Exploring Sequential Recommendation for Scientific Reading

Anonymous ACL submission

Abstract

The accelerating growth of the scientific literature makes it increasingly difficult for researchers to decide what to read next. Yet most scientific paper recommendation systems still treat recommendation as a static ranking problem, limiting their ability to capture how a researcher's interests evolve through prior reading. To address this limitation, we formulate scientific paper recommendation as a sequential recommendation task: given a researcher's prior reading history, the goal is to predict *the next paper they should read*. We introduce a method for constructing reading sequences from article introductions, using cited papers as proxies for scholarly reading sequences. We use the resulting sequences to evaluate widely used retrieval models in a next-paper prediction setting and to study training strategies for this task. Our results suggest that while reading history has a noticeable impact, sequential modeling gains are inconsistent across embedding models and prediction slices. Closing the gap with the strongest retrieval baselines remains an open challenge, particularly for final target prediction.

1 Introduction

The rapid growth of scientific publications has made it increasingly difficult for researchers to decide what to read when entering a new field. Survey papers provide valuable overviews, but they are often unavailable, outdated, or incomplete. Academic search engines and scientific paper recommendation systems can retrieve relevant papers, but they typically present results as ranked lists, offering limited guidance on how those papers should be read over time (Beel et al., 2016; Kreutz and Schenkel, 2022; Flicke et al., 2025).

Prior work on scholarly recommendation has explored ways to support this process. Reading list generation recommends sets of papers that provide an entry point into a new research area (Ekstrand et al., 2010; Gordon et al., 2017). Other approaches go beyond selecting papers by organizing them according to topic evolution, methodological dependencies, or prerequisite relations (Sesagiri Raamkumar et al., 2017; Figueira et al., 2019; Ding et al., 2022). While these methods move beyond simple relevance ranking, they typically define reading order using reader-independent criteria, such as citation structure, topical progression, or manually specified constraints. As a result, they provide useful general guidance, but do not directly model how an individual researcher's interests and knowledge evolve through prior reading.

A promising alternative lies in sequential recommendation, a branch of recommender systems that models a user's interaction history and adapts its predictions to the individual user. Sequential models have been widely studied in domains such as product, media, and behavior recommendation, where large-scale user interaction logs are available (Kang and McAuley, 2018; Sun et al., 2019; Boka et al., 2024). Scientific reading, however, presents a different setting: reading sequences are sparse, difficult to observe, and rarely available at scale. This lack of directional data has limited the direct application of sequential recommendation methods to scientific paper recommendation.

In this paper, we formulate scientific paper recommendation as a next-paper prediction task. Given a researcher's prior reading history, the goal is to predict the next paper they should read. To make this setting practical without relying on large-scale user logs, we propose an automatic method for constructing reading sequences from existing paper collections. Our key idea is to use citation sequences in article introductions as proxies for scholarly reading sequences: introductions often present prior work in an order that reflects how a reader can build up the context needed to understand a contribution.

Our contributions are:

- The introduction of a new sequential recommendation task for scholarly articles, along with an evaluation framework specifically adapted to this setting.

- A new dataset of article reading sequences automatically annotated with queries, where each sequence is paired with a keyword-based and a question-based query derived from the target article and serving as a proxy for the researcher's information need. The dataset is publicly available on HuggingFace.

- A comparative evaluation of sparse, dense and hybrid model as well as task-specific sequential recommendation models built on top of widely used scholarly retrievers.

We release our code on GitHub[1] and our dataset on HuggingFace[2].

2 Sequential Recommendation of Scientific Articles

2.1 Task Definition

We propose a new sequential recommendation task tailored to scientific literature.

Let D be a research domain, Q_d an initial query related to this domain formulated by a user, and $H_d = (a_1, \ldots, a_n)$ with $a_i \in D$ be the user's reading history. Sequential recommendation aims to predict the next relevant article by combining these two signals such as:

$$a^*_{n+1} = \arg\max_{a \in D} \; P(a \mid Q_d, H_d).$$

2.2 Temporal Data Split

To ensure minimal data overlap between the training, validation, and test sets, we use a temporal partitioning of our data (Gusak et al., 2025).

We restrict the training data to sequences that occur before a specified cutoff date, while the validation and test sets include only sequences that occur after that date. The temporal split is conducted on each sequence's target article publication date, ensuring that no full test sequence is part of a training subsequence.

[1] https://anonymous.4open.science/r/ACL-citepath-AD12/

[2] https://huggingface.co/datasets/taln-ls2n/ACL-rs

2.3 Sequential Evaluation

The evaluation paradigm must take into account the sequential nature of the reading history. Evaluating only the final article in a reading history would amount to ignoring the intermediate steps that led to the construction of the sequence. To this end, we draw inspiration from the successive evaluation strategy proposed by Gusak et al. (2025), which we adapt to the specific characteristics of our data.

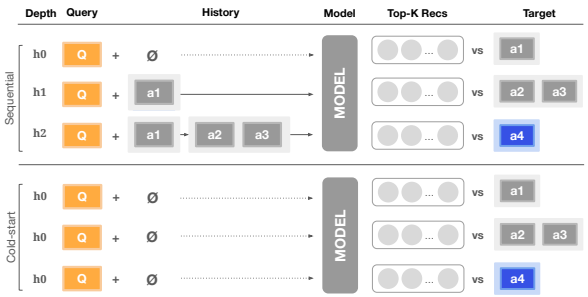


Figure 1: Comparison between sequential and cold-start evaluation for same input sequence.

Figure 1 illustrate differences between a linear and sequential evaluation protocol. We evaluate each model on the full set of subsequences derived from each query–sequence pair. For a sequence of N co-citation groups leading to a target article, we generate N evaluation instances, each corresponding to a different history depth $h \in \{0, \ldots, N-1\}$. At depth h, the model is given the query and the first h co-citation groups as history, and must predict the $(h+1)$-th group. This protocol yields two complementary evaluation slices. The subsequence slice aggregates performance over all intermediate steps ($h < N-1$), measuring how well the model predicts co-citation groups from partial history. The target slice evaluates the final step ($h = N-1$), where the model must predict the article that closes the sequence given its full reading history. To quantify the contribution of sequential history, we additionally report a cold start baseline for each slice: the same targets are evaluated using predictions generated from empty history ($h = 0$), isolating the gain attributable to accumulated context.

3 Constructing a Dataset of Reading Sequences

3.1 What are Reading Sequences?

The CARS (Create A Research Space) model (Swales, 1990) describes how authors

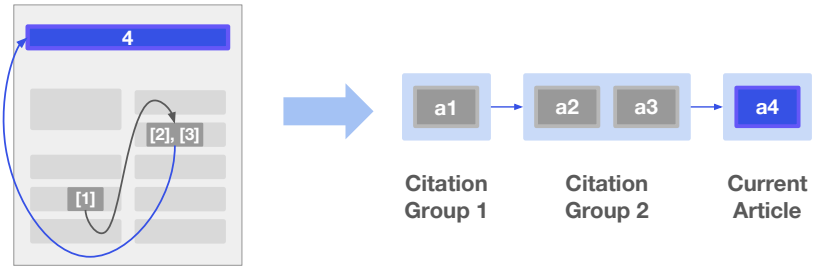


Figure 2: Procedure for extracting a reading sequence from an initial article.

