Hybrid Evidence-Aware User Simulation for Scientific Search Sessions at LongEval 2026 Task 3

LongEval-USim at CLEF 2026

Santiago Andres Orejuela Cueter[1], Jairo Enrique Serrano Castañeda[1], Juan Carlos Martinez-Santos[1] and Edwin Puertas[1]

[1]Universidad Tecnologica de Bolivar, Cartagena de Indias, Colombia

Abstract

This paper describes a hybrid approach to LongEval-USim, Task 3 of the LongEval Lab at CLEF 2026. The task asks participants to simulate the next query of a scientific search user: given a search session in which the real final query is hidden, a system must submit five ranked candidate queries. Our approach treats next-query prediction as an evidence-aware candidate generation and selection problem. It combines four complementary signals: the visible query history, scientific documents shown in the search engine results page, clicked documents as weak relevance feedback, and a transition memory learned from training sessions. Candidate queries are generated through deterministic reformulation heuristics, nearest-neighbor retrieval over historical query transitions, and local instruction-model reformulation. The final ranked list is selected with a relevance–diversity objective inspired by maximal marginal relevance, with additional filters for lexical support, naturalness, and redundancy. We discuss four organizer-evaluated run identifiers: an initial no-LLM baseline, two hybrid evidence-aware variants, and a portfolio selector. Organizer-provided preliminary results show that the initial baseline was the most robust of our runs across snapshots, while the hybrid and portfolio variants obtained small gaps between their top-ranked and best available candidates, indicating effective candidate selection.

Keywords

User simulation, Next query prediction, Scientific search, Query reformulation, LongEval, Maximal marginal relevance

1. Introduction

User simulation for information access aims to reproduce plausible user actions without requiring a live user study for every system variant [1]. In search, one important instance of this problem is next-query prediction. Given the actions taken during a search session, a simulator must infer how the user would reformulate the information needed [2, 3]. This setting is especially demanding in scientific search, where query reformulations may be short, technical, and strongly influenced by document titles, abstracts, and clicked results.

LongEval-USim, Task 3 of the LongEval Lab at CLEF 2026, evaluates this problem under temporal change [4, 5]. The released sessions are grouped into snapshots that correspond to different time periods. For each target session, the last query is hidden, and the system must submit five candidate queries ordered from most to least plausible. The evaluation measure combines rank-sensitive semantic matching with diversity among the submitted candidates [3]. Therefore, a strong system should not only generate a near paraphrase of the true final query; it should also provide a compact set of plausible alternatives that are not merely duplicates of each other.

Our system was designed around that tension. We observed that a purely retrieval-based memory can produce realistic queries. Still, we may overfit to training sessions, while a language model can generate natural text but may drift away from the actual session evidence. We therefore used a hybrid architecture in which each component contributes candidates. Still, the final decision is made by a

CLEF 2026 Working Notes, 21 – 24 September 2026, Jena, Germany

sorejuela@utb.edu.co (S. A. O. Cueter); jserrano@utb.edu.co (J. E. S. Castañeda); jcmartinezs@utb.edu.co (J. C. Martinez-Santos); epuerta@utb.edu.co (E. Puertas)

0009-0003-3959-407X (S. A. O. Cueter); 0000-0001-8165-7343 (J. E. S. Castañeda); 0000-0003-2755-0718 (J. C. Martinez-Santos); 0000-0002-0758-1851 (E. Puertas)

controlled ranker that rewards relevance to the session state and penalizes redundancy [6, 7]. The system also includes an explicit portfolio variant that combines several internal candidate sources while preserving a strict lexical diversity guard.

The runs described in this paper were submitted under the team name VerbaNexAI. This working note documents the submitted method, the validation steps used to ensure that the run files were well-formed and consistent with the task requirements, and preliminary organizer-provided results for our own runs. Comparative results against other participants are left to the task overview papers.

