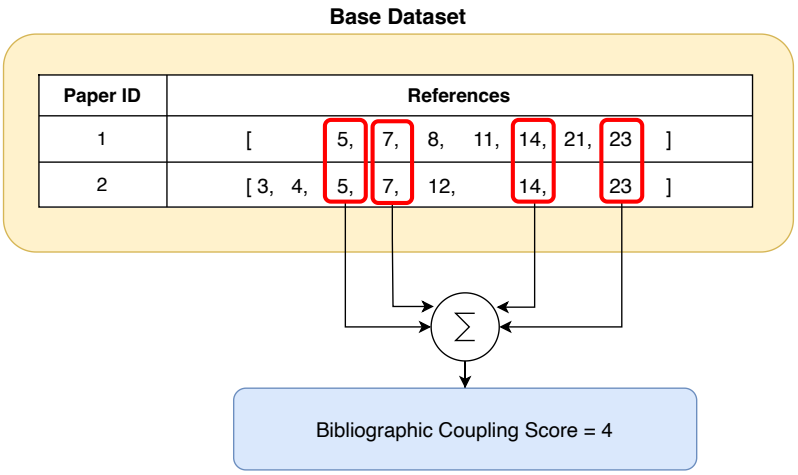


Figure 3.4: Computation of bibliographic coupling scores from the base dataset. The *references* column contains four common items for the papers 1 and 2, resulting in a bibliographic coupling score of 4. For visualisation purposes, numeric paper IDs are utilized as identifiers in the figure, whereas unique Semantic Scholar URLs are used in the actual dataset.

citation analysis and bibliographic coupling scores with all candidate papers are computed and stored in a dataframe with corresponding document IDs. Thus, time complexity during training grows quadratically with the corpus size. For n documents in the dataset, $n^2 - n$ pairwise scores are computed for each of the two features.

To circumvent quadratic growth for the memory and storage requirements, the Citation Recommender stores only the top 100 candidate papers with the highest co-citation analysis and bibliographic coupling scores for each query paper (see Section 3.4.3). Although not storing all pairwise scores to disk conserves memory, this approach does not decrease runtime since identifying the top 100 scores requires calculating all pairwise scores first.

Feature Weighting

The Citation Recommender generates recommendations using a weighted average of the global document characteristics and citation-based features. Therefore, not all features exert the same influence on the Citation Recommender. Instead, users can assign weights to each feature based on their preferences. The weights are normalized to sum up to one, ensuring the results are not affected by the absolute magnitude of the weights. Candidate papers with the best weighted average score are recommended to the user.

A caveat of this approach is that the raw feature values, such as the publication date (represented as a date) and the paper citation count (an integer), are not directly comparable. To aggregate all five features into a single score, a rank-based method is used.

From Scores to Ranks

The Citation Recommender first ranks all candidate papers according to each of the five features individually. The ranking process assigns the top rank 1 to the most relevant candidate paper and increments the rank by 1 for each subsequent paper. Candidate papers with more recent publication dates, higher citation counts, higher co-citation analysis and higher bibliographic coupling scores receive better rankings. Ranks for global document characteristics remain constant across different query papers and must be computed only once. In contrast, ranks for citation-based features are specific to each query paper and must be recomputed for each query paper.

The decision to store only the top 100 candidate papers for the co-citation analysis and bibliographic coupling scores prohibits the computation of exact ranks for all candidates beyond the top 100. Consequently, the ranks for the remaining candidates are uniformly set to 101, implying that there is no distinction between the 101st and all worse-ranked candidate papers. Due to this nonlinearity, outliers in the positive direction with high scores have a stronger impact on the final recommendation ranking than outliers in the negative direction with low scores. For consistency, the same ranking scheme is applied to the publication date, paper citation count, and author citation count features.

From Ranks to Points

The final recommendation ranking could simply be calculated as the weighted average of the individual feature ranks, with lower average ranks leading to earlier recommendations. However,

this method doesn't readily convey whether a particular weighted rank of e.g. 412 is good or bad without knowing the specifics of the ranking system and the number of features involved.

For greater interpretability, the Citation Recommender translates feature ranks into *points*, which are inversely related to the rank: the lower the rank, the higher the point score. More precisely, the points are calculated as the difference between the maximum rank of 101 and the actual feature rank, resulting in a range of 0 to 100 points. These points are then weighted by the feature weights and summed to a weighted points score. Candidates with higher weighted points are considered more relevant and appear earlier in the final recommendation ranking. The point-based system assigns zero contribution to candidate papers beyond the top 100, which is more intuitive than a rank contribution of 101.

The linear scaling of points from 0 to 100 in steps of 1 is chosen to diminish the weighted score's dependence on individual features. Superlinear scaling, like awarding 100 points to the top-ranked candidate, 50 to the next and 10 to the third, excessively emphasizes outliers in a single feature. For instance, a paper that is co-authored by Yoshua Bengio, the most-cited author in the dataset with a citation count of $372,099$, but ranks last in all other features might outperform a paper that ranks third across all features. Hence, although the second paper is likely more relevant to the query paper, it would be recommended later than the first paper.

3.3.2 Language Recommender

In contrast to the Citation Recommender, the Language Recommender uses only a single column of the *readnext* dataset: the paper abstracts. Its objective during training is to compute and store the similarity between any two abstracts in the corpus. To achieve this, the Language Recommender employs one of eight language models to embed each abstract into a vector space. Then, pairwise cosine similarities between all abstract embeddings are computed and stored.

Since each of the eight language models can be selected by the user during inference, embeddings and cosine similarities must be precomputed for all of them during the training stage. As illustrated in Figure 3.5, once the paper abstracts are tokenized, different language models do not interact with each other. Therefore, embeddings and cosine similarities can be efficiently computed in parallel for all language models.

The cosine similarities are used to find abstracts most similar to a given query abstract. For instance, when requested with the task to find the semantically most similar document to the paper *Attention Is All You Need* by Vaswani et al. [69] with BERT as the embedding model, the Language Recommender selects the data containing all precomputed BERT cosine similarities, filters it for the paper ID of *Attention Is All You Need*, and returns the candidate paper with the highest cosine similarity. Similar to the co-citation analysis and bibliographic coupling scores, only the top 100 cosine similarities are stored for each query paper.

For the particular example above, 81 out of $10,000$ papers have cosine similarities above 0.9 with

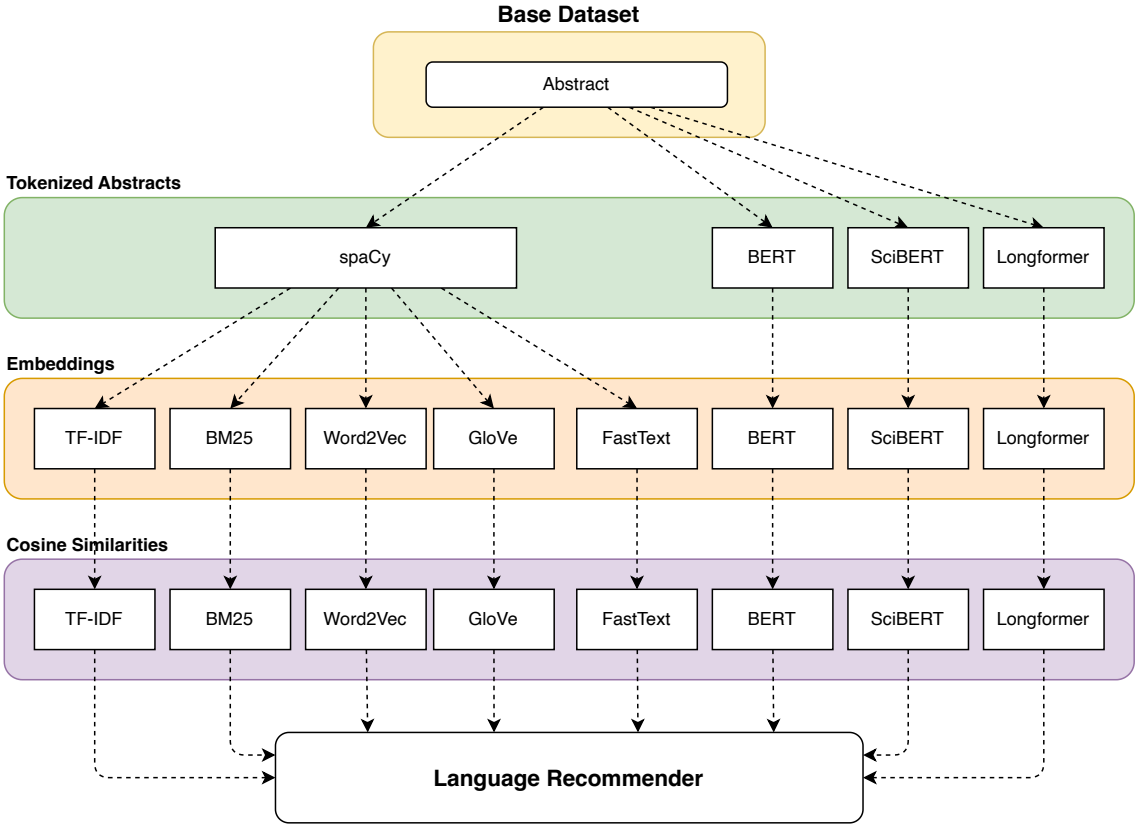


Figure 3.5: Structure of the Language Recommender. First, the paper abstracts are tokenized. BERT, SciBERT and Longformer use a specialized tokenizer, while the remaining models use the `spaCy` [102] tokenizer. Then, each language model embeds the tokenized abstracts into a vector space. Finally, pairwise cosine similarities between all document embeddings are computed for all language models.

the paper *Attention Is All You Need* as measured by BERT. The semantically most similar paper is *Effective Approaches to Attention-based Neural Machine Translation* by Luong et al. [103] with a cosine similarity of 0.967. Using a different language model for the embedding step would result in a different ranking of the candidate papers. The following sections describe the implementation and configuration of each language model in detail.

Keyword-based Models

As explained in Section 2.3.2, TF-IDF [59] and BM25 [61] are keyword-based models that generate sparse vector embeddings, where each embedding dimension corresponds to a single word in the vocabulary. The vocabulary consists of all unique words in the preprocessed text corpus, which in this case is the collection of all paper abstracts contained in the *readnext* dataset. The abstract

embeddings are sparse vectors of the same length as the vocabulary, where each non-negative value represents the TF-IDF or BM25 score of the corresponding word in the abstract.

Tokenization and Preprocessing

Before document embeddings can be computed, the raw abstracts must be tokenized and preprocessed. In case of TF-IDF and BM25, the tokenization process is performed by the `spaCy` [102] library, more specifically by the pretrained `en_core_web_sm` model [9] that is able to tokenize and preprocess English text. `spaCy` splits the text into a list of individual tokens and collects metadata about each token, such as its part-of-speech tag, its lemma, and whether it is a stop word. This information can be used for further text processing steps.

The preprocessing steps by `spaCy` for the TF-IDF and BM25 models include:

- Converting all tokens to lowercase.
- Lemmatizing all tokens. Lemmatization helps to reduce the vocabulary size by reducing a word to its base form, e.g. *running* to *run*.
- Removing non-alphanumeric characters, non-ASCII characters, punctuation, digits, and whitespace.
- Removing stop words. Stop words are words that occur frequently in the English language and are thus considered uninformative for any particular document. Examples of stop words are *the*, *a*, and *is*. The `spaCy` library uses a predefined list of 326 English stop words that do not contribute to the document embeddings.

Figure 3.6 illustrates the tokenization and preprocessing process of `spaCy` on a practical example. Post tokenization, the tokens can be passed to the TF-IDF and BM25 models to compute abstract embeddings.

TF-IDF

The TF-IDF model is implemented with the `scikit-learn` library [60] according to Equation (2.11) for the term frequency and Equation (2.12) for the inverse document frequency. The TF-IDF values for all abstract tokens are computed and stored in a sparse matrix of dimension $|D| \times |V|$, where $|D| = 10,000$ is the number of documents in the training corpus and $|V| = 21,240$ is the size of the vocabulary. In practice, specialized storage formats for sparse matrices are used to reduce memory consumption.

Each row of the matrix represents the TF-IDF document embedding of a single abstract. To keep the connection to the corresponding paper, the embeddings are stored together with their document IDs in a dataframe. Based on this representation, the semantic similarity between any

[9] `https://spacy.io/models/en#en_core_web_sm`

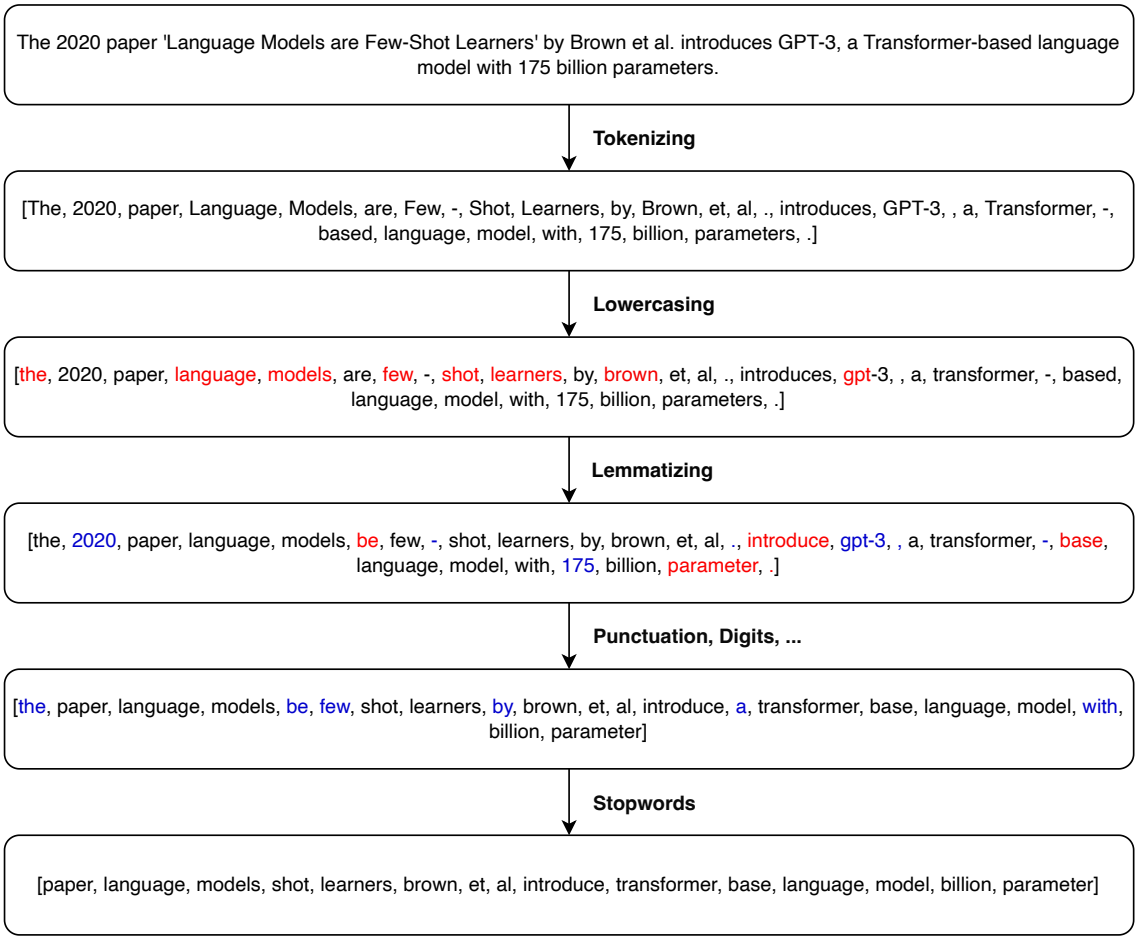


Figure 3.6: Tokenization and text preprocessing with `spaCy`. Red indicates tokens that were modified from the previous step, while blue indicates tokens that are removed in the next step. First, the input sentence is tokenized into a list of individual tokens. Each token is then lowercased, lemmatized, and stripped of non-alphanumeric characters, punctuation, digits, and whitespace. Finally, stop words are removed. The preprocessed tokens are used by language models to compute document embeddings.

two abstracts can be computed by looking up the corresponding rows in the dataframe by their IDs and calculating the cosine similarity between their embeddings.

BM25

The BM25 model is implemented in its BM25+ variant with Equation (2.18) for the term frequency and Equation (2.15) for the inverse document frequency. The free parameters are set to $k_1 = 1.5$, $b = 0.75$, and $\delta = 1$ as implemented in the `rank_bm25` Python library [104].

The vocabulary size and thus the dimensions of the BM25 document embeddings are identical to the dimensions of the TF-IDF document embeddings since the vocabulary is determined by the underlying text corpus and not by any of the two models. Similarly, the cosine similarities are computed in the same fashion as for the TF-IDF model.