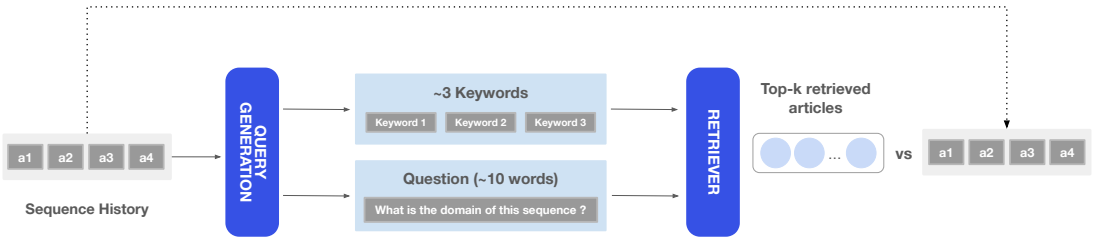


Figure 3: Protocol for automated annotation of the information need from existing reading sequences.

position their work within a research field, identifying three rhetorical moves: surveying prior work, identifying a gap, and presenting a novel contribution. We hypothesize that this progression mirrors a natural reading sequence, and use it as a principled basis for constructing reading sequences from scientific articles. From this perspective, we assume the following:

1. Because authors intentionally structure introductions to establish background knowledge and motivate their work, the ordering of citations **may encode a meaningful progression through the underlying literature**.

2. When an article introduction provides a synthesis of a research field, it generally builds upon previously cited works to progressively narrow the discussion toward the addressed research problem, thereby creating a **coherent progression of ideas, in which the citing article represents the final step**.

Together, these properties suggest that citation sequences extracted from scientific introductions reflect the rhetorical organization of the text, providing coherent entry points into a research area and naturally supporting literature navigation. We therefore investigate the use of these sequences as a proxy for reading lists.

Prior work has shown that citation context provides strong signals about citation function and rhetorical intent (Teufel et al., 2006; Abu-Jbara et al., 2013), reflecting the tendency of authors to cite related work together and isolated references to serve more specific argumentative roles. This motivates using multi-citation groups (one or more citations in same citation group) as prediction targets, allowing the model to recover not just individual references but coherent clusters of related work.

We propose a method for automatically constructing a reading sequence that enables the development of expertise on a specific research topic (see fig. 2). The reading history of a research domain, extracted from an initial article, is represented as an ordered list of multi-citation groups: for each article in a group, the set of articles in the next group constitutes the most probable set of subsequent readings. Thus, for any article, the introduction is treated as a reading history, and the article itself as the continuation of that history.

3.2 Data sources

For data collection, we focus on the field of Natural Language Processing (NLP) and use the Association for Computational Linguistics (ACL) Anthology[3] as our primary source of papers from a

[3] https://github.com/acl-org/acl-anthology

wide range of NLP venues. The full collection consists of 118,325 papers across the NLP literature. From this corpus, we then select papers published in major venues of the ACL, whose peer-review process ensures high publication quality. We extract the full text and bibliographic references of these selected papers using GROBID[4], a tool for structured extraction of scientific documents. Since GROBID outputs may be noisy or imprecise, we subsequently align the extracted citations with the ACL Anthology collection to ensure their validity and consistency.

We focus on the first section of each paper, which largely corresponds to the introduction. Within this section, we identify multi-citation groups and their citation contexts, from which we extract both content and ordering information. To construct the actual sequences, we remove duplicates by retaining only the first occurrence of each citation. We keep sequences of citation groups whose length ranges from 3 to 20 papers, following prior work on scientific reading lists (Aubert-Béduchaud et al., 2025). This procedure results in the identification of 30,224 reading sequences.

4 Query Generation for Reading Sequences

The procedure in section 3.2 automatically extracts reading sequences from papers whose introductions contain citations. However, the resulting dataset contains only the sequences themselves, it does not include an explicit user request, query, or information need that could serve as input to a recommendation model. To address this practical limitation, we augment each extracted sequence with automatically generated queries.

Since no explicit queries are available, we generate them automatically from the papers in each sequence. Rather than using only the initial paper, we aim to capture the broader context of the full sequence with a short, generic query (Figure 3). This query serves as a plausible input from which the sequence could be retrieved or approximated. We generate two types of queries, following statistics reported in scientific query collections: keyword queries of about three terms (Boteva et al., 2016; Singh et al., 2023), and natural language questions of about ten words (Voorhees et al., 2021; Nguyen et al., 2022). To this end, we follow statistics reported in scientific query collections, targeting approximately three keywords (Boteva et al., 2016; Singh et al., 2023) or a natural language question of about ten words (Voorhees et al., 2021; Nguyen et al., 2022).

4.1 Candidate Query Generators

We compare several approaches for generating queries from reading sequences. Each article is represented by its title and abstract.

Statistical keyword extraction We use MultiPartiteRank (Boudin, 2018)[5] to extract keywords independently from each article. We then aggregate keywords across the sequence by frequency, yielding sequence-level keyword queries.

Document expansion. We use docT5query (Nogueira et al., 2019)[6] to generate one candidate query per article in the sequence. We retain the most representative query by computing TF-IDF cosine similarity between each candidate query and all articles in the sequence.

Large language models. We prompt GPT-4.1[7], Gemma3[8], Ministral3[9], Qwen3[10] to generate both keyword queries and natural language questions from the sequence. Full instructions are available in appendix A.

4.2 Evaluating Query Generation

We evaluate query generators by measuring whether their queries can recover the papers in the corresponding reading sequence from the full collection. This gives us a *retrieval-based validation task*: better queries should rank the sequence papers higher.

We use PyTerrier[11] to evaluate two retrieval settings: sparse retrieval with BM25 and dense retrieval with `allenai-specter`[12]. To limit computational cost, we run this comparison on a random subsample of 1,000 sequences. We select query

[4] https://github.com/allenai/grobid
swh:1:dir:324a18113b0c7624a66a21550bd0e8522e328b4e

[5] https://github.com/boudinfl/pke/blob/master/pke/unsupervised/graph_based/multipartiterank.py

[6] https://github.com/castorini/docTTTTTquery

[7] https://openai.com/index/gpt-4-1/

[8] https://huggingface.co/collections/google/gemma-3-release

[9] https://huggingface.co/collections/mistralai/ministral-3

[10] https://huggingface.co/collections/Qwen/qwen3

[11] https://pyterrier.readthedocs.io

[12] https://huggingface.co/sentence-transformers/allenai-specter

generators based on the best trade-off between retrieval effectiveness and query length. Retrieval effectiveness is measured with Recall@100 (R@100) and Mean Average Precision (MAP). For query length, we measure the deviation from the target length using z-scores. We compute the standard deviation of each query length, then normalize each value by measuring its distance from the target in standard deviation units. Based on the absolute z-score, deviation is classified as *low* (within 1 std), *moderate* (within 2), or *high* (beyond 2).