2. Task and Data

The task input consists of search sessions from a scientific search setting. Each row in a session file represents one query step and includes a user identifier, a session number, the query string, a query timestamp, a list of document identifiers shown in the SERP, a search identifier, and click information. The click field contains document identifiers, timestamps, and click types when available. The auxiliary document collections contain scientific metadata, including at least titles and abstracts, indexed by document identifier [6].

Table 1 summarizes the session files used by the system. The training sessions were used for development and for constructing a transition memory. The two target snapshots were used for the final submissions. The number of rows exceeds the number of sessions because each session contains multiple query steps.

Table 1
Session data used by the system. Session lengths are measured in the number of visible query rows per session in the released files.

Split	Period	Rows	Sessions	Median length
Training	training snapshot	575	180	3
Target	June–August 2025	277	119	2
Target	September–November 2025	387	182	2

The auxiliary scientific collections are much larger than the number of documents actually touched by the sessions. For efficiency, the system builds a sparse document index: it first collects the document identifiers appearing in SERPs and clicks, then scans the compressed JSONL document files and stores only the title, abstract, and a concatenated text field formed as title plus abstract for those identifiers. It avoids loading the full collection into memory while still making document evidence available for session representation and candidate generation.

The task's expected output is a JSON file for each snapshot. Each session key maps to exactly five candidate queries. In our submissions, every run file contained three snapshot JSON files with 180, 119, and 182 sessions, respectively, and all sessions were checked to have five non-empty, unique candidates.

3. System Overview

The submitted system follows the pipeline shown in Listing 1. The main design choice is to separate candidate generation from candidate selection. Generation is intentionally broad: it gathers lexical reformulations, document-derived expansions, historical next query examples, and language-model rewrites. Selection is intentionally conservative: it reduces the candidate pool to five ranked queries using semantic relevance, document support, and diversity constraints [8, 9, 7].

Listing 1: High-level pipeline used for each session.

```
1. Parse SERP and click fields into Python lists.
2. Build embeddings for queries and available document texts.
3. Compute a recency-weighted session state vector.
```

```
4. Generate a candidate pool from:
   a) query reformulation heuristics,
   b) nearest neighbors in transition memory,
   c) local instruction-model reformulation,
   d) role-guided rewrites of strong seed candidates.
5. Normalize and deduplicate candidates.
6. Score each candidate by state relevance, last-query relevance,
   lexical support, naturalness, and source reliability.
7. Select five candidates with an MMR-style relevance-diversity
   ranker.
8. Validate output format and write the TIRA snapshot JSON files.
```

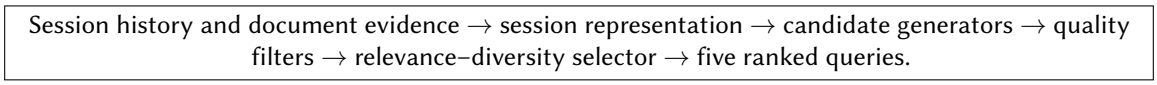

Figure 1: High-level workflow of the submitted systems.

The four organizer-evaluated configurations instantiate this same pipeline with different generation and selection policies. The run identified by the organizers as `VerbaNexAI_v1` corresponds to the first uploaded system: a no-LLM baseline optimized on the training sessions. It uses heuristic candidates from the last visible query and document evidence, then applies an MMR-style selector with strict lexical-diversity constraints. A second configuration uses the hybrid generator with transition memory, a local instruction model, and a relevance–diversity ranker with moderate role-guided reformulation. A third configuration is more quality-aware, placing stronger penalties on weak or unnatural candidates. The fourth configuration is a portfolio selector: it receives several internally generated candidate lists for the same session. It combines them with source reliability weights, rank decay, and a strict diversity filter. It allowed us to submit both direct hybrid systems and a more conservative ensemble without relying on official target labels.

4. Session Representation

The system converts each session step into a small set of numerical and lexical features. Query strings are embedded directly. For document evidence, the system maps SERP and clicks document identifiers to text using the sparse document index. Each document's text is represented as the concatenation of its title and abstract. We use a compact sentence embedding model, `all-MiniLM-L6-v2`, through the SentenceTransformers library [10, 11]. The choice was motivated by speed and by the need to embed many short queries and document snippets on a local machine.