Static Embedding Models

Static embedding models, such as Word2Vec [56], FastText [65], and GloVe [64], generate fixed-length dense vector embeddings for each word in the vocabulary. As for the keyword-based models, tokenization is performed by the `spaCy` library with the same preprocessing pipeline. The processed tokens are subsequently passed to each of the three static embedding models to compute document embeddings for all abstracts in corpus. Within *readnext*, Word2Vec, FastText, and GloVe are implemented via their `gensim` interface [105] using pretrained models

Token embeddings within the same document are averaged to compute document embeddings. As before, document embeddings are stored together with their corresponding document IDs in a dataframe. Document similarities are again computed as the cosine similarity between any pair of document embeddings.

Word2Vec For Word2Vec, we implemented the *word2vec-google-news-300* model. This model is pretrained on the Google News Corpus [56] with approximately 100 billion words. The embedding dimension is set to 300. The model is, again provided by the from the *gensim-data* GitHub repository[10]. The file size of the downloaded model is 3.64 GB.

FastText We used the FastText model from Grave et al. [106], pretrained on English Wikipedia and the Common Crawl corpus. As for Word2Vec, the embedding dimension is set to 300. Specific to the FastText model, the character n-gram length is set to 5. The pretrained model can be downloaded from the FastText website[11]. The size of the downloaded model is 7.24 GB.

GloVe For GloVe, we opted for the *glove.6B.300d* model, pretrained on Wikipedia 2014 and Gigaword 5 [64] with around 6 billion tokens and a vocabulary size of $400,000$. The embedding dimension is again set to 300. The pretrained word embeddings can be downloaded from the Stanford NLP website[12]. The size of the downloaded embeddings is 1.04 GB.

Contextual Embedding Models

Models such as BERT [70], SciBERT [71], and Longformer [72] are contextual embedding models that produce dense vector representations of fixed length for each token in the input text. The

[10] `https://github.com/RaRe-Technologies/gensim-data`
[11] `https://fasttext.cc/docs/en/crawl-vectors.html#models`
[12] `https://nlp.stanford.edu/projects/glove/`

distinctive characteristic of these representations is their contextual nature: the vector representation of a token is influenced by the surrounding context in which the token appears. Thus, unlike static embedding models that map each word to a single embedding vector, contextual embedding models generate different vector representations for polysemous words depending on their meaning in the given context.

Since BERT, SciBERT and Longformer are all based on the Transformer architecture [69], they are referred to as *Transformer models* in the following.

Tokenization for Transformer Models

Within *readnext*, Transformer models are implemented using the open-source `transformers` library [107]. This library provides a unified interface to numerous pretrained LLMs. `transformers` is actively developed and maintained by Hugging Face[13].

In contrast to keyword-based and static embedding models that use `spaCy` for tokenization, the text preprocessing steps for Transformer models are tailored to each model individually, as it is crucial that the input format during inference matches the one used during pretraining. Whereas `spaCy` produces *text* tokens, Transformer tokenizers generate integer token *IDs* that correspond to the indices of the tokens in the pretrained model's vocabulary. The subsequent paragraphs delineate the tokenization process for each Transformer model in detail.

BERT

We employed the *bert-base-uncased* version of BERT pretrained on the BooksCorpus (800 million words) and English Wikipedia (2,500 million words). The *bert-base-uncased* model can be accessed via the Hugging Face Model Hub[14].

BERT employs the *WordPiece* tokenizer [79], a subword tokenization method that initially tokenizes at whitespace and punctuation. If the resultant tokens are not found in the model's vocabulary, they are further split into smaller subwords until all word pieces belong to the vocabulary. Subwords are prefixed with `##` to distinguish them from regular words. The vocabulary of the *bert-base-uncased* model comprises roughly $30,000$ tokens [70].

In BERT's input representation, special tokens play a pivotal role. The `[CLS]` token starts each input sequence and acts as a collective representation for classification tasks. The `[SEP]` token separates different sentences or sequence pairs. For single-sequence tasks, it is attached at the end of the sequence, while for two-sequence tasks, it is added at the end of both sequences. The `[PAD]` tokens are utilized to pad shorter sequences to a common length, and `[UNK]` tokens represent words not found in the vocabulary after WordPiece tokenization. The `[MASK]` tokens are used for the masked language model objective during pretraining [70].

[13] `https://huggingface.co/`
[14] `https://huggingface.co/bert-base-uncased`

The maximum input length for BERT is 512 tokens, inclusive of the special tokens. This limitation becomes critical in the context of this thesis, as some paper abstracts in the training corpus exceed this limit. Beltagy et al. [72] elucidate common workarounds to manage this issue. The simplest strategy is to truncate all abstracts to a maximum length of 512 tokens [108]. However, this strategy might discard potentially relevant information. An alternative strategy is to divide the abstracts into smaller sequences with maximum length of 512 tokens, processing each sequence separately, and combining the resulting sequence embeddings [109]. Although this strategy retains all information, it is computationally expensive.

Given that only 0.58% of all abstracts in the *readnext* corpus exceed the maximum sequence length of 512 tokens, we decided to truncate longer abstracts to this maximum length. Conversely, shorter abstracts are padded with `[PAD]` tokens to a length of 512 tokens to ensure that all tokenized abstracts are of the same length.

BERT generates token embeddings of dimension 768. The document embedding for the entire abstract is computed as the mean of all token embeddings within the abstract. The resulting document embeddings are stored alongside their corresponding document ID in a dataframe from which the pairwise cosine similarities can be computed.

SciBERT

For SciBERT, we selected the *scibert_scivocab_uncased* version, accessible via the Hugging Face Model Hub[15]. While SciBERT and BERT share the same architecture, they differ in their pretraining corpus and vocabulary. SciBERT is pretrained on a large corpus of scientific text consisting of 1.14 million papers from the Semantic Scholar corpus (amounting to 3.17 billion tokens) [71]. The corpus is composed of papers from a variety of scientific disciplines, including Computer Science, Physics, and Biomedicine.

Consequently, SciBERT's vocabulary is tailored to scientific text and includes a significant number of scientific terms that are not part of BERT's vocabulary. Therefore, scientific terms that would be tokenized into multiple subwords or even into `[UNK]` tokens by BERT's tokenizer can often be tokenized into a single subword by SciBERT's tokenizer, resulting in a more accurate representation of the input text. The vocabulary size of the *scibert_scivocab_uncased* model is set to $30,000$ tokens to align with the vocabulary size of the *bert-base-uncased* model [71].

The tokenization process and the special tokens in SciBERT are identical to those in BERT. SciBERT also uses WordPiece tokenization, adds the same special tokens to the input sequence, has the same maximum sequence length of 512 tokens, and generates token embeddings of dimension 768. The document embedding computation is also identical to BERT: the document embedding is computed as the mean of all token embeddings within the abstract.

[15] `https://huggingface.co/allenai/scibert_scivocab_uncased`

Longformer

We opted for the *longformer-base-4096* version of the Longformer model, again provided by the Hugging Face Model Hub[16]. Whereas BERT is pretrained with MLM and NSP tasks, Longformer relies solely on the MLM task. Unlike BERT, Longformer's pretraining does not start from scratch; instead, it uses the weights of the RoBERTa model [110] as initialization. The Longformer model is pretrained on a compiled corpus of long documents comprising subsets of the BooksCorpus [111], the English Wikipedia, the Realnews dataset [112] and the Stories corpus [113].

While the BERT tokenizer truncates input sequences to a maximum length of 512 tokens, the Longformer tokenizer can handle input sequences of up to $4,096$ tokens due to its sliding window attention mechanism described in Section 2.3.5. By default, Longformer's sliding window attention mechanism uses a size of 512 tokens. As none of the abstracts in the *readnext* corpus exceed a length of $4,096$ tokens, truncation is not required for the Longformer model. Similar to BERT, all abstracts are padded with special `<pad>` tokens to a common length of $4,096$ tokens.

The principal methodological difference to BERT's tokenizer is that Longformer's tokenizer uses Byte-Pair Encoding (BPE) [114] instead of WordPiece tokenization [79]. Consequently, the Longformer tokenizer employs a different set of special tokens. The start and end tokens are `<s>` and `</s>`, respectively. Unlike the BERT tokenizer, which uses `##` to indicate subwords, the Longformer tokenizer adds the special token `Ġ` to represent spaces. In Longformer, subwords are distinguished from regular words by not being prefixed with a space token.

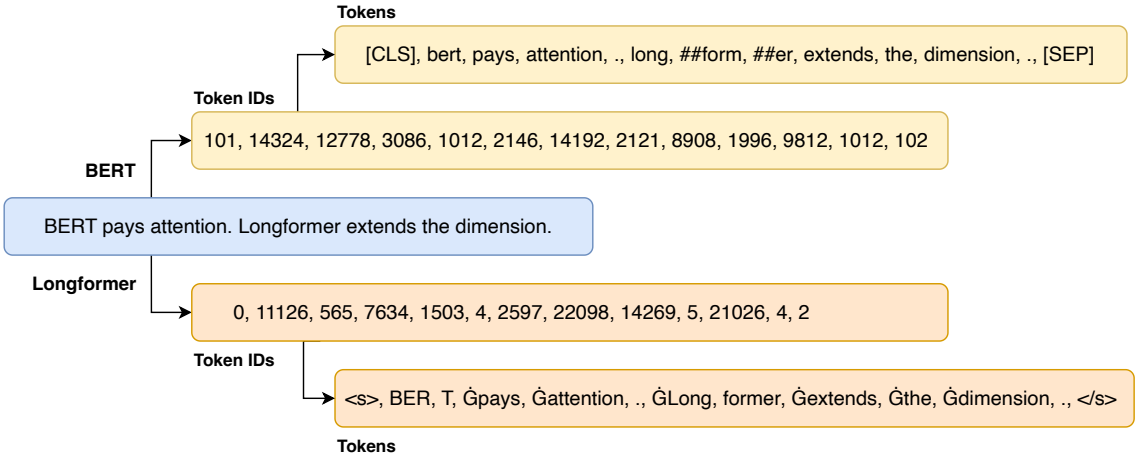


Figure 3.7: Tokenization example for BERT and Longformer. Both tokenizers produce token IDs rather than text tokens. Each token ID corresponds to a unique word piece in the vocabulary. BERT uses the `##` symbol to indicate subwords, whereas Longformer uses the `Ġ` symbol for whitespace.

Table 3.1 illustrates the conceptual differences between BERT and Longformer, whereas Figure 3.7

[16] `https://huggingface.co/allenai/longformer-base-4096`

Feature	BERT	Longformer
# Parameters	110M (BERT-base)	125M (RoBERTa-base)
Embedding Dimension	768	768
Max. Sequence Length	512	4096
Training Data	BooksCorpus, Wikipedia	Long Documents (BooksCorpus, Wikipedia, Realnews, Stories)
Training Task	MLM & NSP	MLM
Weight Initialization		RoBERTa
Attention Mechanism	Full Self-Attention	Sliding Window Attention
Attention Window Size	512	512
Tokenization Algorithm	WordPiece	Byte-Pair Encoding
Start Token	`[CLS]`	`<s>`
End Token	`[SEP]`	`</s>`
Pad Token	`[PAD]`	`<pad>`
Subword Prefix	`##`	
Space Token		`Ġ`

Table 3.1: Model properties of BERT and Longformer. Longformer initializes its weights from the RoBERTa model encompassing 125 million parameters compared to 110 million parameters for BERT-base. The Longformer tokenizer uses Byte-Pair Encoding instead of WordPiece tokenization and a different set of special tokens. The most significant difference between the two models is that Longformer uses a sliding window attention mechanism instead of full self-attention to better process long documents. Accordingly, Longformer is pretrained on a corpus of exclusively long documents and can handle input sequences of up to $4,096$ tokens.

compares their tokenization processes on the example sentence pair *BERT pays attention. Longformer extends the dimension.* Note that, unlike traditional tokenization approaches, Transformer models do not remove punctuation from the input text. Instead, punctuation is treated as a separate token.

Like BERT and SciBERT, Longformer aggregates token embeddings into document embeddings by averaging all token embeddings within the abstract. Cosine similarities between document embeddings are computed in the same manner as for previous models.

3.4 Inference

The *inference* stage aims to generate personalized recommendations tailored to the user's query paper. The inference process is designed to operate iteratively: Users can select one of the recommended papers as their new query paper, prompting the system to generate recommendations for the new query next. This iterative approach forms a document sequence of queries and recommendations. At any point in this sequence, users can adjust input parameters to better suit their preferences.

Before starting the inference process, preparatory setup steps are required to ensure the recommender system has access to all necessary data. These include assembling the base dataset,

downloading pretrained language models or word embeddings, and precomputing the co-citation analysis, bibliographic coupling, and cosine similarity scores in the preceding training stage. A detailed setup guide is provided by the *readnext* documentation[17].

3.4.1 User Interface

Input Parameters

To generate customized recommendations, the system requires three input parameters: a query paper identifier, feature weights for the Citation Recommender, and a language model for the Language Recommender.

The query paper serves as the anchor of the recommendation process, as all recommendations are tailored to this paper. Query papers might be ones the user is currently reading, recommended by a colleague, or found through a search engine. There are four options to specify the query paper:

- The *Semantic Scholar URL*, viewed in a web browser's address bar on the Semantic Scholar[18] site.
- The *Semantic Scholar ID*, a 40-character hexadecimal string appearing after the last forward slash in the Semantic Scholar URL.
- The *arXiv URL*, observed in a web browser's address bar on the arXiv[19] site.
- The *arXiv ID*, a sequence starting with four numbers, followed by a dot, then another five numbers, found after the final forward slash in the arXiv URL.

Since all identifiers are rooted in the Semantic Scholar or arXiv databases, *readnext* currently supports inference for query papers present on either of these platforms.

The second input encompasses feature weights for the Citation Recommender. These weights, defined by numerical values (either as floats or integers), apply to the publication date, paper citation count, author citation count, co-citation analysis score, and bibliographic coupling score. They signify each feature's relative importance in the Citation Recommender. As stated in Section 3.3.1, the weights are normalized with the L1 norm, ensuring the results are not affected by the absolute magnitude of the weights.

The third input designates the language model for the Language Recommender, with choices like TF-IDF, BM25, Word2Vec, FastText, GloVe, BERT, SciBERT, and Longformer, detailed in Section 3.3.2. The selected language model is used to compute the semantic similarity between the abstracts of the query paper and all candidate papers.

[17] `https://joel-beck.github.io/readnext/setup`
[18] `https://semanticscholar.org`
[19] `https://arxiv.org`

While specifying the query paper is mandatory, both feature weights and the language model are optional. If feature weights or language model are not provided, the system defaults to the best performing parameters identified in the evaluation stage (Chapter 4). However, users are encouraged to explore various configurations to find the one best suited to their needs.

Output Format

The *readnext* function returns a result object containing data about various components and intermediate results of the recommendation process. Above all, this object comprises four recommendation lists with 20 recommended papers each. The recommendation lists are presented in a tabulated format, where rows represent individual recommendations. The columns provide additional information such as the candidate paper's title, author, publication date, arXiv categories, and URLs for the corresponding Semantic Scholar and arXiv websites.

Depending on whether the Citation Recommender or the Language Recommender determines the recommendation list's ranking order, the table format differs slightly.

1. If the table contains the Citation Recommender candidates or the L $\rightarrow$ C hybrid model's recommendations, the ranking order is determined by the *weighted points score* of the Citation Recommender. Candidates are sorted by their score in descending order, with the highest-scoring recommendation first.
2. If the table contains the Language Recommender candidates or the C $\rightarrow$ L hybrid model's recommendations, the ranking order is determined by the *cosine similarity* between the abstract embedding of the query paper and the abstract embedding of the candidate paper. Candidates are sorted by their cosine similarity in descending order, placing the most similar candidate paper at the top.

3.4.2 Seen vs. Unseen Papers

Inference applies to papers included in the *readnext* dataset (termed *seen* papers) and those not included (termed *unseen* papers). Although the user interface remains the same for both categories, their underlying inference methodologies differ markedly. The subsequent sections elucidate the inference procedure for both seen and unseen papers.

Input Validation

Initially, the user input is parsed and validated through the `pydantic` library[20]. Invalid inputs cause the program to exit early, displaying an informative error message.

The following checks are performed:

- Exactly one of the four available query paper identifiers is provided.

[20] `https://docs.pydantic.dev/latest/`

- When specified, the Semantic Scholar ID is a 40-character hexadecimal string.
- When specified, the Semantic Scholar URL starts with `https://www.semanticscholar.org/paper/`.
- When specified, the arXiv ID starts with four digits followed by a dot followed by five more digits.
- When specified, the arXiv URL starts with `https://arxiv.org/abs/`.
- If feature weights are provided, they are non-negative numeric values.
- If a language model is provided, it belongs to one of the eight implemented models.

Post-validation, the query paper identifier is used to determine whether the query paper is included in the *readnext* dataset, i.e., whether it is a seen or an unseen paper.

Inference for Seen Papers

For seen query papers, the inference process is fast and efficient, as it mainly involves retrieving precomputed data from the training phase. Figure 3.8 illustrates the inference process for seen query papers schematically.

First, the query identifier provided by the user is converted into an internal document ID, which then serves as a key to access precomputed data from the data storage. The lookup process is illustrated in Figure 3.9 at the example of co-citation analysis scores.