	R@100		MAP		Length
	dense	sparse	dense	sparse	#words
Keywords					
multipartite rank	**0.116**[2]	0.337	**0.033**[1]	**0.135**[1]	3.981
gpt-4.1-mini	0.082	0.332	0.013	0.081	3.635
gpt-4.1	0.065	**0.360**[3]	0.009	0.083	3.086
google/gemma-3-4b-it	0.062	0.318	0.008	0.075	3.449
google/gemma-3-12b-it	0.054	0.341	0.008	0.089	3.163
mistralai/Ministral-3-8B-Instruct-2512	**0.149**[1]	**0.388**[1]	**0.029**[2]	**0.119**[2]	5.324
mistralai/Ministral-3-14B-Instruct-2512	0.083	0.325	0.012	0.079	3.834
Qwen/Qwen3-4B	**0.086**[3]	**0.372**[2]	**0.015**[3]	**0.112**[3]	3.616
Qwen/Qwen3-14B	0.068	0.339	0.009	0.091	3.188
Question					
castorini/doc2query-t5-base-msmarco	0.135	0.306	0.040	0.121	6.050
gpt-4.1-mini	**0.284**[1]	**0.438**[1]	0.059	0.149	10.651
gpt-4.1	0.247	**0.422**[2]	0.048	0.128	10.108
google/gemma-3-4b-it	0.203	0.331	0.039	0.100	9.958
google/gemma-3-12b-it	0.232	0.380	0.043	0.124	10.205
mistralai/Ministral-3-8B-Instruct-2512	**0.280**[2]	0.409	**0.060**[3]	0.148	10.991
mistralai/Ministral-3-14B-Instruct-2512	0.262	**0.415**[3]	0.054	**0.152**[3]	10.893
Qwen/Qwen3-4B	0.268	**0.438**[1]	**0.068**[1]	**0.190**[1]	8.123
Qwen/Qwen3-14B	**0.274**[3]	**0.422**[2]	**0.061**[2]	**0.170**[2]	10.439

Table 1: Results by model and evaluation strategy. Best 3 models on R@100 and MAP are **highlighted**. Query length deviation from the target is computed using z-scores: low, moderate, high.

Table 1 summarizes the results on the subsample. For keyword queries, Qwen3-4B provides the best balance between retrieval effectiveness (ranking second or third across most metrics) and query length control (3.6 words). Ministral-3-8B performs well at first glance, but its performance is partly driven by longer queries, averaging 5.3 words instead of the target length of 3. MultiPartiteRank shows a similar pattern, although to a lesser extent. For question generation, Qwen3-14B performs strongly across metrics and is comparable to GPT-4.1-mini. Qwen-4B is also competitive, but tends to generate overly short questions (8.1 words). We therefore use **Qwen3-4B** to generate keyword queries and **Qwen3-14B** to generate natural language questions in the remainder of the paper.

4.3 Query Quality Control

We conduct additional checks to assess whether the selected generators remain reliable at larger scale. A key challenge is that queries are generated from ordered sequences of papers, whose length may affect the model's ability to summarize the underlying reading intent. Short sequences provide limited evidence about the broader intent, while longer sequences require the model to aggregate information across more documents. Analysis of this effect (see Figure 5 in Appendix) shows that the first document is generally well captured, while performance decreases as sequences grow longer, with a slight recovery for the last document. This pattern is consistent across both query types and input ordering strategies.

Because the queries are generated automatically, we further apply a retrieval-based filtering step to improve dataset quality. We keep only sequences for which the generated queries retrieves at least one original sequence element in the top 1,000 results. Table 2 reports retrieval results after this filtering step. This filtering step does not guarantee perfect query quality, but removes clear failures where the query is unrelated to the sequence under our retrieval criterion.

	Perfect	**Partial**	**Invalid**
Sequences	13,722	16,487	15
Items per sequence	4.45	6.75	3.93
Keywords per sequence	3.57	3.61	3.80
Keyword length (chars)	32.82	33.62	34.93
Question length (words)	10.57	10.34	10.80
Question length (chars)	75.75	74.09	72.07

Table 2: Query characteristics by consistency level.

The final dataset contains **30,209** reading sequences, each paired with two generated queries: a keyword query and a natural language question. We split the dataset into train, validation, and test sets using an approximate 80-10-10 ratio, yielding 25,276, 2,467, and 2,466 sequences, respectively.

5 Next Article Prediction

We evaluate models for sequential scientific article recommendation. Given an input query and a reading history composed of previously read papers, the task is to recommend the next article in the sequence.

5.1 Experimental setup

We conduct experiments on the ACL Anthology collection, consisting of 118,325 articles. We use R@10 as the primary metric because the task is to recommend a small set of useful papers rather than optimize the full ranking of the collection. This

reflects a realistic setting in which a researcher is shown a short list of suggested articles.

Because our dataset is built from citations in introductions, each reading sequence contains multiple prediction steps: intermediate co-citation groups and the final citing *target* article. We therefore report performance at intermediate steps (*subsequences*), final step (*target*), as well as aggregate performance across all steps. Scores are computed for each unique citing article associated with a reading sequence.

5.2 Retrieval Baselines

We consider two retrieval baselines: sparse retrieval and dense retrieval. Both share a common interpolation mechanism between the current query signal and the accumulated reading history, controlled by a parameter $\alpha \in [0, 1]$.

Sparse retrieval. We use BM25, which represents queries and documents in a sparse term-weight space. The final score is computed by interpolating between the BM25 score obtained from the query $\mathbf{s}_{\text{query}}$, with the mean BM25 score obtained from the articles in the reading history $\mathbf{s}_{\text{history}}$:

$$\mathbf{s} = (1 - \alpha) \cdot \mathbf{s}_{\text{query}} + \alpha \cdot \mathbf{s}_{\text{history}} \quad (1)$$

Dense retrieval. We encode queries and documents into continuous embeddings and retrieve documents by similarity. The search representation is computed by interpolating the query embedding $\mathbf{e}_{\text{query}}$ with the mean embedding of the articles in the reading history, $\mathbf{e}_{\text{history}}$, similarly to eq. (1).

Hybrid retrieval combines sparse and dense signals by interpolating their final score vectors:

$$\mathbf{s}_{\text{hybrid}} = (1 - \beta) \cdot \mathbf{s}_{\text{sparse}} + \beta \cdot \mathbf{s}_{\text{dense}} \quad (2)$$

where $\mathbf{s}_{\text{sparse}}$ and $\mathbf{s}_{\text{dense}}$ are the interpolated sparse and dense score vectors defined above, each with their own history interpolation parameter α. The parameter $\beta \in [0, 1]$ controls the trade-off between the two retrieval modalities.

We compare three embedding models: SPECTER2[13] (110M), a domain-specific model pre-trained on scientific literature, and two general-purpose models with larger parameter counts, E5-large-v2[14] (0.3B) and Qwen3 Embeddings[15] (8B). This comparison allows us to assess whether model scale can compensate for weaker domain specificity. The interpolation parameter α is tuned on the validation set by maximizing R@10, and the selected value is used for test evaluation (see appendix D).

5.3 BERT4Scholar Architecture

We propose a new model architecture aimed at tackling the sequential recommendation of scientific articles. Our approach draws inspiration from the BERT4Rec architecture (Sun et al., 2019), with additional components designed to better suit the specific requirements of our setting. Figure 6 provides an overview of the architecture.