For a single query step, three vectors are computed: the query embedding, the mean embedding of the top SERP documents available in the index, and the mean embedding of clicked documents. Clicked document embeddings are weighted by click type and by the document's position in the SERP when that position is known. It gives slightly more weight to documents the user engaged with more strongly and to those that were visible near the top of the result list [8, 9].

The full session state is a recency-weighted average of the step vectors. Let q_t be the query embedding at step t, c_t the clicked document vector, and s_t the SERP-document vector. A step representation is computed as

$$h_t = \mathrm{norm}(w_q q_t + w_c c_t + w_s s_t),$$

where missing document components are omitted, the session representation is then:

$$H = \mathrm{norm}\left(\sum_{t=1}^{T} \gamma^{T-t} h_t\right),$$

with $\gamma < 1$ giving higher weight to recent interactions. This state is used both for nearest-neighbor retrieval in the transition memory and for ranking candidate next queries.

In parallel, the system keeps lexical term sets extracted from the visible queries, clicked documents, and SERP documents. These terms are later used to assess whether a candidate is grounded in the session context. This lexical support feature is useful because semantic embeddings can sometimes assign high similarity to fluent but underspecified candidates, whereas the task rewards queries that plausibly arise from the concrete session.

5. Candidate Generation

5.1. Heuristic Reformulations

The first candidate source is deterministic. It starts from the last visible query, because query reformulation in search sessions is often incremental [2]. The system keeps the original last query, removes stopwords, truncates overly long phrases to roughly 8–10 content terms depending on the generator, and combines the cleaned query with salient terms extracted from clicked abstracts and high-ranked SERP abstracts. The threshold was chosen during hidden-last-query development to avoid sentence-like candidates while preserving enough technical context. The generator also creates small recombinations of document keywords. These candidates are not expected to be perfect, but they provide robust fallbacks and often capture the stable core of the information needed.

The heuristic generator is deliberately conservative. Normalization includes trimming whitespace, collapsing repeated spaces, removing surrounding punctuation or quotation marks, rejecting empty strings, and using a lowercase copy only for deduplication and lexical matching. The displayed query text keeps its original casing when possible. The generator also avoids very short candidates and limits the maximum length of each query. It matters for the final diversity objective: a broad but noisy pool can lower quality if the ranker is forced to fill all five positions from poor alternatives.

5.2. Transition Memory

The second source is a transition memory learned from the training sessions. For every training session with at least two query steps, the system hides the last query, computes the state representation of the visible prefix, and stores the pair:

$$\text{visible session state} \rightarrow \text{observed next query}.$$

At inference time, the current session state is compared with all stored training states using cosine similarity. The observed next queries from the closest neighbors are directly added to the candidate pool [7, 12]. We did not use a single hard similarity threshold at retrieval time. Instead, retrieval is intentionally permissive and relevance is enforced downstream by candidate scoring: retrieved queries must compete with heuristic and model-generated candidates under state relevance, last-query relevance, lexical support, naturalness, and diversity constraints. In local training-prefix evaluation, self-neighbors are excluded so that the memory cannot recover the hidden target query from the same session.

This memory is attractive because it retrieves real user queries rather than synthetic text. It also captures transitions that are hard to encode manually, such as narrowing from a broad topic to a named method, adding an application domain, or switching terminology after inspecting results. In this sense, *retrieval-based memory* means that the system stores examples of previous session-state to next-query transitions and reuses the next-query side of similar examples as candidates for a new session.

5.3. Local Instruction-Model Reformulation

The third source uses a locally hosted instruction-tuned language model served through a local inference application [13]. In the final local experiments, the model was Qwen 3.5 9B served locally; it was not

fine-tuned by the authors. The model is prompted with the visible query history, compact document evidence, clicked-document titles when available, and a set of seed candidates. These seeds are selected from the strongest heuristic and transition-memory candidates before final ranking. The expected output is a JSON object containing a small list of candidate queries. The prompt asks for concise scientific search queries rather than explanatory sentences.