Second, the Citation Recommender uses the query paper's identifier to filter global document characteristics, precomputed co-citation analysis scores, and bibliographic coupling scores. In this way, only the relevant information for the query paper is retrieved. Next, point scores for each individual feature are computed as described in Section 3.3.1. The individual points are weighted by the user-specified feature weights and summed to obtain the weighted points score. The top $n = 20$ candidates with the highest score are returned as the Citation Recommender's candidate list.

Third, the Language Recommender filters the precomputed cosine similarities for the user-specified language model by the query paper's identifier. The top $n = 20$ candidates with the highest cosine similarity are returned as the Language Recommender's candidate list.

Finally, the Hybrid Recommender forms the final recommendation rankings from the candidate lists of the Citation Recommender and the Language Recommender. For the C $\rightarrow$ L hybrid model, the *readnext* corpus is first reduced to the Citation Recommender's candidates. The Language Recommender then sorts these papers by the cosine similarity between the query paper's abstract and the candidate papers' abstracts. Conversely, for the L $\rightarrow$ C hybrid model, the *readnext* corpus is reduced to the Language Recommender's candidates and sorted by the weighted points score of

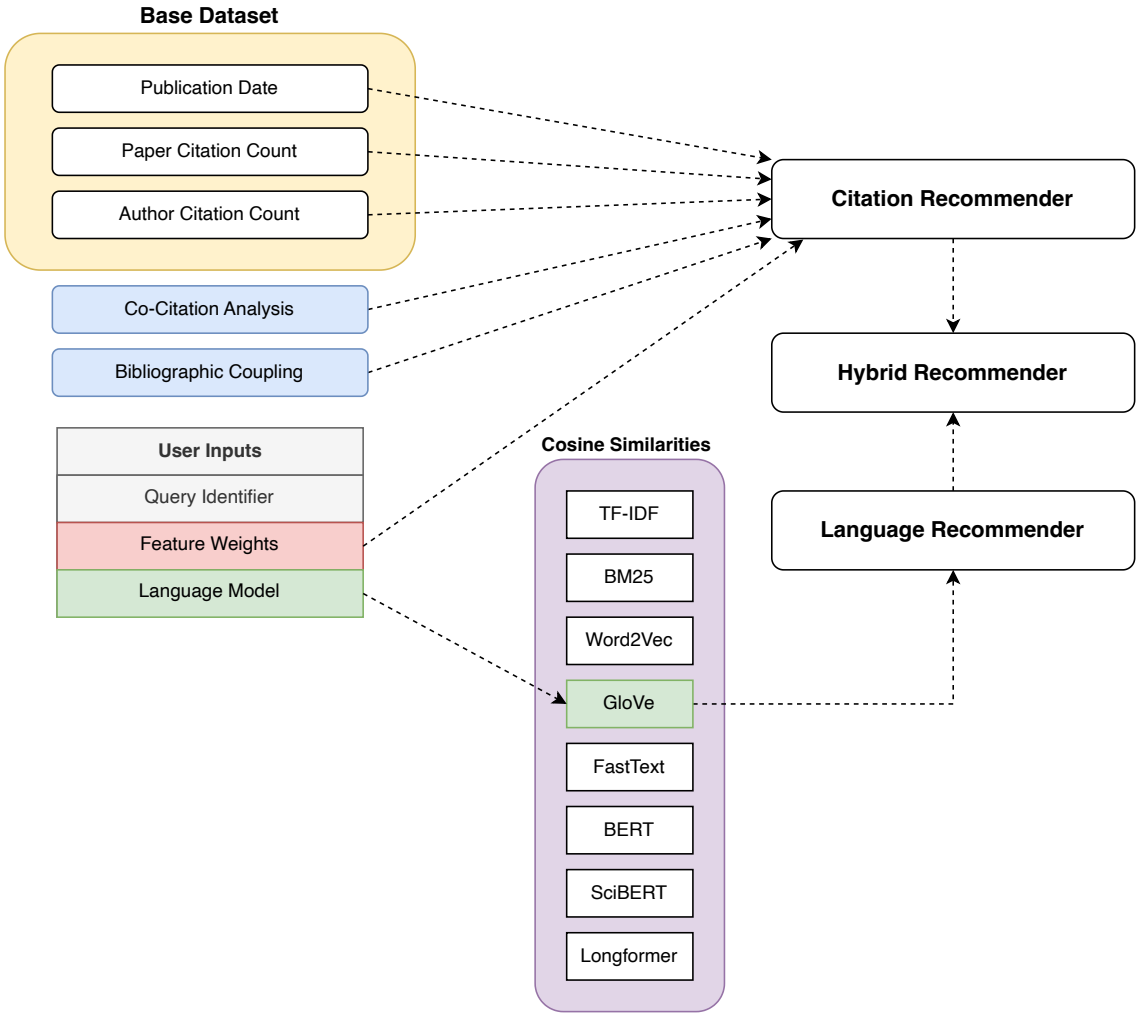


Figure 3.8: Inference process for seen query papers. The Citation Recommender extracts global document features from the base dataset. Co-citation analysis scores and bibliographic coupling scores are retrieved from the precomputed data. The Language Recommender uses the precomputed cosine similarities. The Hybrid Recommender combines the candidate lists from the Citation Recommender and the Language Recommender to form the final recommendation ranking.

the Citation Recommender. The final and candidate rankings of both hybrid orderings are returned to the user.

Inference for Unseen Papers

For unseen query papers, the inference process is more complex, as query-specific data cannot be retrieved from the training stage and must be computed on the fly. The inference process for

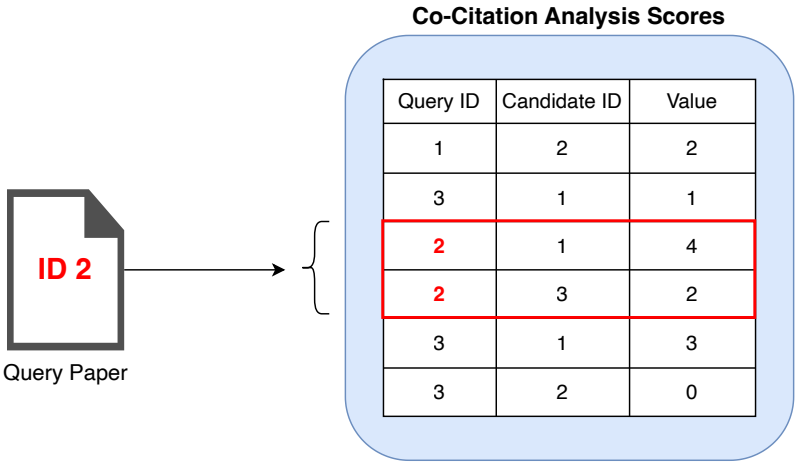


Figure 3.9: Lookup process for seen papers during inference using the example of co-citation analysis scores. The query paper's identifier is used to retrieve the relevant information from the precomputed scores. The scores are stored with their corresponding query and candidate paper IDs in long data format. Only the subset with a matching query paper ID is considered, as indicated by the red rectangle. The query-specific co-citation analysis scores are then ranked and converted to points.

unseen query papers is illustrated in Figure 3.10.

First, the user-provided query identifier is converted to a valid Semantic Scholar URL or arXiv URL to fetch the query paper's metadata from the Semantic Scholar Academic Graph API, including the paper's title, authors, abstract, citations, and references. Information regarding the query's global document characteristics — like the publication date, paper citation count, and author citation count — is not required. This is because the candidate rankings for these features do not depend on the query paper's characteristics. As arXiv categories cannot be obtained from the Semantic Scholar API, label information is not available for unseen papers during inference.

Next, query-specific co-citation analysis and bibliographic coupling scores are computed by matching the query paper's citations and references with the corresponding lists of all candidates, as detailed in Figure 3.4. Similar to the inference process for seen papers, the Citation Recommender computes weighted point scores for all candidates and returns the top $n = 20$ candidates as the Citation Recommender's candidates.

The Language recommender first tokenizes and embeds the query paper's abstract using the user-specified language model. The candidate embeddings can be retrieved from the precomputed data. Then, the query-specific cosine similarities between the query embedding and all candidate embeddings are computed. The subsequent steps mirror the inference process for seen papers, including the re-ranking performed by the Hybrid Recommender.

3.4.3 Performance Considerations

Performance has been a key focus from the start of this project. The value of a highly accurate recommender system diminishes if users face long waiting times to receive recommendations. For instance, if a single inference run took 10 minutes, it would be impractical to use the system for exploratory purposes, such as comparing different model configurations. In addition, searches with less personalized but faster systems could be refined several times in the same time frame, potentially yielding better results.

Although optimizing the computational complexity of both training and inference stages is desirable, the inference stage is more critical This is because the training stage is performed only once, while the inference stage is executed each time a user requests recommendations. Therefore, in case of trade-offs between training and inference performance, the inference stage should be prioritized.

The main bottleneck for both training and inference runtime is the size of the training corpus, creating a trade-off between performance and coverage breadth. On the one hand, a larger training corpus increases the coverage of the recommendations, minimizing the risk of excluding relevant papers from the corpus. On the other hand, as the corpus size increases, many operations during both training and inference become more complex and time-consuming.

Corpus Size & Computational Complexity

For seen query papers, the inference runtime is marginally, or not at all, affected by the number of candidate papers in the dataset. This is because the only change when doubling the corpus size is a lookup within a larger dataframe. Since lookups based on hash tables are run in constant time, the time complexity of this operation remains largely unaffected by the corpus size.

In contrast, inference runtime for unseen papers grows linearly with the corpus size. This is due to the need to compute query-specific co-citation analysis and bibliographic coupling scores as well as cosine similarities between the query paper and all candidate papers on-the-fly, with the number of candidate papers equaling the corpus size. Thus, when the corpus size doubles, the number of pairwise scores that must be computed also doubles, resulting in a linear increase in runtime.

The most significant effect of the corpus size on the time complexity is during training. Pairwise co-citation analysis scores, bibliographic coupling scores, and cosine similarities need to be computed for *every* pair of papers in the *readnext* dataset. Consequently, the time complexity of these operations grows quadratically with the corpus size. In addition, memory and storage requirements also grow quadratically as the pairwise scores are saved to disk during training and loaded into memory during inference.

Strategies for Performance Optimization

To tackle runtime issues, the base dataset is limited to the top $10,000$ most cited papers from the D3 dataset. This corpus size strikes a balance between performance and coverage: Inference time is reduced to less than two seconds for seen query papers and between 10 and 30 seconds for unseen query papers. In the latter case, the most time-consuming aspect is loading large pretrained language models or word embeddings into memory, a process that is difficult to optimize without resorting to smaller and less accurate models.

At the same time, the corpus size is large enough to cover a wide range of popular papers across various Computer Science disciplines. The cut-off threshold for the $10,000$ most cited papers stands at 141 citations; papers cited more often are included in the corpus. This way, even lesser-known papers, especially those outside a researcher's field of expertise, can still be recommended.

To save on memory and storage, only the top 100 candidate papers with the highest co-citation analysis, bibliographic coupling, and cosine similarity scores are stored for each query paper. Section 3.3.1 highlights how this limitation affects the Citation Recommender's scoring mechanism. By storing a constant number of candidate papers for each query paper, the growth of memory and storage requirements is linear rather than quadratic.

When combined with a corpus size limit of $10,000$ papers, the storage requirements remain manageable. For instance, the precomputed cosine similarities require 5 - 7 MB of storage space, depending on the language model. In comparison, increasing the corpus size to $100,000$ papers while storing *all* pairwise scores would inflate these files to 0.5 - 0.7 GB. Further scaling to $1,000,000$ papers would necessitate 50 - 70 GB of storage space. Thus, building a recommender system that scales to millions of papers requires a more complex storage architecture or a more sophisticated approach of handling the $O(n^2)$ complexity, like a preceding clustering step as proposed by Kanakia et al. [38].

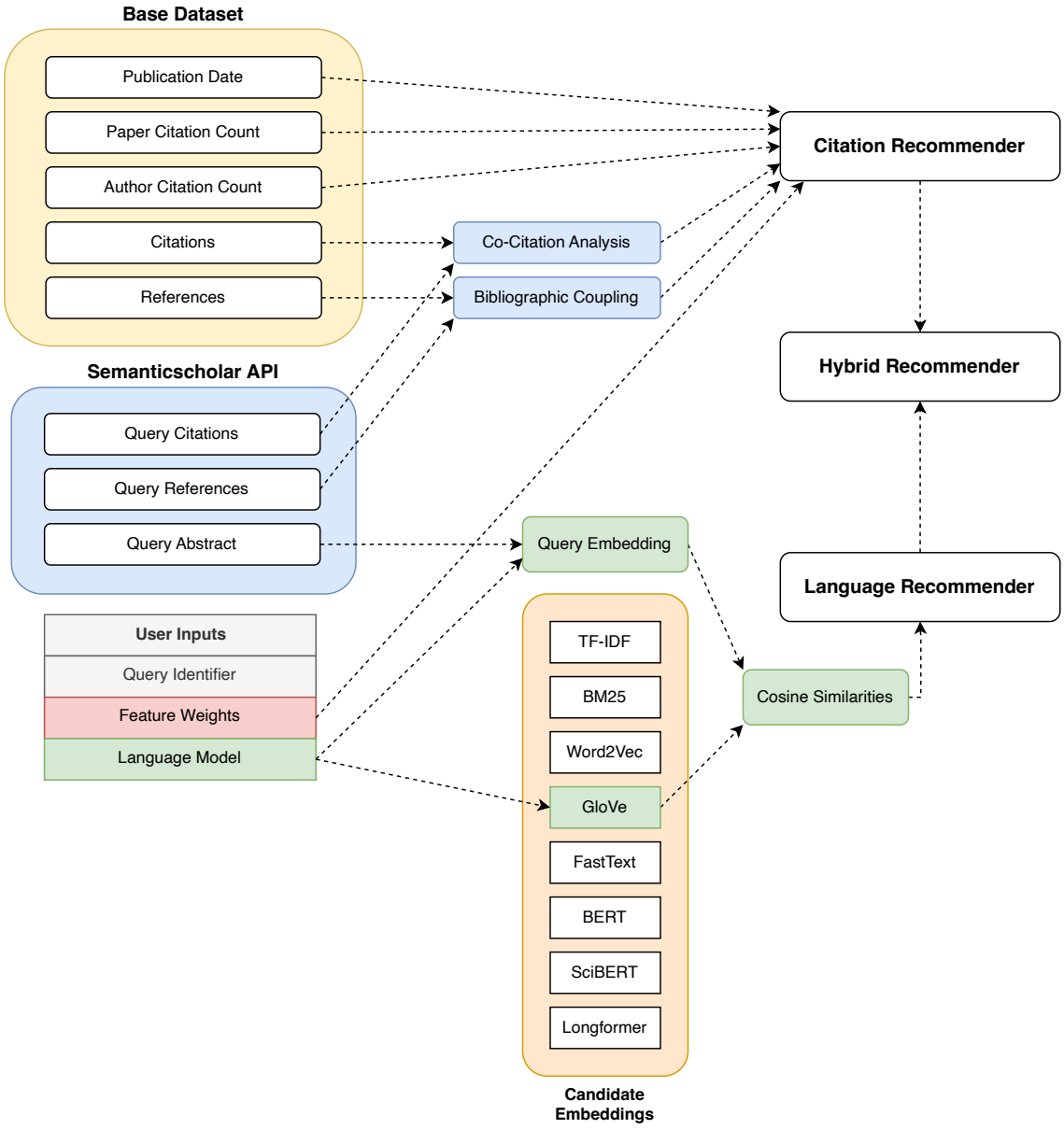


Figure 3.10: Inference process for unseen query papers. Unlike the inference process for seen papers as detailed in Figure 3.8, citations, references, and the abstract for the query paper are retrieved from the Semantic Scholar API. The Citation Recommender computes query-specific co-citation analysis scores and bibliographic coupling scores using global document characteristics and citation-based features for the candidates from the base dataset. The Language Recommender derives query-specific cosine similarities based on the query embedding and the stored candidate embeddings. Similar to the process for seen query papers, the Hybrid Recommender combines the candidate lists from both the Citation and Language Recommenders to generate the final recommendation ranking.

Chapter 4

Evaluation

This chapter addresses **RT 3** and **RT 4** as defined in Section 1.2. Section 4.1 describes the data-driven evaluation strategy used to assess the efficacy of the Hybrid Recommender. Subsequently, Section 4.2 presents and discusses the results of the evaluation. As a first evaluation step, Section 4.2.1 establishes a baseline performance against which all subsequent results are compared. Following this, Section 4.2.2 explores the impact of different feature weights on the Citation Recommender's performance. In a similar vein, Section 4.2.3 investigates the effect of various language models on the Language Recommender's performance. Lastly, Section 4.2.4 assesses whether the hybridization strategy influences the MAP or the recommendation diversity of the Hybrid Recommender.