Semantic article representations. Unlike standard sequential recommenders that learn item embeddings from IDs, BERT4Scholar represents each article using its title and abstract. Article embeddings are produced by a frozen document encoder and used as input item representations. This provides the model with semantic priors from scientific text while reducing training cost. The same encoder is used to represent all candidate articles in the collection.

Query-aware sequence modeling. Unlike the original BERT4Rec, our architecture incorporates query intent directly into the sequence representation. The query is encoded in the same semantic space as the article embeddings. At each sequence position, a learned gate $\alpha \in [0, 1]$ controls the balance between the query representation and the reading-history representation.

Co-citation group embeddings. Because co-citations can indicate semantic relationships, we enhance sequence positional embeddings with a group embedding that encodes the co-citation group position of each item in the sequence.

Output Layer. The transformer hidden states are projected back into the article embedding space using a linear layer followed by a GELU activation. Logits are then computed via a tied-weight inner product with the frozen article embedding matrix, together with an additional learned per-article bias term.

5.4 BERT4Scholar Parameters

We instantiate BERT4Scholar with the same three pretrained encoders used in the retrieval baselines: SPECTER2, E5-large-v2, and Qwen3-Embedding.

[13] https://huggingface.co/allenai/specter2
[14] https://huggingface.co/intfloat/e5-large-v2
[15] https://huggingface.co/Qwen/Qwen3-Embedding-8B

This allows us to assess whether training on sequential reading data improves over using the encoders directly for retrieval.

We use a batch size of 128, training for up to 100 epochs with early stopping (patience=10) based on target R@10 on the validation set, a maximum sequence length of 20, and a fixed seed. Optimization uses AdamW with learning rate 5^{-5}, weight decay 10^{-2}, and 5% linear warmup.

Training follows a two-phase process: the model first is trained on all subsequences using last-group prediction to learn sequential representations across history depths, then fine-tuned on flat sequences (full history → final article) to specialize toward the recommendation task, with a reduced learning rate of 5^{-6} and no warmup.

6 Experimental Results

6.1 Models Comparison

	Keywords			Question		
	All	Subseqs	Targets	All	Subseqs	Targets
Sparse retrieval						
BM25	0.205	0.187	0.262	0.249	0.249	0.275
Dense retrieval						
SPECTER2	0.147	0.124	0.212	0.203	0.205	0.221
E5-large-v2	0.202	0.202	0.221	0.259	0.275	0.250
Qwen3-Embedding-8b	0.242	0.243	0.262	0.311	0.332	0.300
Hybrid retrieval						
BM25 + SPECTER2	0.223	0.203	0.290	0.266	0.269	0.292
BM25 + E5-large-v2	0.231	0.222	0.269	0.282	0.291	0.288
BM25 + Qwen3-Embedding-8b	**0.258**	0.250	**0.301**	**0.322**	0.339	**0.319**
Trained Architectures						
SPECTER2 $_{1\ \text{Phase}}$	0.166	0.215	0.075	0.215	0.269	0.124
SPECTER2 $_{2\ \text{Phases}}$	0.172	0.216	0.094	0.232	0.282	0.152
E5-large-v2 $_{1\ \text{Phase}}$	0.193	0.251	0.085	0.234	0.306	0.107
E5-large-v2 $_{2\ \text{Phases}}$	0.196	0.248	0.105	0.251	0.314	0.145
Qwen3-Embedding-8b $_{1\ \text{Phase}}$	0.236	0.292	0.139	0.286	**0.356**	0.170
Qwen3-Embedding-8b $_{2\ \text{Phases}}$	0.254	**0.298**	0.189	0.309	0.354	0.248

Table 3: Performance of sparse, dense, hybrid and trained models across query types (Keywords, Question) and evaluation slices (All, Subsequence, Target). Best overall scores are **highlighted**. Performances of trained models beating baselines are underlined.

Table 6 reports R@10 on the test set under our sequential evaluation protocol, broken down into three slices: *all steps*, *subsequences only*, and *final target only*.

Overall, hybrid retrieval consistently outperforms both sparse and dense models in isolation, with BM25 + Qwen3-Embedding-8B achieving the best results across both query types (R@10=0.258 for keywords, R@10=0.322 for questions). This highlights that lexical and semantic signals are complementary for citation recommendation.

Among dense models, Qwen3-Embedding-8B dominates, outperforming both SPECTER2 and E5-large-v2 by a large margin despite the latter being domain-adapted to scientific text. This suggests that larger general-purpose embedding models transfer well to citation recommendation, likely due to their stronger semantic representations.

Training models yields mixed results. Sequential modeling often improves over baseline static embeddings, suggesting it captures additional signals beyond embedding similarity. Notably, gains (see underlined values in table 6) are concentrated on subsequence prediction, while target slice performance remains weak across all trained models. We attribute this primarily to data scarcity: final article targets are rarer than subsequences steps by construction, and long reading sequences are underrepresented in the dataset. This is further highlighted by our two-phase training strategy, where pre-training on subsequences before fine-tuning on full sequences generally yields better performances than only one phase. This suggests the model can learn sequential structure when given sufficient signal, but that data volume remains the primary bottleneck for target prediction.

6.2 History vs Cold Start

	Keywords			Question		
	All	Subseq.	Target	All	Subseq.	Target
Sparse						
BM25 (Q-Only)	0.208	0.194	0.242	0.245	0.252	0.251
+ History Δ	–0.003	–0.007 ***	+0.020 **	+0.004	–0.003	+0.024 ***
Dense						
SPECTER2 (Q-Only)	0.121	0.120	0.129	0.193	0.203	0.190
+ History Δ	+0.026 ***	+0.004 *	+0.083 ***	+0.011 ***	+0.002	+0.031 ***
E5-large-v2 (Q-Only)	0.191	0.198	0.187	0.257	0.275	0.244
+ History Δ	+0.011 ***	+0.004 **	+0.034 ***	+0.002	0.000	+0.006 *
Qwen3-Embedding-8b (Q-Only)	0.234	0.239	0.239	0.311	0.332	0.300
+ History Δ	+0.008 ***	+0.004 ***	+0.023 ***	-	-	-
Hybrid						
SPECTER2 (Q-Only)	0.219	0.211	0.247	0.259	0.270	0.261
+ History Δ	+0.004	–0.008 **	+0.043 ***	+0.006 ***	–0.001	+0.031 ***
E5-large-v2 (Q-Only)	0.225	0.224	0.237	0.279	0.294	0.273
+ History Δ	+0.006 *	–0.002	+0.032 ***	+0.003	–0.003	+0.015 ***
Qwen3-Embedding-8b (Q-Only)	0.251	0.247	0.277	0.318	0.337	0.311
+ History Δ	+0.007 ***	+0.003	+0.024 ***	+0.004 **	+0.001	+0.008 *
Mean Δ	+0.008	0.000	+0.037	+0.004	0.000	+0.016

Table 4: Base scores report query-only (cold-start) R@10; Δ rows show the change after incorporating history at the optimal α. Significance over cold-start is assessed with a paired Wilcoxon signed-rank test (*,p<0.05, **,p<0.01, ***,p<0.001).