The language-model component is used as a controlled reformulator, not as an unrestricted generator. It is allowed to produce more natural alternatives and role-guided rewrites. Still, every generated query is passed through the same normalization, deduplication, support, naturalness, and diversity checks as the heuristic and memory candidates. In practice, this made the local model useful for improving query fluency while preventing it from dominating the ranked output when its suggestions were unsupported by the session evidence.

Role-guided refinement is applied only to strong seed candidates. A seed is considered strong when it has high preliminary relevance to the session state or last query, passes the naturalness and lexical support filters, and is not already a near duplicate of a stronger candidate. Instead of asking the model to invent the next query from scratch, the prompt asks it to rewrite or specialize candidates from complementary perspectives, such as preserving the user's original terminology, narrowing to a specific method or dataset, broadening to a related research area, or adding terminology from clicked documents. This design was chosen to increase naturalness without losing grounding [1, 13].

6. Candidate Selection

Each candidate is embedded and compared with two references: the full session state and the last visible query. The first comparison rewards candidates that match the whole interaction context; the second protects against excessive topic drift. We also compute lexical support against the query and document term sets, a naturalness score based on length and token patterns, and a source bonus that reflects whether a candidate came from memory, heuristics, local model reformulation, or multiple sources. Lexical support is computed from content-word overlap with terms extracted from the visible queries, clicked documents, and high-ranked SERP documents. Candidates with no support from the session context are penalized or filtered unless they are strongly supported by semantic relevance.

The final five candidates are selected using a maximal-marginal-relevance style procedure [14]. The weights and thresholds were selected with the hidden-last-query protocol on the training sessions, using coarse manual sweeps over the local RDS proxy and qualitative inspection of failure cases. The initial no-LLM baseline used $\lambda = 0.35$, a maximum token Jaccard threshold of 0.30, a pool of 80 candidates, and a minimum content-term requirement. The later submitted configurations used different relevance–diversity balances and Jaccard thresholds, with stricter settings in the quality-aware and portfolio variants. At each step, the ranker considers the remaining candidates that pass quality gates and chooses the candidate with the best balance between relevance and novelty:

$$\text{score}(x) = \lambda \,\text{rel}(x, H) - (1 - \lambda) \max_{y \in S} \text{red}(x, y),$$

where H is the session state and S is the set of already selected candidates. Redundancy combines embedding-level and token-level Jaccard similarities. A hard Jaccard threshold prevents near-duplicate candidates from occupying several positions in the final list. If strict filtering leaves fewer than 5 candidates, the system relaxes the diversity threshold and fills the remaining slots with the best-supported alternatives [3].

The portfolio configuration uses a related but simpler selector. It assigns each internal candidate list a source reliability weight, discounts lower ranks inside each list, aggregates candidate scores across sources, keeps the first candidate from the strongest source when available, and then applies a tight Jaccard diversity guard. Source reliability weights were chosen from local validation and error analysis: the strongest hybrid candidate lists received weight 1.0, while auxiliary memory-only and baseline lists received lower weights. This portfolio strategy is useful when several candidate-generation policies

produce complementary outputs: repeated candidates gain confidence, while overly similar alternatives are filtered.

7. Development Protocol

The training sessions provide the only released labels, because their final queries are visible. For development, we used a hidden-last-query protocol: for each training session, the final query was treated as the target, and the previous steps were treated as the visible session. This setup mirrors the target task and supports error analysis of candidate generation and ranking. The local metric used for model selection followed the Rank-Diversity Score idea introduced in Sim4IA-Bench [3]: a candidate receives reciprocal-rank credit when its normalized cosine similarity to the true next query reaches a semantic threshold, and the aggregate score is multiplied by one minus the mean pairwise Jaccard redundancy among the submitted candidates. The local metric was used only as a development proxy.