4.1 Evaluation Strategy

This section begins with a discussion of the evaluation requirements and challenges. Then, it introduces the proposed evaluation strategy and explains the rationale behind its design. Finally, each step of the evaluation process is described in detail.

Evaluation Requirements

The evaluation strategy must satisfy the following conditions:

- It must have the capacity to address **RT 3** and **RT 4** in a comprehensive manner. This includes the ability to determine the optimal parameters for the Citation Recommender, the Language Recommender, as well as the best hybridization order.
- The results should be objective and reproducible. To achieve this, the evaluation must be systematic, data-driven, and independent of any prior knowledge.
- The process must be computationally manageable considering the available resources and time constraints.

To set the stage and clarify the rationale behind our assessment method, we first describe the straightforward but computationally infeasible approach. The simplest way to evaluate the Hybrid Recommender's performance is to conduct an exhaustive grid search over all conceivable input combinations. More specifically, we would need to: (i) construct an evaluation dataset where each row corresponds to a combination of query paper from the base dataset, all possible feature weights, and all available language models, (ii) conduct one inference call for each row, and (iii) calculate the AP for each candidate and final ranking. Those feature weights and language models yielding the highest MAP scores across all inference runs are selected as the optimal parameters.

This approach encounters both computational and conceptual challenges:

1. The feature weights are numerically unbounded since each weight can be set to any real non-negative value. Thus, there are infinitely many combinations of feature weights we would need to evaluate. Further, even a large finite subset of feature weights may render the search space too large to make an exhaustive grid search computationally feasible.

2. Using query documents from the *readnext* dataset for inference only allows the assessment of in-sample performance for seen documents. Evaluating the generalization performance for new and unseen query papers is not possible with this approach. However, inference for unseen query papers is the primary use case of the recommender system, particularly in the long run with more and more papers being published in the future. Thus, quantifying the generalization performance is essential.

Proposed Evaluation Strategy

To overcome the aforementioned challenges, this thesis' evaluation strategy incorporates the following components:

1. **Randomized Search for Feature Weights**: Instead of an exhaustive grid search, a randomized search is utilized for the feature weights. The search space is confined to non-negative integer values to enhance the interpretability of the weight vectors.

2. **Data Splitting**: A traditional random split of the data into training, validation, and test sets is adopted. The split ratio is set such that the training set includes 80%, and the validation and test sets each contain 10% of the data. With a total of $10,000$ documents in the *readnext* corpus, this results in $8,000$ documents for training and $1,000$ documents for validation and testing, respectively. Note that this split is only relevant for the evaluation process. When using the recommender system for inference, recommendations can be drawn from the entire corpus of $10,000$ documents.

3. **Training Set**: The training set serves two purposes. First, it is used to train the keyword-based language models TF-IDF and BM25 with all remaining models being used in their pretrained form. Second, all candidate papers, and thus all potential recommendations, are drawn from

the training set.

4. **Validation Set**: The validation set is employed to conduct the randomized search for the best feature weights of the Citation Recommender. This includes both finding appropriate ranges for the feature weights as well as specific candidate weight vectors within these ranges.

5. **Test Set**: The final performance evaluation that all results presented in Section 4.2 are based on is conducted on a separate test set. This separation ensures an unbiased assessment of the generalization performance for new and unseen query papers.

The subsequent section elaborates on the three-step data-driven evaluation strategy of this thesis, visualized in Figure 4.1 and Figure 4.2. The number of sampled feature weight vectors (n), sampled input combinations (N), and selected candidates (k) are chosen to balance computational feasibility with statistical validity. Generally, higher values for n, N, and k lead to more accurate results but in return increase the computational complexity of the evaluation process.

Step 1: Finding Feature Weight Ranges

The first step aims to identify appropriate ranges for the feature weights of the Citation Recommender on the validation set. Its goal is to discover ranges that are neither too narrow nor too wide. Both extremes lead to suboptimal results: Overly narrow ranges lead to a highly restricted search space, which may exclude the optimal feature weights. Conversely, overly wide ranges may render the process computationally intractable. Further, the selected weights from a very broad search space may still be far from the optimal weights.

The iterative search proceeds as follows:

1. Initialize all feature weight ranges with the integer interval $[0, 5]$.

2. Sample $n = 50$ random weight vectors and construct all combinations with the 1000 query documents in the validation set and the 8 language models, yielding an input space of $400,000$ values.

3. Randomly sample $N = 1000$ input combinations, ensuring an average of 20 samples for each weight vector. Compute the marginal MAP for all weight vectors, averaged over all query documents and language models, and analyze the 20 top-performing candidates. If any weight within those vectors is at the boundaries of the current ranges, the ranges are expanded, as the optimal weights might lie outside the current ranges. Subsequently, the sampling and evaluation steps are repeated with the adjusted ranges. Conversely, if none of the weights are at the boundaries, the current ranges are deemed appropriate and the process moves to step 2.

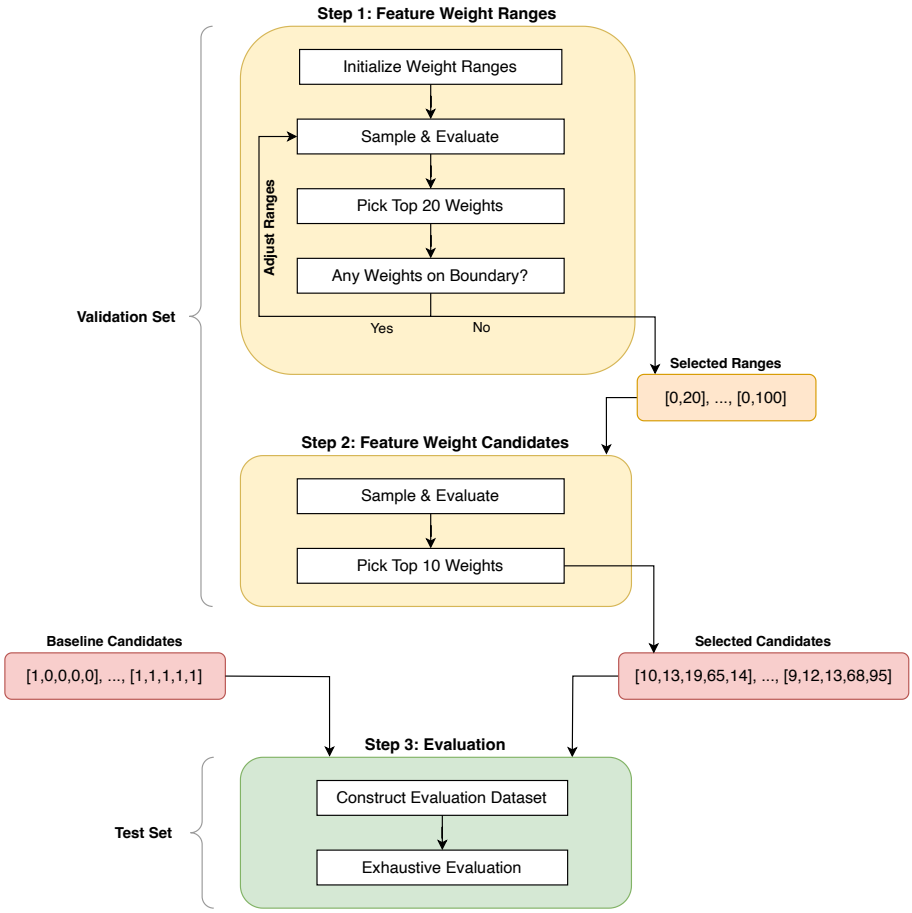


Figure 4.1: Evaluation Strategy. The evaluation strategy consists of three steps. The first two steps - finding suitable feature weight ranges and specific candidate weight vectors from those ranges - are performed on the validation set. The top ten candidates and six baseline weight vectors are used in the final step to construct the evaluation dataset and measure the performance of the Hybrid Recommender on the test set.

Step 2: Randomized Feature Weights Search

The second step, conducted on the validation set, assesses a larger number of feature weights within the ranges selected in step 1 and identifies the top $k = 10$ best-performing weight vectors.

This step is performed as follows:

1. Sample $n = 200$ feature weight combinations from the identified ranges of step 1, resulting in $1,600,000$ input combinations, accounting for the 8 language models and 1000 query documents of the validation set.

2. Randomly select $N = 10,000$ combinations, ensuring an average of 50 samples for each

weight vector.

3. Similar to step 1, compute the marginal MAP for the weight vectors, averaged over all query documents and language models, and select the top $k = 10$ candidates for the final evaluation in step 3.

Since step 2 does not operate sequentially like the first step, the process can be parallelized for more efficient computation.

Step 3: Performance Evaluation

The third step evaluates the inference performance of various input combinations on the test set.

The evaluation procedure is as follows:

1. Augment the selected feature weights of step 2 with the 6 baseline weight vectors $[1, 0, 0, 0, 0], [0, 1, 0, 0, 0], \ldots, [0, 0, 0, 0, 1], [1, 1, 1, 1, 1]$ resulting in $k = 16$ candidates. These simple weight combinations are added for interpretability: Each unit weight vector isolates the effect of a single feature while the vector of ones contributes the same impact to each of the five features. Coupled with the 8 language models and 1000 query documents of the test set, the evaluation dataset comprises $128,000$ combinations.

2. Unlike previous steps, the evaluation is exhaustive without any prior subsampling. Inference runs for each of the $128,000$ input combinations are initiated. The resulting candidate and final recommendation rankings are evaluated using the MAP and the measure for recommendation diversity as defined in Section 3.2.2.

Similar to the second step, step 3 can be parallelized for efficient computation. Figure 4.2 visualizes the third and final component of the evaluation process.

Discussion

The proposed three-step evaluation strategy has several advantages. It is systematic and reproducible, data-driven, and computationally feasible. However, it also has some drawbacks. In reference to step 1, the range of a given feature expands as soon as its corresponding weight reaches the interval boundary for *any* of the 20 top-performing weight vectors. This conservative approach may lead to overly broad feature weight ranges. Moreover, range adjustments may not lead to improved candidates in the first place, as a weight vector may perform well *despite* one of its values being at the interval boundaries, not *because* of it.

One way to address this shortcoming is to require a minimum number of weights at the boundary before expanding the range. Alternatively, the process of adjusting the ranges could be bidirectional by iteratively making the ranges smaller and larger until the optimal ranges are found. However, to maintain the evaluation strategy's simplicity and clarity, we opted against these modifications.

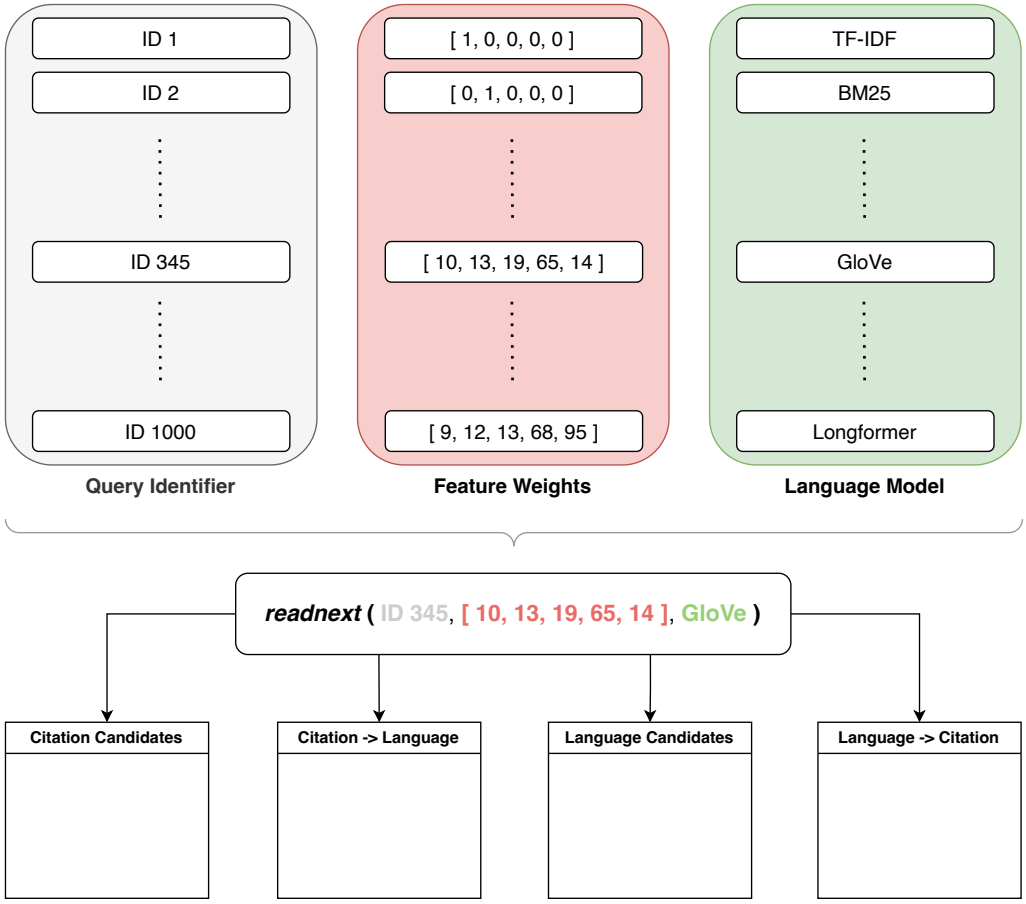


Figure 4.2: Evaluation of the Hybrid Recommender. The evaluation dataset is constructed from each combination of the $1,000$ test set query papers, 16 feature weights, and 8 language models, resulting in $128,000$ rows. One inference run is conducted for each row, returning recommendation lists of the Citation Recommender candidates, the Language Recommender candidates, and final rankings of the C $\rightarrow$ L and the L $\rightarrow$ C Hybrid Recommenders. All recommendation rankings are evaluated using the MAP and the measure for recommendation diversity.

4.2 Evaluation Results

This section presents and discusses the results of our evaluation. Section 4.2.1 establishes a baseline for the subsequent results comparison. The discussion then pivots to **RT 3**, delving into the identification of optimal feature weights for the Citation Recommender and pinpointing the most effective language model for the Language Recommender. These are elaborated upon in Section 4.2.2 and Section 4.2.3, respectively. Finally, Section 4.2.4 addresses **RT 4** by comparing the two hybridization orders in terms of MAP and recommendation diversity and analyzing the interactions between the Citation Recommender and the Language Recommender.

4.2.1 Establishing a Baseline Performance

To contextualize MAP scores of various feature weights and language models, a baseline performance is needed. This baseline is established by constructing a *Null Model* that randomly selects papers from the training set as recommendations.

In contrast to the Precision metric, where the baseline performance can be easily determined analytically, the MAP requires a simulation-based approach. This approach involves sampling recommendation lists filled with 0s (irrelevant) and 1s (relevant) at random and computing the AP for each list. The baseline MAP is derived from averaging these AP values across all simulation runs. Two parameters influence the simulation results: the number of recommendations for each run and the probability of drawing a relevant item. Table 4.1 presents simulation results for various combinations of these parameters.

0/1 Ratio	# Recommendations	MAP
20/80	10	0.843
20/80	20	0.828
20/80	50	0.815
50/50	10	0.608
50/50	20	0.569
50/50	50	0.536
71.3/28.7	10	0.433
71.3/28.7	**20**	**0.384**
71.3/28.7	50	0.338
80/20	10	0.350
80/20	20	0.308
80/20	50	0.257

Table 4.1: MAP for simulated recommendation lists. Each list contains binary values: 0 for irrelevant and 1 for relevant recommendations. The samples are randomly drawn from a Bernoulli distribution. The first column displays the proportions of irrelevant and relevant items, or, equivalently, the probability p of the Bernoulli distribution to draw a relevant item. The second column indicates the length of the recommendation list, and the third column the resulting MAP. The bold values highlight the parameters for the Null Model baseline in this thesis. Generally, the MAP increases with a higher proportion of relevant items but decreases with a higher number of recommendations.

The simulation parameters for this thesis's evaluation are based on the properties of the test set. Thus, the probability of sampling a relevant candidate is set to 0.287, which corresponds to the average fraction of relevant query papers in the test set. The number of recommendations is set to 20, which is the default number of recommendations for the hybrid system. According to these parameters, we simulate $100,000$ recommendation rankings and calculate the AP for each, yielding a baseline MAP of 0.384.

4.2.2 Evaluating the Citation Recommender

The evaluation strategy presented in Section 4.1 identified feature weight ranges of $[0, 20]$ for the publication date, document citation count, and author citation count, and $[0, 100]$ for the co-citation analysis and bibliographic coupling features. As the ranges for the citation-based features were adjusted to include higher values than the ranges for the global document characteristics, the citation-based features exert a more substantial influence on recommendation performance.

Figure 4.3 shows the MAP values for the Citation Recommender candidates averaged over all query papers in the test set. The y-axis displays the selected weight candidates from step 2 of the evaluation process. The integer values correspond to the order [publication date, document citation count, author citation count, co-citation analysis, and bibliographic coupling].