As shown in Table 4, history provides a meaningful signal, but its benefit is concentrated on the target slice for baselines and hybrid approaches. Across all models and retrieval strategies, adding history yields consistent gains on Target (mean

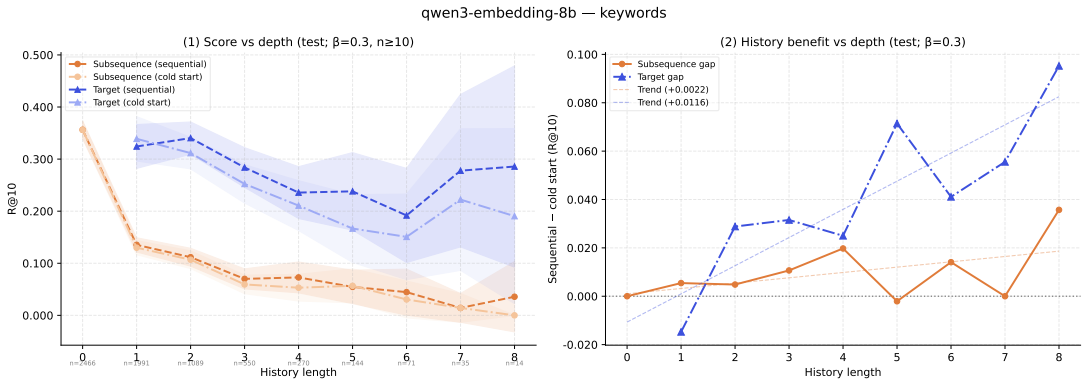


Figure 4: Retrieval performance by history length for BM25 + Qwen3-Embedding-8b on the Keywords query type.

Δ=+0.037 for Keywords, +0.016 for Question), while Subsequence gains are negligible. This asymmetry grows with history depth, as seen for example on hybrid BM25 + Qwen3-Embedding-8b (Figure 4, slope=+0.012 for Target vs. +0.002 for Subsequence), suggesting that accumulated citations are most useful for predicting the final article in a sequence.

On subsequences, several models show small negative deltas when history is added, confirming that intermediate co-citation groups lack strict sequential dependencies. This suggests that while history is a useful signal, naive averaging is insufficient: proper representation of the reading sequence is necessary to fully exploit it.

6.3 Sequentiality Ablation

	Keywords			Question		
	All	Subseq.	Target	All	Subseq.	Target
Q+Hist	0.172	0.216	0.094	0.232	0.282	0.152
Δ Q-Only	-0.003	+0.002	-0.016 **	-0.004	-0.003	-0.010
History Order						
Δ Shuffled	-0.007 ***	-0.008 ***	-0.002	-0.009 ***	-0.010 ***	-0.006 **
Δ Reversed	-0.014 ***	-0.019 ***	-0.001 **	-0.017 ***	-0.017 ***	-0.014 ***
Groups Order						
Δ Shuffled	0.000	0.000	-0.003	0.000	0.000	0.000
Δ Reversed	-0.001	0.000	-0.002	0.000	0.000	0.000

Table 5: Effect of history and citation-group ordering on BERT4Scholar with SPECTER2 (avg. over 5 seeds). Significance is assessed with a paired Wilcoxon signed-rank test (* p<0.05, ** p<0.01, *** p<0.001).

To assess whether history and groups ordering carries meaningful signal, we evaluate BERT4Scholar under three conditions: original sequence order, reversed, and randomly shuffled. Results are reported in Table 5.

Query-only retrieval slightly underperforms Q+Hist across most slices, though differences are generally small and not statistically significant, suggesting that the architecture may not effectively exploit history. In contrast, disrupting history order consistently degrades performance, with reversed order hurting more than shuffled order (e.g. Δ=-0.019 vs. -0.008 on Keywords Subseq.), indicating sensitivity to reading direction. Within-group order lack of impact suggests that co-citation groups, rather than individual article order, carry the relevant history signal.

7 Conclusion

This work introduces sequential recommendation for scientific literature as a novel task. We release a dataset of citation sequences drawn from ACL article introductions and an evaluation protocol tailored to their sequential structure. Each sequence is paired with queries generated from the original sequence to serve as a proxy for the initial information need. Baseline results reveal that history is most informative for predicting the final article in a sequence, while intermediate steps remain largely order-independent, suggesting the need for models that better represent reading progression. We propose BERT4Scholar, a recommendation architecture that adapts existing text encoders to the sequential structure of scientific reading, directly leveraging semantic representations of papers and queries. Experiments show that, although BERT4Scholar improves performance on subsequence slices, it still relies predominantly on query signals, highlighting the need for better integration of multiple signals.

We hope this work encourages further research in this direction, ultimately helping researchers navigate unfamiliar literature more effectively.

Limitations

Sequentiality of Articles. This work investigates the sequential nature of scientific reading behavior through a next-paper recommendation task. While reading history improves recommendation quality (particularly at the final step of a sequence) the contribution of ordering information remains modest. A key limitation is dataset scale and history sparsity, which likely constrains the model's ability to capture sequential patterns. Future work on richer behavioral signals and architectures designed for sparse sequential interaction data may better expose the role of sequentiality in scientific reading.

Scope of the Collection. Our dataset is derived from ACL Anthology, a high-quality but narrow corpus focused exclusively on NLP and computational linguistics. This limits both the diversity of research domains and the total number of documents available for retrieval. The relatively small collection size may inflate retrieval scores compared to more realistic large-scale settings, and restricts the generalizability of our findings to other scientific fields.

Automatic Citation Extraction. Reading sequences are constructed from citation graphs extracted using GROBID, which is not error-free. Extraction errors may introduce noise in the sequences, leading to incomplete or partially incorrect reading sequences. To mitigate this, we filter citations to ensure that referenced articles exist within the ACL Anthology, reducing the impact of spurious citations, but some noise inevitably remains.

Generated Information Needs. Our evaluation relies on automatically generated queries (keywords and questions) as proxies for user information needs, since ground-truth search intents are unavailable. These queries cannot fully capture the diversity of real user intent, and their quality is inherently tied to the generalization ability of the underlying language model over sequences of articles. Queries generated from long sequences may be less faithful to individual articles, as shown in our ordering analysis. Moreover, generated queries may already implicitly encode sequential context, making it difficult to disentangle the contribution of the query from that of the reading history in our evaluation.

Sequential Model Architecture. Our model is adapted from ID-based sequential recommendation architectures, which were not originally designed to operate over dense semantic representations. Freezing the item embeddings may prevent the model from learning task-specific representations that better capture the relationship between query and history signals. More dedicated architectures that jointly encode query context and reading history could better leverage the complementary signals identified in our retrieval analysis.

References

Amjad Abu-Jbara, Jefferson Ezra, and Dragomir Radev. 2013. Purpose and polarity of citation: Towards NLP-based bibliometrics. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 596–606, Atlanta, Georgia. Association for Computational Linguistics.

Julien Aubert-Béduchaud, Florian Boudin, Béatrice Daille, and Richard Dufour. 2025. ACL-rlg: A dataset for reading list generation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 4910–4919, Abu Dhabi, UAE. Association for Computational Linguistics.

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

Tesfaye Fenta Boka, Zhendong Niu, and Rama Bastola Neupane. 2024. A survey of sequential recommendation systems: Techniques, evaluation, and future directions. *Information Systems*, 125:102427.