We additionally ran a temporal robustness check without using hidden labels from the target snapshots. The check compared candidate-source distributions, candidate lengths, and redundancy patterns across the training and target periods. Large changes in these diagnostics would suggest temporal drift or a malfunctioning generator. This sanity check is weaker than the official evaluation, but it is useful for detecting obvious submission problems before upload.

For the hybrid diagnostic runs, selected candidates were mostly produced by deterministic heuristics (73.6%), followed by transition memory (21.1%), local instruction-model generation (4.9%), and role-guided refinement (0.3%). This distribution suggests that the model-based component acted mainly as a controlled reformulator, while the final output remained grounded in deterministic and historical user-transition evidence.

8. Submitted Run Files

The organizer-provided results include four VerbaNexAI run identifiers. Table 2 includes the identifiers used in the results, the corresponding local artifacts, and their methodological descriptions. Each submitted archive contains `snapshot-1.json`, `snapshot-2.json`, and `snapshot-3.json`.

Table 2
Submitted system configurations.

Run identifier	Local artifact	Description
`VerbaNexAI_v1`	`task3_tira_submission.zip`	Initial no-LLM baseline optimized on training sessions; heuristic/document candidates selected with MMR.
`VerbaNexAI_v8`	`task3_tira_submission_v8.zip`	Hybrid evidence-aware system with transition memory, local reformulation, and relevance–diversity ranking.
`VerbaNexAI_v8b`	`task3_tira_submission_v8b.zip`	Quality-aware hybrid variant with stronger naturalness, lexical support, and candidate-quality penalties.
`VerbaNexAI_v13`	`task3_tira_submission_v13.zip`	Portfolio selector over complementary candidate lists with source reliability, rank decay, and strict diversity filtering.

All submitted run files follow the same TIRA format: one JSON file per snapshot, a metadata block with team and run description, and one list of five candidate queries for each session. The validation script checked the number of sessions per snapshot, the presence of exactly 5 candidates per session, the absence of empty strings, and the absence of duplicate candidates within a session. This format validation was performed independently of the official evaluator.

9. Preliminary Results

Table 3 reports the preliminary RDS scores provided by the organizers for our runs only. Relative change is computed from snapshot 1 to later snapshots, so lower values indicate more stable behavior over time. The run identified as v1 obtained the best RDS on snapshots 2 and 3 and the lowest relative change. This result suggests that the initial no-LLM baseline produced particularly stable rankings under temporal change. Among the later hybrid configurations, the portfolio selector produced the strongest RDS on snapshots 2 and 3 and the lowest relative change from snapshot 1.

Table 3
Preliminary organizer-provided RDS results for the submitted VerbaNexAI runs.

Run	S1	S2	S3	RC 1–2	RC 1–3
v1	0.4734	0.4583	0.2811	0.032	0.406
v8	0.4783	0.4334	0.2524	0.094	0.472
v8b	0.4915	0.4332	0.2522	0.119	0.487
v13	0.4778	0.4380	0.2569	0.083	0.462

The organizers also provided cosine-similarity and SERP-overlap results. For these two files, the labels *Best* and *Top1* were later corrected by the organizers; here, Top1 refers to the first-ranked candidate submitted by the system, while Best refers to the highest-scoring candidate among the five. Table 4 summarizes the mean Best–Top1 gap across the three snapshots. Small gaps mean that the candidate with the highest metric value was usually already placed near the top of the submitted ranking. The hybrid and portfolio runs show especially small gaps, supporting the observation that their candidate selection mechanisms ranked strong candidates effectively.

Table 4
Mean Best–Top1 gap across snapshots after applying the organizer-provided label correction for cosine similarity and SERP overlap.

Run	Cosine gap	SERP-overlap gap
v1	0.0467	0.0515
v8	0.0113	0.0063
v8b	0.0158	0.0086
v13	0.0121	0.0043

10. Discussion

The most important lesson from development and preliminary evaluation was that naturalness and evaluation alignment are not the same objective [1, 3]. A local language model can produce queries that read well, but some are too broad, overly explanatory, or not sufficiently tied to the visible search session. Conversely, transition-memory candidates are grounded in real user behavior, but they may repeat training patterns that are only partially related to the target session. The ranking stage, therefore, became the central component of the system: it has to accept helpful natural rewrites while rejecting unsupported drift.