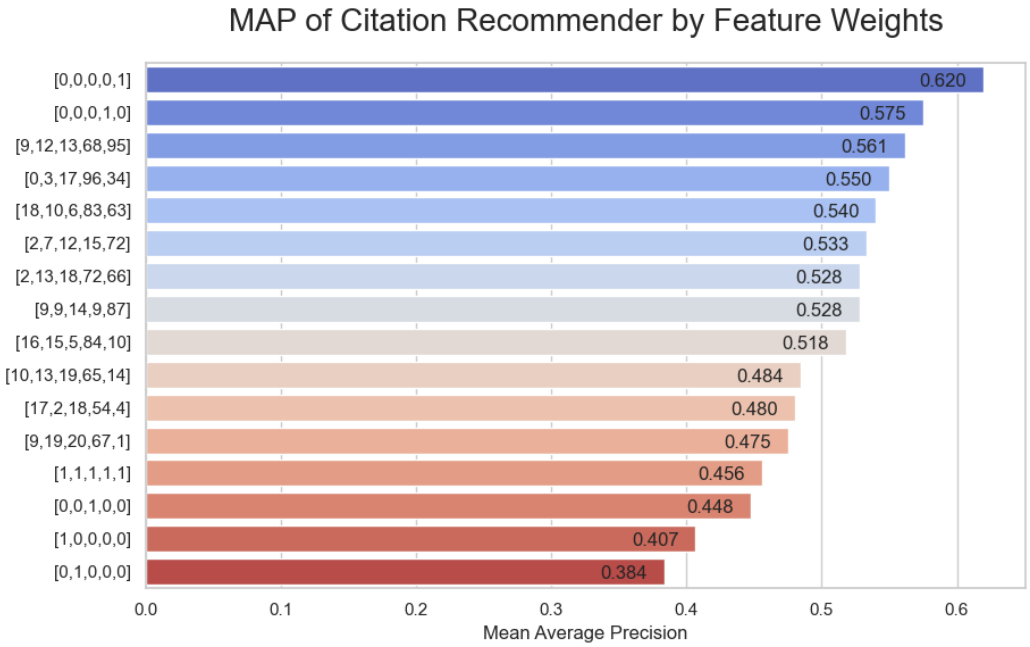


Figure 4.3: Performance evaluation of the Citation Recommender using different feature weights. The y-axis shows the feature weights for the publication date, document citation count, author citation count, co-citation analysis, and bibliographic coupling in that order. The x-axis shows the MAP values for the Citation Recommender candidates averaged over all query papers in the test set. Larger weights for the co-citation analysis and bibliographic coupling scores generally lead to higher MAP values. Bibliographic coupling performs best with a MAP of 0.620, whereas the worst performing feature is the paper citation count with a MAP of 0.384.

The bibliographic coupling weight varies heavily among the top 10 candidates, taking on high values like 95 or 87 in some instances and lower values like 1 or 4 in others. A similar pattern can

be observed for the co-citation analysis weight, though its variation is slightly less pronounced. The weights for the publication date, document citation count, and author citation count spread fairly evenly within their $[0, 20]$ range.

Figure 4.3 indicates significant performance differences across different weight vectors. For instance, the lowest MAP of 0.384 is achieved by the vector $[0, 1, 0, 0, 0]$ isolating the paper citation count, while the highest MAP of 0.620 is observed for $[0, 0, 0, 0, 1]$, which puts the full weight on the bibliographic coupling score. In contrast, the vectors with full weight on the global document features indicate the worst performance. Using only the paper citation count to generate recommendations yields a MAP of 0.384, which is exactly equal to the baseline MAP of the Null Model. This suggests that, averaged over all query papers in the test set, selecting the most cited papers is not more effective than the random selection of arbitrary papers from the training set.

Weight vectors identified by the randomized search predominantly lead to good but not great results. They range from a MAP of 0.475 for the weights $[9, 19, 20, 67, 1]$ to a MAP of 0.561 for the vector $[9, 12, 13, 68, 95]$. In line with prior observations, we can identify the general trend that higher weights for the bibliographic coupling and co-citation analysis features lead to higher MAP values. A potential reason that weight vectors found by the randomized search do not generalize well to the test set is that the candidate selection process optimized for the Hybrid Recommender performance (averaging the MAP scores of the C $\rightarrow$ L and L $\rightarrow$ C orders) rather than for the Citation Recommender performance. In Figure 4.3, only the latter is portrayed.

4.2.3 Evaluating the Language Recommender

To identify the optimal language model for the Language Recommender, Table 4.2 shows the MAP values for the Language Recommender candidates averaged over all query papers in the test set.

Language Model	MAP
TF-IDF	0.628
BM25	0.596
Word2Vec	0.588
FastText	0.575
GloVe	0.581
BERT	0.603
SciBERT	**0.660**
Longformer	0.524

Table 4.2: Performance evaluation of the Language Recommender candidates using different language models. The MAP per language model is calculated by averaging over all query papers in the test set. Models with similar properties tend to produce similar results. The SciBERT model, highlighted in bold, achieves the highest MAP of 0.660. The Longformer model performs worst with a MAP of 0.524.

The SciBERT model achieves the highest MAP score of 0.660, with TF-IDF and BERT following

at 0.628 and 0.603, respectively. The Longformer model performs worst with a MAP of 0.524. Comparing the findings with Figure 4.3, it is evident that the Language Recommender candidates outperform the Citation Recommender candidates. This suggests that processing paper abstracts with language models is generally more effective for paper recommendation than relying on global document characteristics and citation-based features.

However, the performance differences between Citation Recommender and Language Recommender can diminish depending on which feature weights and language model are selected. For example, the Citation Recommender with the weight vector $[0, 0, 0, 0, 1]$ outperforms the Language Recommender for 6 of the 8 language models with a MAP of 0.620. Conversely, the Language Recommender employing the Longformer model only ranks ninth out of 16 within the feature weights ranking of the Citation Recommender. Nonetheless, the Language Recommender produces more robust and reliable results. Even its lowest MAP score of 0.524 significantly surpasses the lowest Citation Recommender score of 0.384. In particular, all language models outperform the Null Model baseline by a large margin.

There is a weak correlation between the language model category and recommendation performance. Models that have similar structural characteristics produce comparable results. For instance, the contextual embedding models SciBERT and BERT as well as the sparse embedding models TF-IDF and BM25 all exhibit above-average performance. Conversely, the static embedding models Word2Vec, FastText, and GloVe all perform below average. The Longformer model is the only outlier, performing worse than all other models despite being a contextual embedding model like BERT and SciBERT.

Longformer's Performance

While the strong performance of SciBERT can be attributed to its domain-specific pretraining on scientific text, the subpar performance of the Longformer model appears surprising at first glance. The primary aspect that distinguishes Longformer is its design to efficiently process long documents beyond the 512 token limit of BERT by modifying the self-attention mechanism of the Transformer architecture. Moreover, it is pretrained specifically on a corpus comprising long documents.

However, only 0.58% of all abstracts in the test set exceed the 512 token threshold, implying that the Longformer's advantage over BERT and SciBERT is relevant for only a small fraction of documents. Due to the discrepancy between the distribution of the pretraining and inference corpora, Longformer's global attention tokens may be optimized for long documents, thereby rendering them less effective for succinct abstracts. As a result of these disparities, Longformer might allocate more capacity for handling long-range dependencies and might not extract finer local patterns that are crucial for short texts like paper abstracts.

While we are unable to pinpoint a definitive reason for Longformer's poor performance in our

evaluations, the underlying factor might even stem from the multitude of parameters that differentiate the Longformer and BERT models, some of which are listed in Table 3.1. These encompass different tokenizers, optimization techniques, learning rates, weight initialization protocols, or even a discrepancy in the number of training epochs. To identify the root cause, a methodical approach could involve varying individual parameters sequentially, observing any resulting changes in the recommendation performance. Given the constraints of computational time and resources, several strategies can be contemplated.

If a full retraining of the Longformer model is feasible, an iterative adaptation of Longformer's and BERT's parameters can be undertaken, including the pretraining corpus, tokenizer, and optimizer, while observing their effect on recommendation performance. Alternatively, if a full-scale retraining proves impracticable but fine-tuning remains achievable, we might consider fine-tuning the Longformer model using a subset of the paper abstracts. If the performance improves significantly, it indicates that the original pretrained weights are not ideal and differences in the pretraining corpus's characteristics might be the root cause. In the event that neither exhaustive retraining nor fine-tuning is a viable option, a comparative assessment of the Longformer and BERT models on either extensively long documents, like full paper texts, or very short documents, like paper titles, could be conducted. A diminished performance disparity for extended documents coupled with an amplified performance gap for short documents would suggest that either the nature of the pretraining corpus or the distinct attention mechanics is the underlying cause of Longformer's poor performance.

4.2.4 Evaluating the Hybrid Recommender

Previous sections have focused on the *marginal* performance of the Citation Recommender (Section 4.2.2) and the Language Recommender (Section 4.2.3). This section shifts focus to the *interactions* between the two recommenders in the hybrid setup. Given that the cascade hybridization strategy follows a sequential approach without communication between the two recommenders, it's intuitive that the Citation Recommender and the Language Recommender operate independently. Subsequent sections investigate whether this assumption holds true.

Figure 4.4 shows the MAP values of the C $\rightarrow$ L Hybrid Recommender for all combinations of feature weights and language models. The first column of the heatmap represents the MAP values for the Citation Recommender candidates before re-ranking.

To conclude independence between the two recommenders, two conditions must be met: First, the re-ranking effect of each language model should not depend on the feature weights. In other words, the column-wise differences to the first column should be constant across all rows. As this condition compares values within the same column of Figure 4.4, we term this condition *vertical independence*.

Second, the relative performance ranking between language models should be the same when

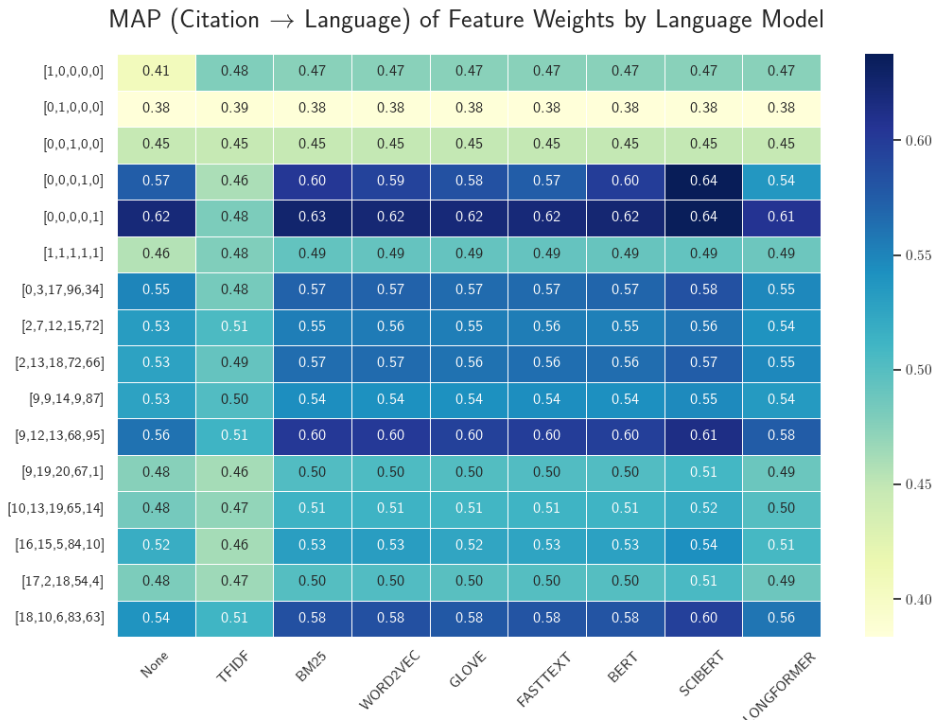


Figure 4.4: Performance evaluation of the C → L Hybrid Recommender using various feature weights and language models. Darker colors correspond to higher MAP values. The first column presents the MAP values for candidates from the Citation Recommender prior to re-ranking. The following columns display MAP values after the re-ranking by the Language Recommender using the respective language model for each feature weight vector. SciBERT consistently delivers the highest performance improvements, while TF-IDF is the sole model that generally results in a decrease in MAP during re-ranking.

the Language Recommender is used for candidate selection (Table 4.2) and when it is used for re-ranking (Figure 4.4). Thus, we expect the highest MAP scores for SciBERT and the lowest for Longformer within each row of the heatmap. As this condition compares values within the same row of Figure 4.4, we term this condition *horizontal independence*.

Figure 4.4 indicates that vertical independence is not met. The re-ranking effect of each language model varies heavily between the feature weights. For example, none of the language models can improve the candidate recommendations generated with weights $[0, 0, 1, 0, 0]$. In contrast, the recommendations for the $[18, 10, 6, 83, 63]$ weight vector in the final row can be significantly improved by re-ranking with all language models except TF-IDF.

However, horizontal independence is almost satisfied. With the exception of the TF-IDF model, the most performant candidate selectors (SciBERT, BERT and BM25) are generally also the most

performant re-ranking models, indicated by the highest MAP values within each row. Conversely, the least performant candidate selector, Longformer, remains the least performant re-ranking model for most feature weight vectors.

A further question of interest is whether high initial MAP scores from the Citation Recommender are associated with either a strong or a weak re-ranking effect. However, Figure 4.4 reveals no clear relationship between the initial MAP scores in the first column and the effect size of re-ranking. For example, the worst weight vector of $[0, 1, 0, 0, 0]$ cannot be improved by re-ranking, while the second to worst weight vector of $[1, 0, 0, 0, 0]$ can be significantly improved by re-ranking. At the higher end of performance, the optimal weight vector $[0, 0, 0, 0, 1]$ shows only minimal improvement when re-ranked. In contrast, the second-best weight vector, $[0, 0, 0, 1, 0]$, is profoundly influenced by the Language Recommender. Notably, the SciBERT model boosts its MAP from 0.57 to 0.64.

The heatmap corresponding to the reverse L $\rightarrow$ C ordering is displayed in Figure 1. We can observe that feature weight vectors performing poorly for candidate selection (Figure 4.3) also underperform during the re-ranking step. However, this trend does not apply to the top-performing feature weights $[0, 0, 0, 1, 0]$ and $[0, 0, 0, 0, 1]$. These vectors consistently yield worse MAP scores in the re-ranking process than weight vectors with a mix of non-zero weights.

Inspecting the TF-IDF Behavior

A striking observation in Figure 4.4 is the underwhelming performance of the TF-IDF model during re-ranking. Out of the 16 weight vectors, TF-IDF worsens the recommendations in 12 cases. In some instances, this decline is dramatic, as evidenced by the drop from 0.62 to 0.48 for the weight vector $[0, 0, 0, 0, 1]$. This finding is unexpected, given that TF-IDF is the second-best candidate selector, as shown by Table 4.2. Moreover, given that the second sparse vector embedding model, BM25, does not have this issue, there is no clear methodological explanation for the poor performance of TF-IDF in the hybrid setup. This section further investigates the impact of the TF-IDF model within the hybrid system.

Figure 4.5 demonstrates the effect of TF-IDF on the Hybrid Recommender's MAP when used for re-ranking (left) or candidate selection (right). For the C $\rightarrow$ L order, there appears to be a negative correlation between the weights of bibliographic coupling and co-citation analysis and the resulting change in MAP from re-ranking: The greater the weights assigned to these two features, the more the MAP decreases after re-ranking with TF-IDF. The most pronounced performance declines correspond to the weight vectors $[0, 0, 0, 0, 1]$ and $[0, 0, 0, 1, 0]$, where one of the citation-based features is given the full weight of 1. In contrast, weight vectors such as $[1, 0, 0, 0, 0]$, $[0, 1, 0, 0, 0]$, and $[0, 0, 1, 0, 0]$, which don't factor in bibliographic coupling and co-citation analysis, and the vector $[1, 1, 1, 1, 1]$, with a minimal relative impact from these two features, see improvements from re-ranking.