Vera Boteva, Demian Gholipour, Artem Sokolov, and Stefan Riezler. 2016. A full-text learning to rank dataset for medical information retrieval. In *Advances in Information Retrieval*, pages 716–722, Cham. Springer International Publishing.

Florian Boudin. 2018. Unsupervised keyphrase extraction with multipartite graphs. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 667–672, New Orleans, Louisiana. Association for Computational Linguistics.

Jiayuan Ding, Tong Xiang, Zijing Ou, Wangyang Zuo, Ruihui Zhao, Chenhua Lin, Yefeng Zheng, and Bang Liu. 2022. Tell Me How to Survey: Literature Review Made Simple with Automatic Reading Path Generation. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 3426–3438.

Michael D. Ekstrand, Praveen Kannan, James A. Stemper, John T. Butler, Joseph A. Konstan, and John T.

Riedl. 2010. Automatically building research reading lists. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, page 159–166, New York, NY, USA. Association for Computing Machinery.

Pablo Figueira, Fabiano Belém, Jussara M. Almeida, and Marcos A. Gonçalves. 2019. Automatic Generation of Initial Reading Lists: Requirements and Solutions. In *Proceedings of the 18th Joint Conference on Digital Libraries*, JCDL '19, page 1–10. IEEE Press.

Markus Flicke, Glenn Angrabeit, Madhav Iyengar, Vitalii Protsenko, Illia Shakun, Jovan Cicvaric, Bora Kargi, Haoyu He, Lukas Schuler, Lewin Scholz, Kavyanjali Agnihotri, Yong Cao, and Andreas Geiger. 2025. Scholar inbox: Personalized paper recommendations for scientists. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 307–317, Vienna, Austria. Association for Computational Linguistics.

Jonathan Gordon, Stephen Aguilar, Emily Sheng, and Gully Burns. 2017. Structured generation of technical reading lists. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 261–270, Copenhagen, Denmark. Association for Computational Linguistics.

Danil Gusak, Anna Volodkevich, Anton Klenitskiy, Alexey Vasilev, and Evgeny Frolov. 2025. Time to split: Exploring data splitting strategies for offline evaluation of sequential recommenders. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, RecSys '25, page 874–883, New York, NY, USA. Association for Computing Machinery.

Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206.

Christin Katharina Kreutz and Ralf Schenkel. 2022. Scientific paper recommendation systems: a literature review of recent publications. *International journal on digital libraries*, 23(4):335–369.

Vincent Nguyen, Maciej Rybinski, Sarvnaz Karimi, and Zhenchang Xing. 2022. Search like an expert: Reducing expertise disparity using a hybrid neural index for covid-19 queries. *Journal of biomedical informatics*, 127:104005.

Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019. From doc2query to doctttttquery.

Aravind Sesagiri Raamkumar, Schubert Foo, and Natalie Pang. 2017. Using author-specified keywords in building an initial reading list of research papers in scientific paper retrieval and recommender systems. *Information Processing & Management*, 53(3):577–594.

Amanpreet Singh, Mike D'Arcy, Arman Cohan, Doug Downey, and Sergey Feldman. 2023. SciRepEval: A multi-format benchmark for scientific document representations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5548–5566, Singapore. Association for Computational Linguistics.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, pages 1441–1450, New York, NY, USA. ACM.

John M. Swales. 1990. *Genre analysis : English in academic and research settings / John M. Swales.*, pages 136–166. The Cambridge applied linguistics series. Cambridge University Press, Cambridge.

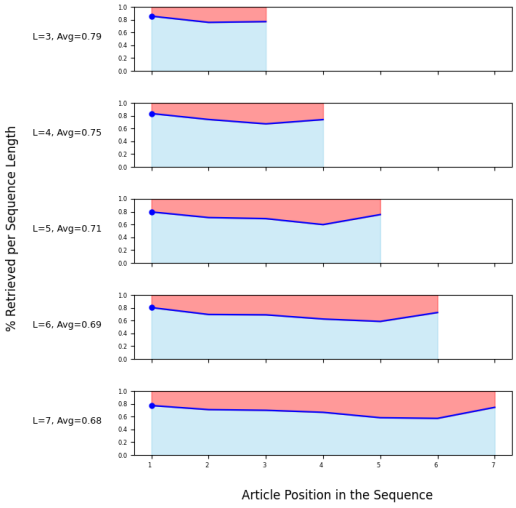
Simone Teufel, Advaith Siddharthan, and Dan Tidhar. 2006. Automatic classification of citation function. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 103–110, Sydney, Australia. Association for Computational Linguistics.

Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2021. Trec-covid: constructing a pandemic information retrieval test collection. 54(1).

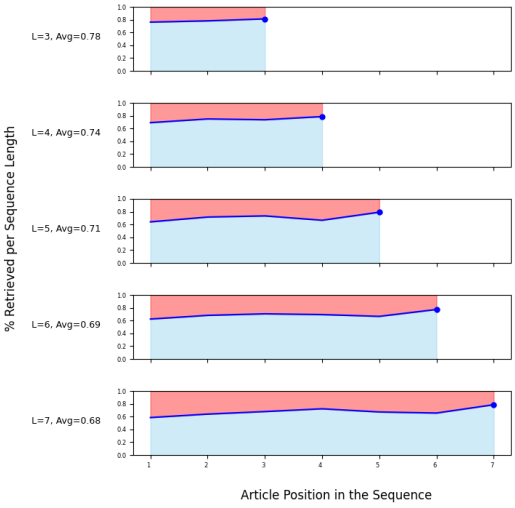
A Query Generation Prompting

```
INSTRUCTIONS = {
    "keywords": (
        "You are given a set of scientific papers (titles and
           abstracts).\n"
        "Generate **up to 3 single-word keywords** that best capture
           the **shared research topic** of these papers.\n"
        "- Each keyword must be a valid English word. Do NOT merge
           multiple words in a single one.\n"
        "- Output **up to 3 words**, separated by commas.\n"
        "- Choose **domain-specific, non-generic words** suitable for
            retrieving similar papers.\n"
        "- Do **not** include phrases, full titles, or stopwords.\n"
    ),
    "question": (
        "You are given a set of scientific papers (titles and
           abstracts).\n"
        "Generate a **single search question query of exactly 10
           words** that best represents the shared research topic of
           these papers.\n"
        "- The query should be **natural, readable, and suitable for
           retrieving similar papers.**\n"
        "- Output exactly the 10-word question, nothing else.\n"
        "- Question should be **domain-specific, non-generic**.\n"
        "- Do **not** include phrases, full titles, or stopwords.\n"
    ),
}
```

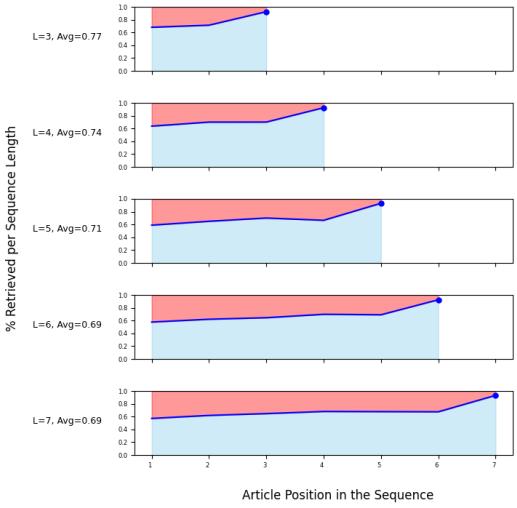
B Sequence Order Effect



(a) Base Ordering



(b) Random Ordering



(c) Reversed Ordering

Figure 5: Effect of sequence order on query generation.