The strong performance of the initial no-LLM baseline was informative. It achieved the most stable RDS values across snapshots without relying on local instruction-model reformulation. We interpret this as evidence that, in this task, grounding and diversity control are at least as important as generation breadth. This configuration kept close to the visible query and document evidence, avoided unsupported reformulations, and selected candidates with a balanced relevance–diversity criterion. This made it less exposed to temporal vocabulary drift than broader candidate pools.

The hybrid and portfolio runs behaved as expected in a different way. Their average scores over all five candidates were affected by the inclusion of more diverse alternatives, but their small Best–Top1 gaps in cosine similarity and SERP overlap suggest that the selector usually placed a strong candidate at

the top. The portfolio run also improved temporal robustness among the hybrid family, which supports the use of source reliability and cross-source agreement when multiple generators are available.

Document evidence was also useful, but had to be handled carefully. Clicked documents provide a strong signal of user interest, yet clicks are sparse and not always available. SERP documents provide a broader context, but using too many result documents can dilute the session state with irrelevant terms. The submitted system, therefore, uses a small number of SERP documents and gives clicked documents greater weight when present [8, 9].

The diversity requirement encouraged a design different from ordinary next-query prediction. If the system submitted five minor variations of the same query, semantic similarity might remain high, but redundancy would hurt the Rank Diversity Score. The MMR-style selection step was introduced specifically to avoid this failure mode. At the same time, the diversity threshold cannot be too loose, because diverse candidates who differ only by topic are not useful simulations of the same user. The final configurations, therefore, use both semantic relevance and token-level redundancy checks.

11. Limitations and Future Work

The current system is intentionally lightweight and mostly unsupervised. It does not train a neural reranker, does not fine-tune the local language model, and uses abstracts rather than full paper text in the submitted stable pipeline. These choices made the approach reproducible on a local machine and reduced the risk of long-running inference during submission preparation, but they also limit the depth of document understanding.

Future work will explore four directions. First, full-text evidence could be summarized into compact, session-specific profiles before candidate generation, rather than adding long documents directly to the prompt. Second, the transition memory could be complemented by a supervised reranker trained on hidden-last-query examples from the training sessions. Third, the role-guided local model could be calibrated with stricter output schemas and better rejection rules. Fourth, candidate-source diagnostics could be expanded to estimate which generators contribute most to robust behavior across snapshots.

12. Conclusion

We presented a hybrid approach to LongEval-USim at CLEF 2026. The system combines session embeddings, document evidence, transition-memory retrieval, heuristic reformulation, and controlled local model rewriting. Candidate selection is performed by a relevance–diversity ranker that attempts to produce five plausible but non-redundant next-query predictions. Preliminary results show that the initial no-LLM baseline was particularly robust across snapshots, while the hybrid and portfolio systems produced small Best–Top1 gaps, suggesting effective ranking of candidate queries. The main idea is that robust user simulation for scientific search benefits from combining real historical transitions, local document grounding, and conservative diversity-aware ranking.

Acknowledgments

The authors thank the LongEval organizers for providing the Task 3 data, submission infrastructure, and guidance on preparing the working notes before the release of official results.

Declaration on Generative AI

During the preparation of this work, OpenAI ChatGPT/Codex was used to help organize the paper structure, draft and edit English text, and prepare the LaTeX source. The authors reviewed and edited the content and take full responsibility for its publication. The submitted Task 3 systems also used a

locally hosted instruction-tuned language model as one of several candidate-generation components, as described in the method sections.

References

[1] K. Balog, C. Zhai, User simulation for evaluating information access systems, Foundations and Trends in Information Retrieval 18 (2024) 1–261. doi:10.1561/1500000098.