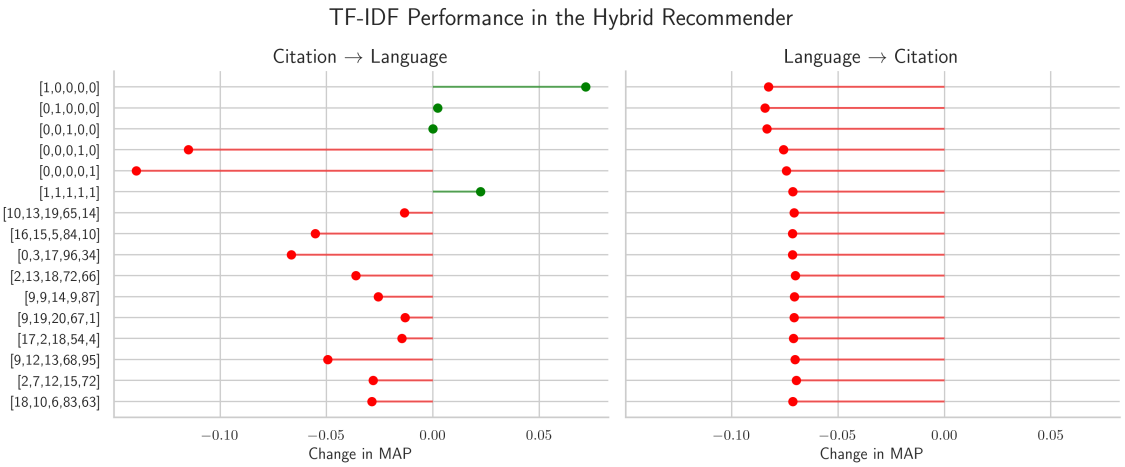


Figure 4.5: Performance changes in MAP due to the TF-IDF model. The left plot depicts the MAP change when TF-IDF is employed for re-ranking, whereas the right plot illustrates the MAP change when TF-IDF is utilized for initial candidate selection. The y-axis displays feature weights for the publication date, document citation count, author citation count, co-citation analysis, and bibliographic coupling, respectively. Green lines signify an MAP increase, and red lines denote a decrease. Except for baseline weight vectors with diminished weights for co-citation analysis and bibliographic coupling, TF-IDF generally reduces the MAP during re-ranking. Conversely, when TF-IDF is applied for candidate selection, the Citation Recommender consistently lowers the MAP during the re-ranking phase.

Besides the connection to the citation-based features, the TF-IDF model shows a relationship with the initial recommendation performance of the Citation Recommender before re-ranking. Revisiting Figure 4.4, we see that weight vectors with larger values for co-citation analysis and bibliographic coupling are strongly associated with high initial MAP scores. On the other hand, weight vectors where TF-IDF positively affects the MAP rank among the lowest candidates. This observation indicates a consistent pattern: TF-IDF tends to enhance recommendations that initially perform poorly and diminish those that start off strong. This results in final rankings approaching each other in terms of MAP.

Inspecting the L → C order in the right plot of Figure 4.5, there's no discernible link between the TF-IDF model's role in candidate selection and the re-ranking effect of specific feature weights. Instead, the Citation Recommender consistently reduces the recommendation performance across all feature weights, with a MAP decrease ranging between 0.07 and 0.09.

In conclusion, the impact of the TF-IDF model within the Hybrid Recommender remains without a clear explanation based on the data. The potential causes — whether they stem from an adverse relationship with citation-based features or from the tendency of TF-IDF to move all MAP values closer together — remain indistinguishable and require further investigation. In general, although

TF-IDF shows great potential in the Language Recommender, it underperforms in the hybrid setup, regardless of the hybridization order.

Comparing Hybridization Strategies

Figure 4.6 displays AP distributions stratified by hybridization strategy. For the candidate rankings, the Language Recommender outperforms the Citation Recommender with a MAP of 0.594 compared to 0.505. This aligns with the findings of Figure 4.3 and Table 4.2, when averaged across all feature weights and language models.

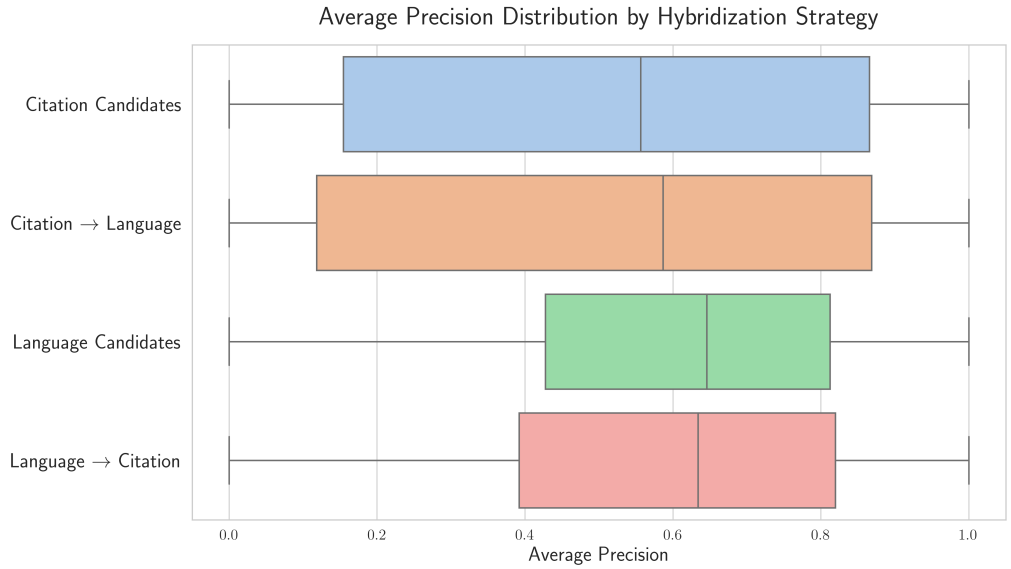


Figure 4.6: Comparison of AP distributions for the candidate and final recommendations generated by the C $\rightarrow$ L and L $\rightarrow$ C Hybrid Recommenders. The boxplots compile all AP scores from the $128,000$ inference runs conducted on the evaluation dataset. The vertical lines in each box represent the medians. The corresponding MAP values, in order from top to bottom, are 0.505, 0.522, 0.594, and 0.584. Each of the four distributions has minimum and maximum AP values of 0 and 1, respectively. The interquartile ranges, represented by the box widths, are wider when the Citation Recommender is applied first, suggesting a higher variability in performance.

For the hybrid models, the L $\rightarrow$ C Hybrid Recommender achieves a MAP of 0.584, outperforming the C $\rightarrow$ L Hybrid Recommender's score of 0.522. Thus, applying the Language Recommender first for candidate selection and the Citation Recommender second for re-ranking yields a higher MAP than the reverse order. Additionally, the variability in MAP for the C $\rightarrow$ L order is much greater than for the L $\rightarrow$ C order, pointing to a higher risk and variability in performance with the C $\rightarrow$ L approach. In particular, 35.86% of the AP scores for the C $\rightarrow$ L Hybrid Recommender are worse than the Null model's MAP of 0.384. In contrast, for the L $\rightarrow$ C Hybrid Recommender, only 24.54% fall short of the Null model's performance.

Comparing candidate and final rankings, only the C → L Hybrid Recommender benefits from re-ranking with a MAP increase of 0.017, while the L → C Hybrid Recommender experiences a MAP decline of 0.010. This suggests that, on average, the quality of recommendations generated by the Language Recommender deteriorates upon re-ranking by the Citation Recommender. Therefore, language models provide valuable information for paper recommendations beyond what global document characteristics and citation-based features offer. The reverse does not hold true. In this way, the performance gap between the final rankings narrows compared to that of the candidate rankings.

Hybridization Strategies by Language Model

Figure 4.7 takes a more granular perspective than Figure 4.6 by comparing the MAP scores stratified by hybridization strategy and language model. This figure provides additional information beyond Figure 4.4 by including the reverse L → C order as well. Moreover, the focus of Figure 4.7 is more on the comparison between hybridization strategies rather than between language models.

Across all language models, the L → C Hybrid Recommender consistently performs better than the C → L Hybrid Recommender. Similar to Figure 4.4, the improvements are most pronounced for the SciBERT model, where the L → C Hybrid Recommender achieves a MAP of 0.626 compared to 0.539 for the C → L Hybrid Recommender. The smallest difference is again observed for the Longformer model, where the L → C Hybrid Recommender achieves a MAP of 0.542 compared to 0.514 for the C → L counterpart.

In contrast to the rather uniform behavior for the C → L Hybrid Recommender where only the TF-IDF model stands out by decreasing the recommendation performance through re-ranking, the picture for the L → C Hybrid Recommender is more diverse. Top-performing language models like SciBERT, TF-IDF, BERT, and BM25 do not benefit from re-ranking by the Citation Recommender where the MAP decline is largest for TF-IDF (0.628 to 0.555) followed by SciBERT (0.660 to 0.626). Conversely, the lower-performing Longformer, GloVe, FastText, and Word2Vec models experience an uptick in MAP post re-ranking with Longformer seeing the largest improvement (0.524 to 0.542). Thus, while the final recommendations become more similar in MAP than the candidate recommendations, it is also notable that the performance of the L → C Recommender levels out across different language models.

Recommendation Diversity

As detailed in Section 3.2.2, recommendation diversity serves as a secondary metric - beyond MAP - to quantify the degree of interdisciplinarity in recommendations. As recommendation diversity is not affected by the re-ranking process, it is computed for the candidate rankings only.

Table 4.3 compares the diversity of candidate recommendations by the Citation and Language Recommenders which correspond to the diversity of the C → L and L → C Hybrid Recommenders,

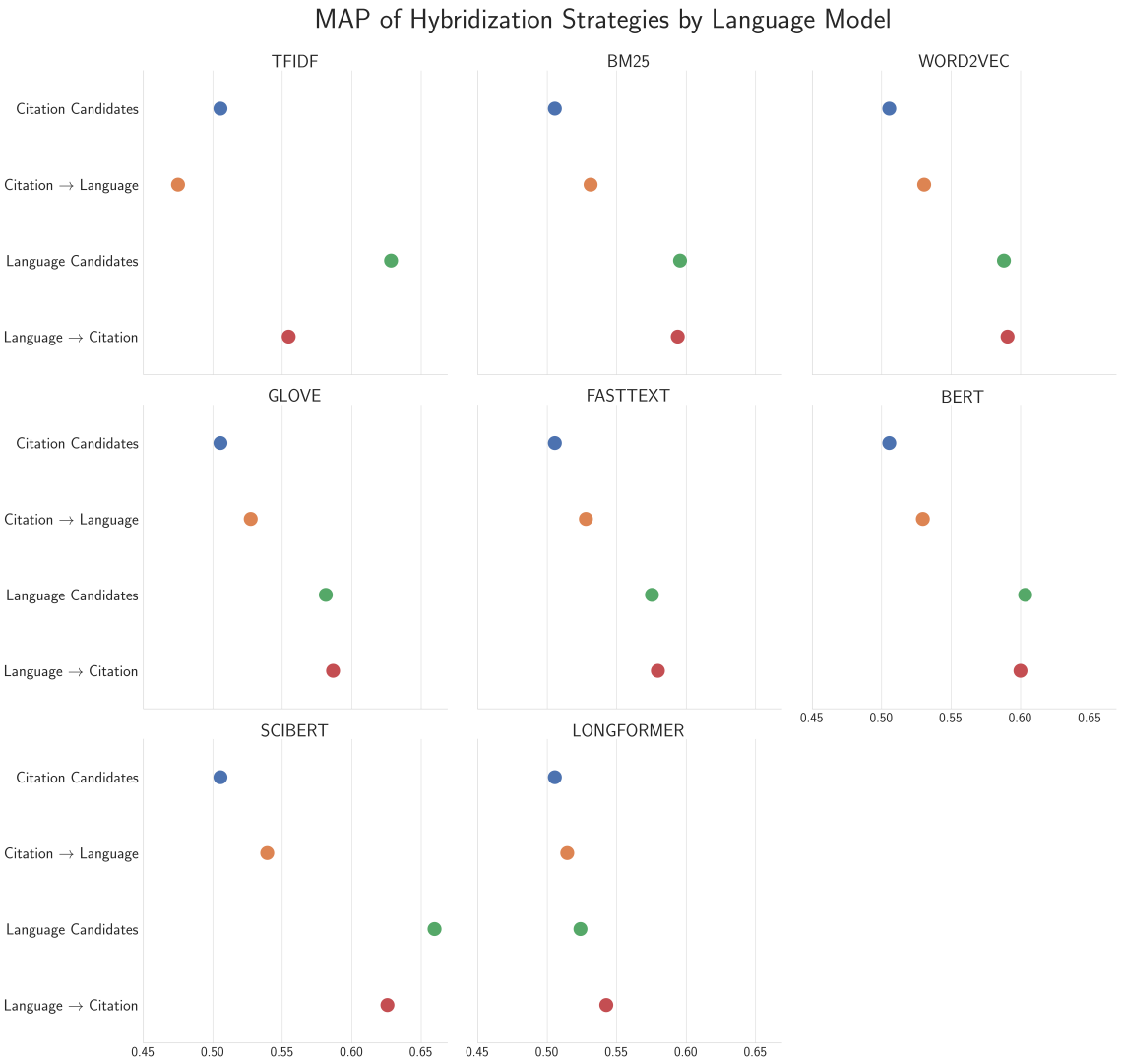


Figure 4.7: Evaluation of hybridization strategies segmented by language model. The colored dots depict the MAP values for both C → L and L → C in their candidate and final recommendations. Except for TF-IDF, all language models enhance the initial recommendations from the Citation Recommender through re-ranking. In the L → C order, the impact of re-ranking differs depending on the language model. In every instance, the L → C Hybrid Recommender surpasses the performance of the C → L Hybrid Recommender. The peak MAP value of 0.660 is attained with the Language Recommender candidates using the SciBERT language model.

respectively. The Language Recommender typically produces more varied recommendations, with a mean of 13.4 unique labels compared to the Citation Recommender's 9.1. Both distributions are slightly right-skewed, as evidenced by medians of 13.0 and 8.0, respectively.

Statistic	Citation → Language	Language → Citation
Mean	9.1	13.4
Standard Deviation	2.6	3.7
Minimum	3.0	4.0
1st Quartile	8.0	11.0
Median	8.0	13.0
3rd Quartile	9.0	16.0
Maximum	24.0	28.0

Table 4.3: Summary statistics for the recommendation diversity of the C → L and L → C Hybrid Recommenders. The Language Recommender typically generates more interdisciplinary recommendations across a broader set of categories compared to the Citation Recommender. As the distribution for the Citation Recommender is more closely centered around its median, the number of research fields in its recommendations is more consistent than that of the Language Recommender.

The spread of unique labels in the Language Recommender's distribution is wider, as indicated by its larger standard deviation (3.7 compared to 2.6) and its broader interquartile range (5.0 versus 1.0). Both distributions signify outliers in both directions, with the Citation Recommender's range spanning from 3.0 to 24.0 and the Language Recommender's range from 4.0 to 28.0.

In conclusion, the Language Recommender excels by providing not only more relevant recommendations on average (Figure 4.6), but also by suggesting papers from a broader array of categories (Table 4.3) than the Citation Recommender. Whereas the variability of the Language Recommender's MAP scores is lower than the variability of the Citation Recommender leading to more robust recommendations, the inverse is true for the recommendation diversity with the Citation Recommender producing more consistent results.

Chapter 5

Conclusion

To conclude this thesis, Section 5.1 summarizes the methodology and the main findings of our experiments. Section 5.2 highlights how our work impacts the field of paper recommender systems. Finally, Section 5.3 points out limitations of our work and outlines directions for future research.

5.1 Summary

This thesis introduces a hybrid recommender system for Computer Science papers designed to support researchers in their literature search. Given a query paper, the system generates specific and relevant recommendations. To address current literature search shortcomings, the system emphasizes customizability and transparency, drawing information from various paper aspects, including metadata, citation analysis, and semantic analysis, to generate recommendations.

The Citation Recommender, the first component of the hybrid system, aggregates factors factors such as publication date, paper citation count, author citation count, co-citation analysis score, and bibliographic coupling score into a single weighted score. This score is based on user-specified weights that quantify the impact of each feature. Candidate papers with the highest weighted score are recommended to the user.

The Language Recommender, the second component of the hybrid system, generates recommendations based on the semantic similarity between the query paper and each candidate paper. The semantic similarity is measured by the cosine similarity between document embeddings of the paper abstracts. The user can choose from one of eight language models to generate the document embeddings. Candidate papers with the highest cosine similarity to the query paper are recommended to the user.

The Hybrid Recommender applies the Citation Recommender and the Language Recommender sequentially. One recommender pre-selects a candidate list of recommendations, while the other recommender re-ranks this list to generate the final recommendation ranking. The classification

of recommendations as relevant or irrelevant is based on the arXiv categories of the query and candidate papers. If query and recommended candidate share one or more categories, the recommendation is considered relevant. The MAP is used as the primary metric to evaluate the recommender system's performance.

Our evaluation reveals that for the Citation Recommender, citation-based features significantly outweigh global document characteristics. In particular, the bibliographic coupling score is the most beneficial feature in terms of MAP, followed by the co-citation analysis score. A general trend we identified is that higher weights assigned to at least one citation-based feature, compared to weights for global document characteristics, correlate with higher MAP scores. Conversely, for global document characteristics, the paper citation count performs no better than recommendations from randomly selected papers.

The SciBERT language model yields the best performance for the Language Recommender, whereas the Longformer model performs worst. When applying only a single recommender, the Language Recommender outperforms the Citation Recommender in terms of MAP and recommendation diversity, which quantifies the degree of interdisciplinarity of the recommendations. Additionally, the Language Recommender demonstrates greater robustness in language model selection than the Citation Recommender does in feature weight selection. This is evident from the significantly smaller performance variations across language models compared to those across feature weights.

Using the Language Recommender for candidate selection and the Citation Recommender for re-ranking yields better and more consistent results than the reverse order. However, peak performance is realized not through a hybrid model but solely with the Language Recommender employing the SciBERT language model.