C BERT4Scholar Detailed Architecture

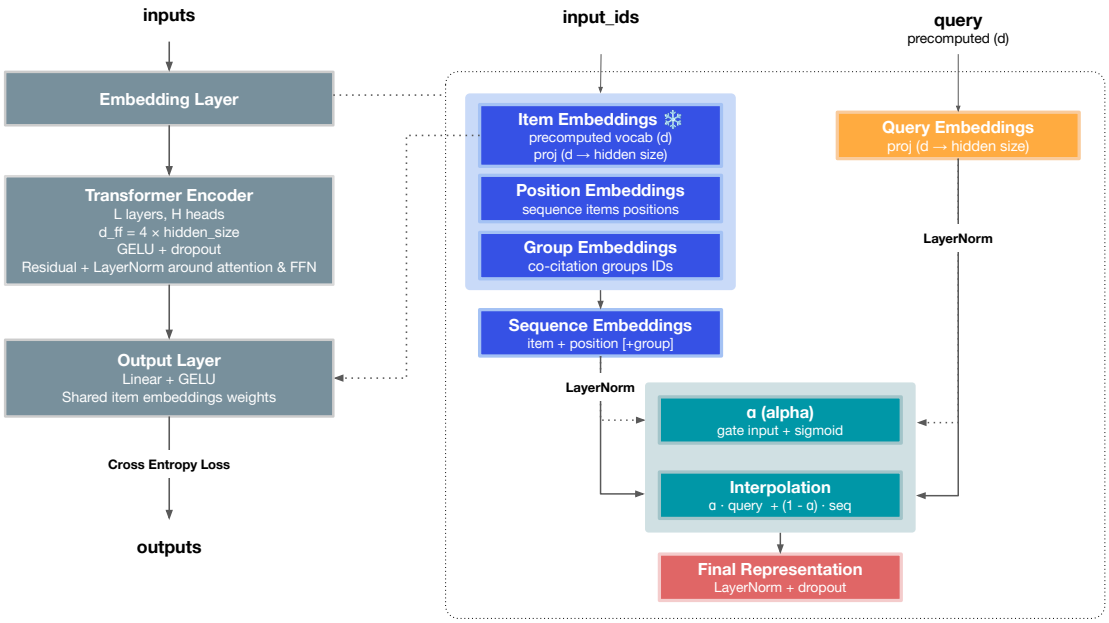
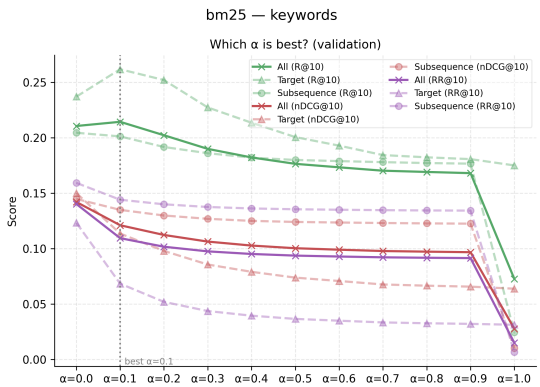


Figure 6: BERT4Scholar architecture overview (left) and embedding layer visualization (right)

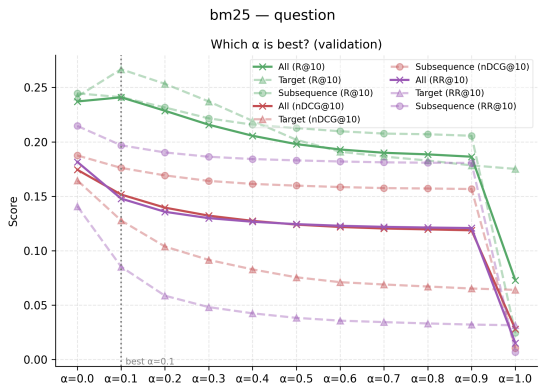
D Optimal α/β Selection for Baselines and Hybrid models

Value	Keywords	Question
Sparse		
BM25	α=0.1	α=0.1
Dense		
SPECTER2	α=0.6	α=0.3
E5-large-v2	α=0.3	α=0.1
Qwen3-Embedding-8b	α=0.2	α=0.0
Hybrid		
BM25 + SPECTER2	β=0.7	β=0.5
BM25 + E5-large-v2	β=0.4	β=0.4
BM25 + Qwen3-Embedding-8b	β=0.3	β=0.2

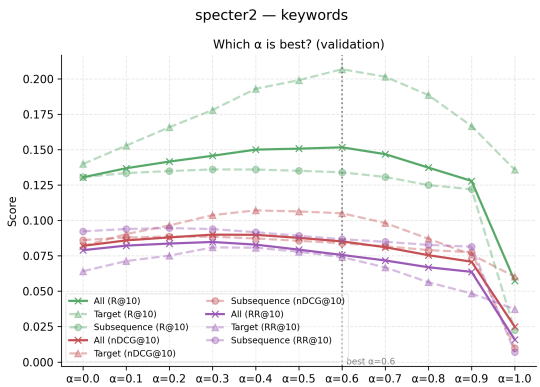
Table 6: Optimal parameters for each baseline and hybrid model. For hybrid methods, the corresponding optimal α values from the baseline models are used.