[2] B. J. Jansen, D. L. Booth, A. Spink, Patterns of query reformulation during web searching, Journal of the American Society for Information Science and Technology 60 (2009) 1358–1371. doi:10.1002/asi.21071.

[3] A. K. Kruff, C. K. Kreutz, T. Breuer, P. Schaer, K. Balog, Sim4IA-Bench: A user simulation benchmark suite for next query and utterance prediction, in: Advances in Information Retrieval: 48th European Conference on Information Retrieval, ECIR 2026, Lucca, Italy, Proceedings, Part IV, Springer, 2026, pp. 594–609. doi:10.1007/978-3-032-21321-1_61.

[4] S. Bouaraba, T. Breuer, M. Cancellieri, A. El-Ebshihy, M. Fröbe, P. Galuscáková, L. Goeuriot, G. Iturra-Bocaz, J. Keller, P. Knoth, A. K. Kruff, P. Mulhem, F. Piroi, D. Pride, P. Schaer, D. Schwab, Overview of the clef 2026 longeval lab on longitudinal evaluation of model performance, in: M. Hagen, M. Potthast, B. Stein, P. Schaer, E. Zangerle, S. MacAvaney, J. M. Struß, E. Sánchez Salido, A. Barrón-Cedeño, A. García Seco de Herrera (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Seventeenth International Conference of the CLEF Association (CLEF 2026), Lecture Notes in Computer Science (LNCS), Springer, 2026.

[5] S. Bouaraba, T. Breuer, M. Cancellieri, A. El-Ebshihy, M. Fröbe, P. Galuscáková, L. Goeuriot, G. Iturra-Bocaz, J. Keller, P. Knoth, A. K. Kruff, P. Mulhem, F. Piroi, D. Pride, P. Schaer, D. Schwab, Extended overview of the clef 2026 longeval lab on longitudinal evaluation of model performance, in: E. Sánchez Salido, A. Barrón-Cedeño, A. García Seco de Herrera, S. MacAvaney, J. M. Struß (Eds.), CLEF 2026 Working Notes, CEUR Workshop Proceedings, CEUR-WS.org, 2026.

[6] B. Carterette, E. Kanoulas, M. M. Hall, P. D. Clough, Overview of the TREC 2014 session track, in: Proceedings of The Twenty-Third Text REtrieval Conference, TREC 2014, National Institute of Standards and Technology, 2014. URL: https://trec.nist.gov/pubs/trec23/papers/overview-session.pdf.

[7] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, H. Li, Context-aware query suggestion by mining click-through and session data, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008, pp. 875–883. doi:10.1145/1401890.1401995.

[8] J. J. Rocchio, Relevance feedback in information retrieval, in: G. Salton (Ed.), The SMART Retrieval System: Experiments in Automatic Document Processing, Prentice-Hall, 1971, pp. 313–323.

[9] V. Lavrenko, W. B. Croft, Relevance-based language models, in: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2001, pp. 120–127. doi:10.1145/383952.383972.

[10] N. Reimers, I. Gurevych, Sentence-BERT: Sentence embeddings using siamese BERT-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, 2019, pp. 3982–3992. doi:10.18653/v1/D19-1410.

[11] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, M. Zhou, MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, in: Advances in Neural Information Processing Systems, volume 33, 2020, pp. 5776–5788.

[12] A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. G. Simonsen, J.-Y. Nie, A hierarchical recurrent encoder-decoder for generative context-aware query suggestion, in: Proceedings of the 24th ACM International Conference on Information and Knowledge Management, 2015, pp. 553–562. doi:10.1145/2806416.2806493.

[13] E. Zhang, X. Wang, P. Gong, Y. Lin, J. Mao, USimAgent: Large language models for simulating search users, in: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2024, pp. 2687–2692. doi:10.1145/3626772.3657961.

[14] J. Carbonell, J. Goldstein, The use of MMR, diversity-based reranking for reordering documents and producing summaries, in: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1998, pp. 335–336. doi:10.1145/290941.291025.