5.2 Broader Impact

This thesis contributes to the field of paper recommender systems in several ways.

First, the evaluation of the Citation Recommender shows that the bibliographic coupling and co-citation analysis scores are markedly more effective than the global document characteristics for generating relevant recommendations. A key distinction between these two feature types is that citation-based features are query-specific, whereas global document characteristics are query-agnostic. Since the query paper provides insight into the user's profile, this finding motivates a stronger focus on CBF for paper recommendation. For example, academic search engines such as Google Scholar or Semantic Scholar might enhance relevance by allowing user-specific papers as input to their searches.

Second, our experiments reveal that the Language Recommender outperforms the Citation Recommender. Thus, we propose to put more emphasis on text-based approaches for paper recommendation in future research. Instead of combining content-based and citation-based approaches,

hybrid systems could combine several content-based approaches as shown by Akkalyoncu Yilmaz et al. [96] and Farber and Sampath [83].

The strong performance of SciBERT, which stands out as the sole language model trained on scientific text, encourages exploration of the benefits of domain-specific approaches in the field of paper recommendation and beyond, emphasizing the importance of fine-tuning. On the flip side, the underwhelming performance of the Longformer model underscores the importance of using models in alignment with their designated purposes. In the case of Longformer, this is processing long documents beyond what traditional models like BERT are capable of, which is only partially fulfilled by the paper abstracts used in this thesis.

Lastly, while the MAP is a valuable metric, it is just one of several metrics used to evaluate the performance of recommender systems. Metrics like diversity, serendipity, novelty, and coverage are more difficult to quantify but crucial to the user experience [21, 40, 45]. In this thesis, we attempted to measure recommendation diversity by gauging the interdisciplinarity of the recommendations. We hope that this approach inspires further research into quantifying metrics beyond accuracy, providing a more holistic measurement of recommender system performance.

5.3 Outlook

Throughout this thesis, we encountered various limitations arising from time constraints and data availability. Addressing these challenges can guide future research in the field of paper recommender systems.

Data Sources

This thesis primarily utilizes the D3 dataset [97]. A limitation of this dataset is its lack of continuous updates with recent papers. Ensuring a regularly updated dataset would facilitate the integration of the latest research into the training corpus, potentially improving recommendation quality during inference.

Furthermore, incorporating an additional data source that provides the complete text of papers, rather than just the abstracts, can offer notable enhancements. The Citation Recommender could use citation-based features that leverage the citation's position within the text, rather than only relying on static citation analysis metrics like co-citation analysis and bibliographic coupling. Location-aware options are CPA [10] (Section 2.2.2) and section-based bibliographic coupling [9] (Section 2.2.3). Additionally, by harnessing the complete text of each paper, the Language Recommender can produce more informative document embeddings. This is especially advantageous when the abstract fails to convey the full depth of the document. Such an approach can enhance keyword-based models like TF-IDF and BM25, especially when crucial keywords from the paper are omitted in the abstract. Moreover, the Longformer's performance may improve as utilizing full

texts would better align the inference corpus with Longformer's training data. However, a key drawback of using the full paper text is the significant increase in computational resources and time required for data processing.

A further limitation of this thesis is its dependence on arXiv categories as the ground-truth for recommendation evaluation. Although arXiv categories provide a convenient classification method, their flat and broad structure could reduce the efficacy of recommendations. This becomes especially evident when considering how the choice of labels and the definition of relevance, based on shared categories, are closely linked.

Adopting a more detailed labeling system, like the ACM Computing Classification System[1] with its multi-level hierarchy, could offer greater flexibility in defining a ground-truth for relevance. For example, relevance could be more stringently defined based on membership within the same specific subcategory. Under the ACM system, papers within the *Language models* subcategory would be required to align with the entire hierarchy, namely *Information Systems* → *Information retrieval* → *Retrieval models and ranking* → *Language models*. Yet, this refined approach might come at a cost: By focusing on highly specific categories for recommendations, the diversity, interdisciplinarity, and serendipity of the recommendations might suffer.

Methodology

In addition to data sources, several extensions to the methodology could be explored in future work.

First, the Hybrid Recommender could be redesigned to accommodate multiple query papers when generating recommendations, rather than only one. Such an approach would better mirror a researcher's varied interests, resulting in more personalized recommendations.

Second, integrating a researcher's reading history could help avoid recommending the same papers repeatedly. Users could opt to exclude specific papers from future recommendations, ensuring they are not suggested previously read papers or those deemed irrelevant to their research.

Third, exploring hybridization techniques that operate concurrently, rather than in a sequential manner like the cascade strategy, could be beneficial. Although this increases computational demands, it gives users more control by allowing them to adjust the importance of the Citation Recommender in relation to the Language Recommender through a weighting scheme.

Evaluation Insights

Several enhancements to the Citation Recommender and the Language Recommender could be explored based on the evaluation outcomes.

[1] `https://dl.acm.org/ccs`

As our experiments have shown that the citation-based features are significantly more important than the global document characteristics for the Citation Recommender, replacing some or all of the latter with more features that utilize citation information might be advantageous.

For the Language Recommender, the SciBERT model, being the only domain-specific model, outperformed all other language models. This underscores the benefits of fine-tuning more language models with scientific text before using them for paper recommendation. Such a step promises two advantages: First, it is likely to increase the recommendation performance of these models in terms of MAP. Second, it ensures a more robust comparison of different language models by neutralizing the domain-specificity factor. Thus, conclusions about the relative performance among model types such as keyword-based sparse embedding models, static embedding models, and contextual embedding models can be drawn with greater confidence.

Lastly, a deeper dive into the anomalous behavior of the TF-IDF model in our experiments is warranted. Specifically, understanding the inverse relationship between TF-IDF and the impact of citation-based attributes on re-ranking performance could be tested across different setups to either confirm this behavior or attribute it to random chance.

Accessibility

The last avenue for future enhancement pertains to the accessibility and user-friendliness of the *readnext* framework. The recommender system is currently only available as a Python package installable from GitHub[2]. To make the framework accessible to a broader and less technical audience, it could be deployed as a web application. Such a transition would allow researchers to bypass the need for setting up a development environment, locally installing the package, and navigating the setup instructions before using the system.

[2] `https://github.com/joel-beck/readnext`

Bibliography

[1] M. Ware and M. Mabe, "The STM Report: An overview of scientific and scholarly journal publishing," 2015.

[2] A. Khadka, I. Cantador, and M. Fernandez, "Capturing and Exploiting Citation Knowledge for Recommending Recently Published Papers," in *2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, Sep. 2020, pp. 239–244.

[3] M. Khabsa and C. L. Giles, "The Number of Scholarly Documents on the Public Web," *PLOS ONE*, vol. 9, no. 5, p. e93949, May 2014. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0093949

[4] L. Bornmann and R. Mutz, "Growth rates of modern science: A bibliometric analysis based on the number of publications and cited references," May 2014. [Online]. Available: http://arxiv.org/abs/1402.4578

[5] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language Models are Few-Shot Learners," Jul. 2020. [Online]. Available: http://arxiv.org/abs/2005.14165

[6] OpenAI, "GPT-4 Technical Report," Mar. 2023. [Online]. Available: http://arxiv.org/abs/2303.08774

[7] B. Gipp, J. Beel, and C. Hentschel, "Scienstein : A Research Paper Recommender System," 2009. [Online]. Available: https://www.semanticscholar.org/paper/Scienstein-%3A-A-Research-Paper-Recommender-System-Gipp-Beel/deab9886bd1f39bea5b85fa76ca8f705fec9a85c

[8] K. Sugiyama and M.-Y. Kan, "Exploiting potential citation papers in scholarly paper recommendation," in *Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries*,

ser. JCDL '13. New York, NY, USA: Association for Computing Machinery, Jul. 2013, pp. 153–162. [Online]. Available: https://dl.acm.org/doi/10.1145/2467696.2467701

[9] R. Habib and M. T. Afzal, "Sections-based bibliographic coupling for research paper recommendation," *Scientometrics*, vol. 119, no. 2, pp. 643–656, May 2019. [Online]. Available: http://link.springer.com/10.1007/s11192-019-03053-8

[10] B. Gipp and J. Beel, "Citation Proximity Analysis (CPA) – A new approach for identifying related work based on Co-Citation Analysis," 2009.

[11] P. Knoth and A. Khadka, "Can we do better than Co-Citations? - Bringing Citation Proximity Analysis from idea to practice in research article recommendation," 2017.

[12] Y. Mass and H. Roitman, "Ad-hoc Document Retrieval using Weak-Supervision with BERT and GPT2," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Nov. 2020, pp. 4191–4197. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-main.343

[13] A. Cohan, S. Feldman, I. Beltagy, D. Downey, and D. S. Weld, "SPECTER: Document-level Representation Learning using Citation-informed Transformers," May 2020. [Online]. Available: http://arxiv.org/abs/2004.07180

[14] H. A. Mohamed, G. Sansonetti, F. Gasparetti, A. Micarelli, and J. Beel, "BERT, ELMo, USE and InferSent Sentence Encoders: The Panacea for Research-Paper Recommendation?" Sep. 2019.

[15] C. Bhagavatula, S. Feldman, R. Power, and W. Ammar, "Content-Based Citation Recommendation," Feb. 2018. [Online]. Available: http://arxiv.org/abs/1802.08301

[16] N. Agarwal, E. Haque, H. Liu, and L. Parsons, "Research Paper Recommender Systems: A Subspace Clustering Approach," in *Advances in Web-Age Information Management*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, W. Fan, Z. Wu, and J. Yang, Eds., vol. 3739. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 475–491. [Online]. Available: http://link.springer.com/10.1007/11563952_42

[17] C. Nascimento, A. H. Laender, A. S. da Silva, and M. A. Gonçalves, "A source independent framework for research paper recommendation," in *Proceedings of the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries*. Ottawa Ontario Canada: ACM, Jun. 2011, pp. 297–306. [Online]. Available: https://dl.acm.org/doi/10.1145/1998076.1998132

[18] C. Breitinger, "Academic Literature Recommendation using Semantic Feature Analysis," 2023.

[19] F. Ricci, L. Rokach, and B. Shapira, "Recommender Systems: Introduction and Challenges," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, and B. Shapira, Eds. Boston, MA: Springer US, 2015, pp. 1–34. [Online]. Available: https://doi.org/10.1007/978-1-4899-7637-6_1

[20] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, Jun. 2005.

[21] X. Bai, M. Wang, I. Lee, Z. Yang, X. Kong, and F. Xia, "Scientific Paper Recommendation: A Survey," Aug. 2020. [Online]. Available: http://arxiv.org/abs/2008.13538

[22] R. Burke, "Hybrid Recommender Systems: Survey and Experiments," *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, Nov. 2002. [Online]. Available: https://doi.org/10.1023/A:1021240730564

[23] J. Beel, B. Gipp, S. Langer, and C. Breitinger, "Research-paper recommender systems: A literature survey," *International Journal on Digital Libraries*, vol. 17, no. 4, pp. 305–338, Nov. 2016. [Online]. Available: https://doi.org/10.1007/s00799-015-0156-0

[24] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, no. 12, pp. 61–70, Dec. 1992. [Online]. Available: https://dl.acm.org/doi/10.1145/138859.138867

[25] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: An open architecture for collaborative filtering of netnews," in *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, ser. CSCW '94. New York, NY, USA: Association for Computing Machinery, Oct. 1994, pp. 175–186. [Online]. Available: https://dl.acm.org/doi/10.1145/192844.192905

[26] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web*. Hong Kong Hong Kong: ACM, 2001, pp. 285–295. [Online]. Available: https://dl.acm.org/doi/10.1145/371920.372071

[27] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009. [Online]. Available: http://ieeexplore.ieee.org/document/5197422/

[28] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, and D. Sampath, "The YouTube video recommendation system," in *Proceedings of the Fourth ACM Conference on Recommender Systems*. Barcelona Spain: ACM, Sep. 2010, pp. 293–296. [Online]. Available: https://dl.acm.org/doi/10.1145/1864708.1864770

[29] P. Covington, J. Adams, and E. Sargin, "Deep Neural Networks for YouTube Recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*, New York, NY, USA, 2016.

[30] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, Jan. 2003. [Online]. Available: http://ieeexplore.ieee.org/document/1167344/

[31] M. J. Pazzani and D. Billsus, "Content-Based Recommendation Systems," in *The Adaptive Web*, P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds., vol. 4321. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 325–341. [Online]. Available: http://link.springer.com/10.1007/978-3-540-72079-9_10

[32] V. Dey, "Collaborative Filtering Vs Content-Based Filtering for Recommender Systems," Aug. 2021. [Online]. Available: https://analyticsindiamag.com/collaborative-filtering-vs-content-based-filtering-for-recommender-systems/

[33] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative Filtering for Implicit Feedback Datasets," in *Proceedings - IEEE International Conference on Data Mining, ICDM*, Dec. 2008, pp. 263–272.

[34] C. Yang, B. Wei, J. Wu, Y. Zhang, and L. Zhang, "CARES: A ranking-oriented CADAL recommender system," in *Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries*, ser. JCDL '09. New York, NY, USA: Association for Computing Machinery, Jun. 2009, pp. 203–212. [Online]. Available: https://doi.org/10.1145/1555400.1555432

[35] R. Burke, "Hybrid Web Recommender Systems," in *The Adaptive Web*, P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds., vol. 4321. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 377–408. [Online]. Available: http://link.springer.com/10.1007/978-3-540-72079-9_12

[36] J. chiang, "7 Types of Hybrid Recommendation System," Jun. 2021. [Online]. Available: https://medium.com/analytics-vidhya/7-types-of-hybrid-recommendation-system-3e4f78266ad8

[37] D. Roy and M. Dutta, "A systematic review and research perspective on recommender systems," *Journal of Big Data*, vol. 9, no. 1, p. 59, May 2022. [Online]. Available: https://doi.org/10.1186/s40537-022-00592-5

[38] A. Kanakia, Z. Shen, D. Eide, and K. Wang, "A Scalable Hybrid Research Paper Recommender System for Microsoft Academic," in *The World Wide Web Conference*, May 2019, pp. 2893–2899. [Online]. Available: http://arxiv.org/abs/1905.08880

[39] J. Beel and S. Langer, "A Comparison of Offline Evaluations, Online Evaluations, and User Studies in the Context of Research-Paper Recommender Systems," in *Research and Advanced Technology for Digital Libraries*, S. Kapidakis, C. Mazurek, and M. Werla, Eds.

Cham: Springer International Publishing, 2015, vol. 9316, pp. 153–168. [Online]. Available: http://link.springer.com/10.1007/978-3-319-24592-8_12

[40] T. Silveira, M. Zhang, X. Lin, Y. Liu, and S. Ma, "How good your recommender system is? A survey on evaluations in recommendation," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 5, pp. 813–831, May 2019. [Online]. Available: https://doi.org/10.1007/s13042-017-0762-9

[41] K. Rink, "Mean Average Precision at K (MAP@K) clearly explained," Jan. 2023. [Online]. Available: https://towardsdatascience.com/mean-average-precision-at-k-map-k-clearly-explained-538d8e032d2

[42] M. Taifi, "MRR vs MAP vs NDCG Rank-Aware Evaluation Metrics And When To Use Them," Jun. 2020. [Online]. Available: https://medium.com/swlh/rank-aware-recsys-evaluation-metrics-5191bba16832

[43] Wikipedia contributors, "Mean reciprocal rank," *Wikipedia*, Aug. 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Mean_reciprocal_rank&oldid=1107032139

[44] ——, "Discounted cumulative gain," *Wikipedia*, Aug. 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Discounted_cumulative_gain&oldid=1105990923

[45] M. Kaminskas and D. Bridge, "Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems," *ACM Transactions on Interactive Intelligent Systems*, vol. 7, no. 1, pp. 2:1–2:42, Dec. 2016. [Online]. Available: https://doi.org/10.1145/2926720

[46] L. C. Smith, "Citation Analysis," 1981. [Online]. Available: https://hdl.handle.net/2142/7190

[47] I. Marshakova-Shaikevich, "System of Document Connections Based on References," 1973. [Online]. Available: https://www.semanticscholar.org/paper/System-of-Document-Connections-Based-on-References-Marshakova-shaikevich/2d871489eb7288dd1bec4be99bc363efd4933d48

[48] M. M. Kessler, "Bibliographic coupling between scientific papers," *American Documentation*, vol. 14, no. 1, pp. 10–25, Jan. 1963. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/asi.5090140103

[49] H. Small, "Co-citation in the scientific literature: A new measure of the relationship between two documents," *Journal of the American Society for Information Science*, vol. 24, no. 4, pp. 265–269, Jul. 1973. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/asi.4630240406

[50] N. Tran, P. Alves, S. Ma, and M. Krauthammer, "Enriching PubMed Related Article Search with Sentence Level Co-citations," *AMIA Annual Symposium Proceedings*, vol. 2009, pp. 650–654, 2009. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2815371/

[51] S. Liu and C. Chen, "The Effects of Co-citation Proximity on Co-citation Analysis," 2011. [Online]. Available: https://www.semanticscholar.org/paper/The-Effects-of-Co-citation-Proximity-on-Co-citation-Liu-Chen/ac860530cc300768b963a90a46cbffb9b20ef540

[52] B. Gipp, N. Meuschke, and C. Breitinger, "Citation-based plagiarism detection : Practicability on a large-scale scientific corpus," 2014. [Online]. Available: http://kops.uni-konstanz.de/handle/123456789/30291

[53] M. Schwarzer, M. Schubotz, N. Meuschke, C. Breitinger, V. Markl, and B. Gipp, "Evaluating Link-based Recommendations for Wikipedia," *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries*, pp. 191–200, Jun. 2016. [Online]. Available: https://dl.acm.org/doi/10.1145/2910896.2910908

[54] Z. Ali, G. Qi, P. Kefalas, W. A. Abro, and B. Ali, "A graph-based taxonomy of citation recommendation models," *Artificial Intelligence Review*, vol. 53, no. 7, pp. 5217–5260, Oct. 2020. [Online]. Available: https://doi.org/10.1007/s10462-020-09819-4

[55] D. Jurafsky and J. Martin, "Speech and Language Processing," Jan. 2022.