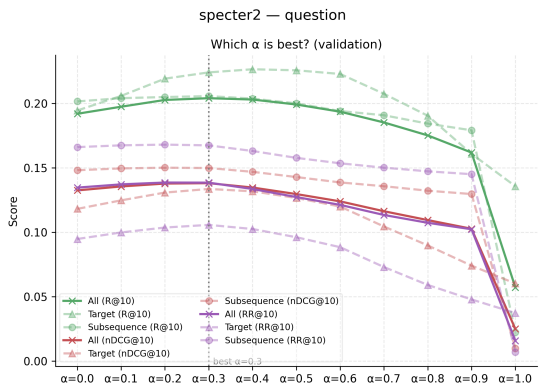
(a) BM25 - Keywords



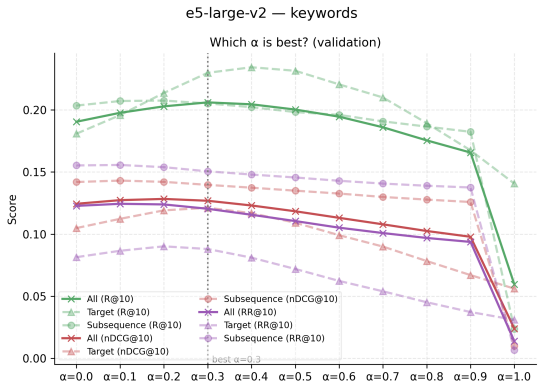
(b) BM25 - Question



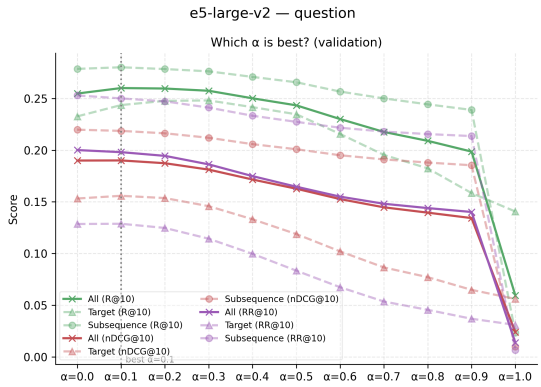
(c) SPECTER2 - Keywords



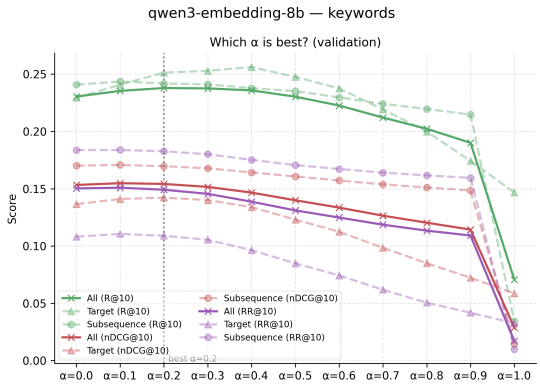
(d) SPECTER2 - Question



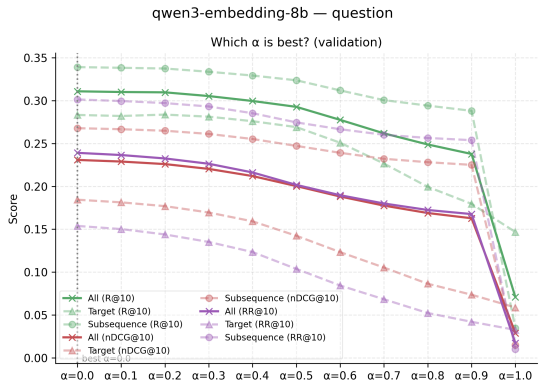
(e) E5-large-v2 - Keywords



(f) E5-large-v2 - Question

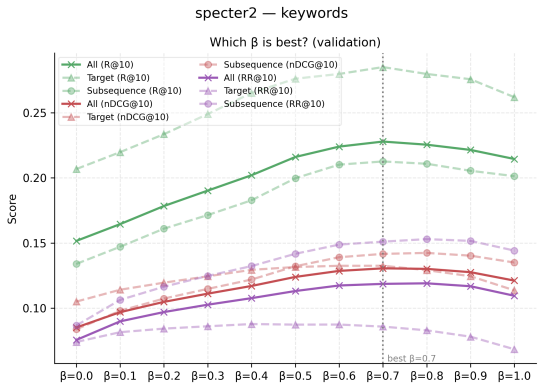


(g) Qwen3-Embedding-8b - Keywords

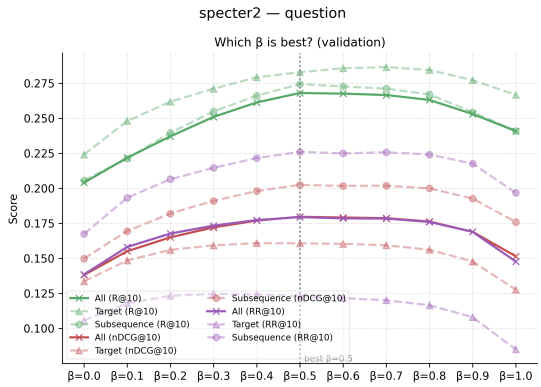


(h) Qwen3-Embedding-8b - Question

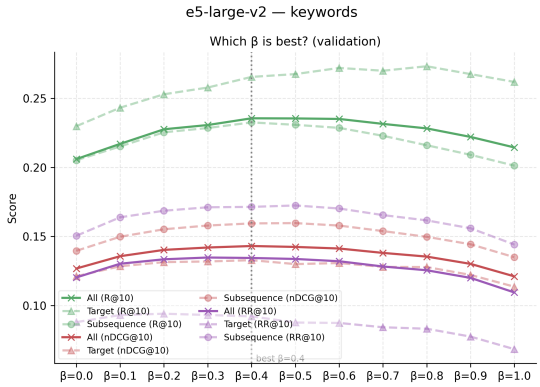
Figure 7: Optimal α ratio computed on the validation set. $\alpha = 0.0$ corresponds to query-only, while $\alpha = 1.0$ corresponds to history-only.



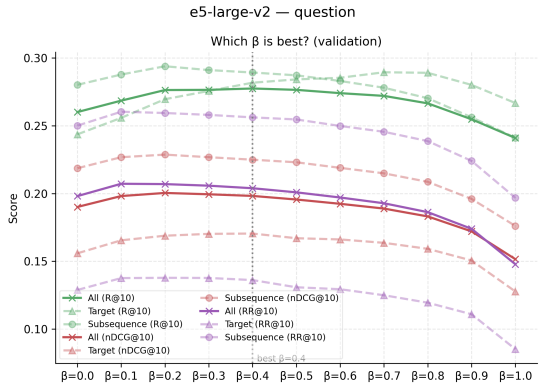
(a) BM25 + SPECTER2 - Keywords



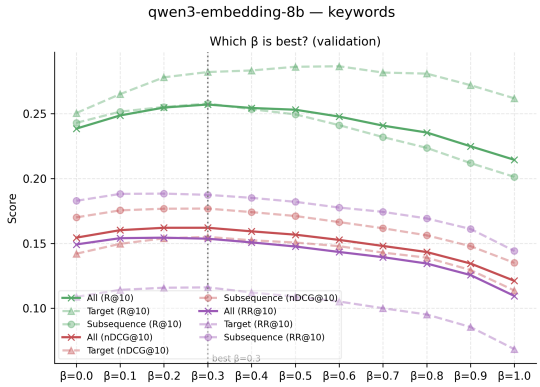
(b) BM25 + SPECTER2 - Question



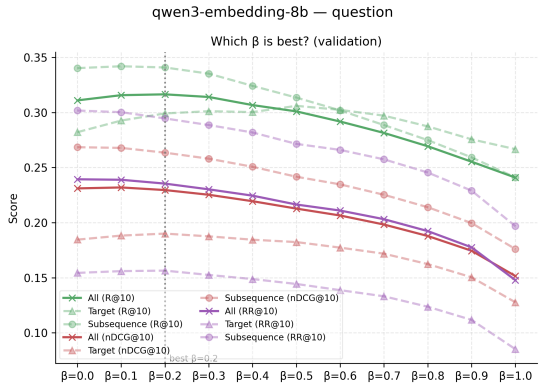
(c) BM25 + E5-large-v2 - Keywords



(d) BM25 + E5-large-v2 - Question



(e) BM25 + Qwen3-Embedding-8b - Keywords



(f) BM25 + Qwen3-Embedding-8b - Question

Figure 8: Optimal β ratio computed on the validation set. $\beta = 0.0$ corresponds to Dense-only, while $\beta = 1.0$ corresponds to BM25-only.