[56] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," Sep. 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[57] Z. S. Harris, "Distributional Structure," WORD, vol. 10, no. 2-3, pp. 146–162, Aug. 1954. [Online]. Available: http://www.tandfonline.com/doi/full/10.1080/00437956.1954.11659520

[58] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, Jul. 2008.

[59] G. Salton and C. Buckley, "Term Weighting Approaches in Automatic Text Retrieval," Nov. 1987. [Online]. Available: https://ecommons.cornell.edu/handle/1813/6721

[60] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[61] S. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford, "Okapi at TREC-3," in *Overview of the Third Text REtrieval Conference (TREC-3)*, Jan. 1995, pp. 109–126. [Online]. Available: https://www.microsoft.com/en-us/research/publication/okapi-at-trec-3/

[62] Y. Lv and C. Zhai, "Lower-bounding term frequency normalization: 20th ACM Conference on Information and Knowledge Management, CIKM'11," *CIKM'11 - Proceedings of the 2011 ACM International Conference on Information and Knowledge Management*, pp. 7–16, 2011. [Online]. Available: http://www.scopus.com/inward/record.url?scp=83055187814&partnerID=8YFLogxK

[63] A. Trotman, A. Puurula, and B. Burgess, "Improvements to BM25 and Language Models Examined," in *Proceedings of the 2014 Australasian Document Computing Symposium*. Melbourne VIC Australia: ACM, Nov. 2014, pp. 58–65. [Online]. Available: https://dl.acm.org/doi/10.1145/2682862.2682863

[64] J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1532–1543. [Online]. Available: http://aclweb.org/anthology/D14-1162

[65] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," Jun. 2017. [Online]. Available: http://arxiv.org/abs/1607.04606

[66] C. McCormick, "Word2Vec Tutorial - The Skip-Gram Model," 2016. [Online]. Available: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

[67] J. Alammar, "The Illustrated Word2vec," Mar. 2019. [Online]. Available: https://jalammar.github.io/illustrated-word2vec/

[68] T. Ruas, W. Grosky, and A. Aizawa, "Multi-sense embeddings through a word sense disambiguation process," *Expert Systems with Applications*, vol. 136, pp. 288–303, Dec. 2019. [Online]. Available: http://arxiv.org/abs/2101.08700

[69] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," Dec. 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[70] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv:1810.04805 [cs]*, May 2019. [Online]. Available: http://arxiv.org/abs/1810.04805

[71] I. Beltagy, K. Lo, and A. Cohan, "SciBERT: A Pretrained Language Model for Scientific Text," Sep. 2019. [Online]. Available: http://arxiv.org/abs/1903.10676

[72] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The Long-Document Transformer," Dec. 2020. [Online]. Available: http://arxiv.org/abs/2004.05150

[73] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," May 2016. [Online]. Available: http://arxiv.org/abs/1409.0473

[74] J. Alammar, "The Illustrated Transformer," Jun. 2018. [Online]. Available: https://jalammar.github.io/illustrated-transformer/

[75] J. Vig, "A Multiscale Visualization of Attention in the Transformer Model," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 37–42. [Online]. Available: https://aclanthology.org/P19-3007

[76] ——, "Deconstructing BERT, Part 2: Visualizing the Inner Workings of Attention," Apr. 2022. [Online]. Available: https://towardsdatascience.com/deconstructing-bert-part-2-visualizing-the-inner-workings-of-attention-60a16d86b5c1

[77] C. McCormick, "BERT Word Embeddings Tutorial," May 2019. [Online]. Available: https://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/#sentence-vectors

[78] J. Alammar, "The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)," Dec. 2018. [Online]. Available: https://jalammar.github.io/illustrated-bert/

[79] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Ł. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," Oct. 2016. [Online]. Available: http://arxiv.org/abs/1609.08144

[80] S. Palachy, "Document Embedding Techniques," Jun. 2022. [Online]. Available: https://towardsdatascience.com/document-embedding-techniques-fed3e7a6a25d

[81] Q. V. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," May 2014. [Online]. Available: http://arxiv.org/abs/1405.4053

[82] M. Ostendorff, T. Ruas, M. Schubotz, G. Rehm, and B. Gipp, "Pairwise Multi-Class Document Classification for Semantic Relations between Wikipedia Articles," Mar. 2020. [Online]. Available: http://arxiv.org/abs/2003.09881

[83] M. Färber and A. Sampath, "HybridCite: A Hybrid Model for Context-Aware Citation Recommendation," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*, Aug. 2020, pp. 117–126. [Online]. Available: http://arxiv.org/abs/2002.06406

[84] T. Ruas, C. H. P. Ferreira, W. Grosky, F. O. de França, and D. M. R. Medeiros, "Enhanced word embeddings using multi-semantic representation through lexical chains," *Information Sciences*, vol. 532, pp. 16–32, Sep. 2020. [Online]. Available: http://arxiv.org/abs/2101.09023

[85] M. Ostendorff, T. Blume, T. Ruas, B. Gipp, and G. Rehm, "Specialized Document Embeddings for Aspect-based Similarity of Research Papers," Mar. 2022. [Online]. Available: http://arxiv.org/abs/2203.14541

[86] J. Briggs, "Semantic Search: Measuring Meaning From Jaccard to Bert," 2020. [Online]. Available: https://www.pinecone.io/learn/semantic-search/

[87] A. Singhal, "Modern Information Retrieval: A Brief Overview," 2001.

[88] P. Jaccard, "Etude de la distribution florale dans une portion des Alpes et du Jura," *Bulletin de la Societe Vaudoise des Sciences Naturelles*, vol. 37, pp. 547–579, Jan. 1901.

[89] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," Aug. 2019. [Online]. Available: http://arxiv.org/abs/1908.10084

[90] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," Jan. 2020. [Online]. Available: http://arxiv.org/abs/1906.08237

[91] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners," 2018.

[92] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y.-H. Sung, B. Strope, and R. Kurzweil, "Universal Sentence Encoder," Apr. 2018. [Online]. Available: http://arxiv.org/abs/1803.11175

[93] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised Learning of Universal Sentence Representations from Natural Language Inference Data," Jul. 2018. [Online]. Available: http://arxiv.org/abs/1705.02364

[94] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," Mar. 2018. [Online]. Available: http://arxiv.org/abs/1802.05365

[95] M. Ostendorff, T. Ruas, T. Blume, B. Gipp, and G. Rehm, "Aspect-based Document Similarity for Research Papers," Oct. 2020. [Online]. Available: http://arxiv.org/abs/2010.06395

[96] Z. Akkalyoncu Yilmaz, S. Wang, W. Yang, H. Zhang, and J. Lin, "Applying BERT to Document Retrieval with Birch," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 19–24. [Online]. Available: https://aclanthology.org/D19-3004

[97] J. P. Wahle, T. Ruas, S. Mohammad, and B. Gipp, "D3: A Massive Dataset of Scholarly Metadata for Analyzing the State of Computer Science Research," in *Proceedings of the Thirteenth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, Jun. 2022, pp. 2642–2651. [Online]. Available: https://aclanthology.org/2022.lrec-1.283

[98] M. Abdalla, J. P. Wahle, T. Lima Ruas, A. Névéol, F. Ducel, S. Mohammad, and K. Fort, "The Elephant in the Room: Analyzing the Presence of Big Tech in Natural Language Processing Research," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 13 141–13 160. [Online]. Available: https://aclanthology.org/2023.acl-long.734

[99] T. Ruas, J. P. Wahle, L. Küll, S. M. Mohammad, and B. Gipp, "CS-Insights: A System for Analyzing Computer Science Research," Jan. 2023. [Online]. Available: http://arxiv.org/abs/2210.06878

[100] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385 [cs]*, Dec. 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[101] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, "A Survey of Neuromorphic Computing and Neural Networks in Hardware," May 2017. [Online]. Available: http://arxiv.org/abs/1705.06963

[102] M. Honnibal and M. Johnson, "An improved non-monotonic transition system for dependency parsing," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 1373–1378. [Online]. Available: https://aclweb.org/anthology/D/D15/D15-1162

[103] M.-T. Luong, H. Pham, and C. D. Manning, "Effective Approaches to Attention-based Neural Machine Translation," Sep. 2015. [Online]. Available: http://arxiv.org/abs/1508.04025

[104] D. Brown, "Rank-BM25: A collection of BM25 algorithms in python," Zenodo, 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4520057

[105] R. Rehurek and P. Sojka, "Gensim–Python framework for vector space modelling," *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, vol. 3, no. 2, 2011.

[106] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning Word Vectors for 157 Languages," Mar. 2018. [Online]. Available: http://arxiv.org/abs/1802.06893

[107] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "HuggingFace's Transformers: State-of-the-art Natural Language Processing," Jul. 2020. [Online]. Available: http://arxiv.org/abs/1910.03771

[108] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le, "Unsupervised Data Augmentation for Consistency Training," Nov. 2020. [Online]. Available: http://arxiv.org/abs/1904.12848

[109] M. Joshi, O. Levy, D. S. Weld, and L. Zettlemoyer, "BERT for Coreference Resolution: Baselines and Analysis," Dec. 2019. [Online]. Available: http://arxiv.org/abs/1908.09091

[110] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," Jul. 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[111] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books," Jun. 2015. [Online]. Available: http://arxiv.org/abs/1506.06724

[112] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi, "Defending Against Neural Fake News," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://papers.nips.cc/paper_files/paper/2019/hash/3e9f0fc9b2f89e043bc6233994dfcf76-Abstract.html

[113] T. H. Trinh and Q. V. Le, "A Simple Method for Commonsense Reasoning," Sep. 2019. [Online]. Available: http://arxiv.org/abs/1806.02847

[114] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," Jun. 2016. [Online]. Available: http://arxiv.org/abs/1508.07909

[115] Y. Zhang, H. Ding, Z. Shui, Y. Ma, J. Zou, A. Deoras, and H. Wang, "Language Models as Recommender Systems: Evaluations and Limitations," 2021.

[116] Y. Hou, J. Zhang, Z. Lin, H. Lu, R. Xie, J. McAuley, and W. X. Zhao, "Large Language Models are Zero-Shot Rankers for Recommender Systems," May 2023. [Online]. Available: http://arxiv.org/abs/2305.08845

[117] J. Li, W. Zhang, T. Wang, G. Xiong, A. Lu, and G. Medioni, "GPT4Rec: A Generative Framework for Personalized Recommendation and User Interests Interpretation," Apr. 2023. [Online]. Available: http://arxiv.org/abs/2304.03879

[118] J. Liu, C. Liu, P. Zhou, R. Lv, K. Zhou, and Y. Zhang, "Is ChatGPT a Good Recommender? A Preliminary Study," Jun. 2023. [Online]. Available: http://arxiv.org/abs/2304.10149

[119] J. Lin, X. Dai, Y. Xi, W. Liu, B. Chen, X. Li, C. Zhu, H. Guo, Y. Yu, R. Tang, and W. Zhang, "How Can Recommender Systems Benefit from Large Language Models: A Survey," Jun. 2023. [Online]. Available: http://arxiv.org/abs/2306.05817

[120] P. Liu, L. Zhang, and J. A. Gulla, "Pre-train, Prompt and Recommendation: A Comprehensive Survey of Language Modelling Paradigm Adaptations in Recommender Systems," Mar. 2023. [Online]. Available: http://arxiv.org/abs/2302.03735

Appendix

Large Language Models as Recommenders

The rise of modern LLMs such as GPT4 [6] has revolutionized the field of NLP, demonstrating impressive capabilities in a variety of tasks. This section provides an overview of the potential of LLMs in the context of recommender systems. In recent years, several studies have explored using LLMs in recommendation tasks, showing promising results [115–117]. Although this thesis does not utilize generative LLMs, this discussion serves to contextualize the broader landscape of advancements in the field and to provide insights into potential future directions.

Li et al. [117] introduced a novel generative framework for personalized recommendations. Their model, inspired by search engines and built on Transformer models, views items as queries. It then adopts a generative method to estimate the conditional probability of a user's next action based on their history. This method effectively captures the content information of items, interprets user interests, and adapts to changing item inventories.

Liu et al. [118] examined the potential of ChatGPT[3] as a general-purpose recommender model. They framed recommendation tasks as a conversation between the user and the system, leveraging ChatGPT to generate recommendations from the user's input. Their findings indicate that ChatGPT is comparable to traditional recommendation methods for several tasks and even excels in tasks that require a deeper understanding of user preferences.

Despite the promising potential of LLMs, their use in recommender systems presents challenges. These encompass the high computational cost of training and deploying such models, the need for vast amounts of data for pre-training, and the model's limited interpretability, which hampers understanding and justification of their recommendations [119, 120]. Moreover, Liu et al. [118] highlight that LLMs often recommend popular items, leading to a lack of diversity and novelty in the recommendations. Finally, the way users phrase their input greatly influences the recommendations, causing inconsistent outcomes [115].

[3] https://openai.com/blog/chatgpt

Hybrid Recommender Evaluation

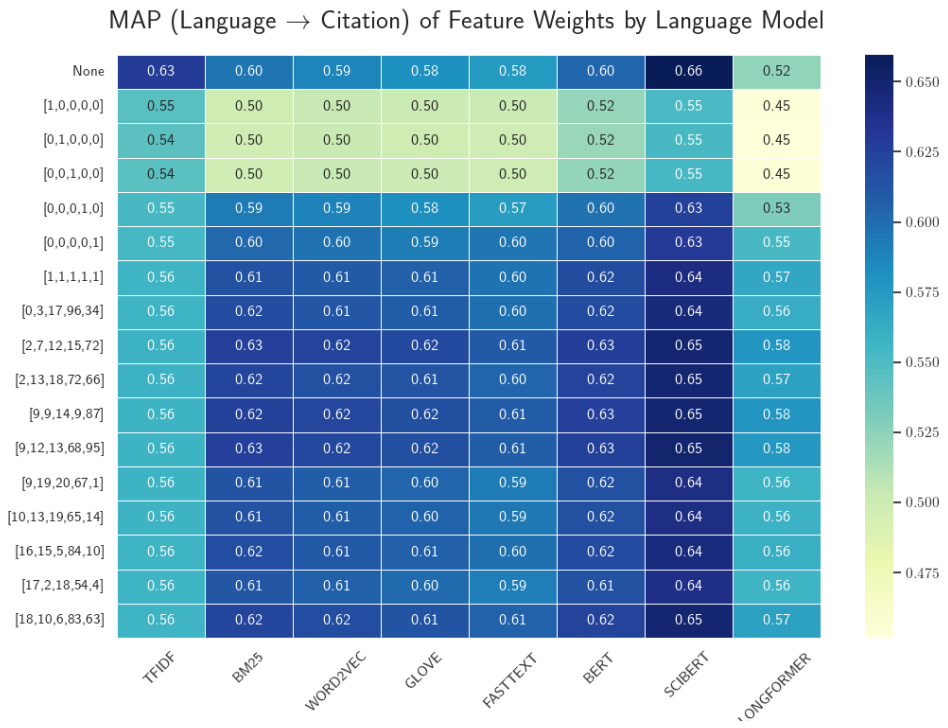


Figure 1: Performance Evaluation of the L → C Hybrid Recommender using different feature weights and language models. Darker colors correspond to higher MAP values. The first row displays the MAP values for the Language Recommender candidates before re-ranking. Subsequent rows show MAP values for each language model after being re-ranked by the Citation Recommender with the corresponding weight vector. The trend concerning which language models improve or degrade due to re-ranking is consistent with the results in Figure 4.7. With the exception of TF-IDF, weight vectors with zero weight for co-citation analysis and bibliographic coupling score notably underperform. Compared to the feature weights ranking for candidate selection (Figure 4.3), the unit weight vectors $[0, 0, 0, 1, 0]$ and $[0, 0, 0, 0, 1]$ perform worse relative mixed non-zero weight vectors. This observation is consistent across all language models, except for TF-IDF.