University of Amsterdam at the CLEF 2026 SimpleText Track

Pascal Mathas, Berkay Chakar, David Carranza Navarrete, Jan Bakker and Jaap Kamps

University of Amsterdam, Amsterdam, The Netherlands

Abstract

This paper reports on the University of Amsterdam's participation in the CLEF 2026 SimpleText track. We participated in Task 1.2 document-level text simplification, Task 2 for overgeneration detection and classification, and Task 3 for research area classification. For document-level text simplification, we submit a baseline, a plan-guided variant, and a retrieval-augmented method. We find that the RAG method, which provides abstract-PLS section pairs as in-context examples to the simplification model, outperforms both our baseline and our plan-guided variant, with the largest gains in the simplification of full Cochrane abstracts.

Keywords

Natural Language Processing, Biomedical NLP, Text Simplification, Overgeneration Detection

1. Introduction

For details on the exact track setup, we refer to the Track Overview paper CLEF 2026 LNCS proceedings [1], as well as the detailed task overviews in the CEUR proceedings [2, 3, 4].

The rest of this paper is structured as follows. In three separate sections, we discuss our experiments for each of the tasks. Section 2 discusses simplify scientific text. Section 3 details identify and avoid hallucination. Section 4 completes our submissions with a zero-shot approach to classification of scientific articles by research area. We end in Section 5 by discussing our results and outlining the lessons learned.

2. Task 1: Text Simplification

In this section, we discuss our proposed methods for SimpleText Task 1.2, document-level simplification of scientific text. We describe the data, the systems we submit, and our results.

2.1. Task and Data

2.1.1. Task

Task 1 of the CLEF 2026 SimpleText Track consists of two subtasks: (1.1) sentence-level scientific text simplification, where each Cochrane abstract is split into individual sentences and evaluated at the sentence level, and (1.2) document-level scientific text simplification, where each abstract is evaluated as a single document. We submit runs only for Task 1.2, as our systems were developed and optimized for document-level simplification.

2.1.2. Data

CLEF 2026 SimpleText. The CLEF 2026 SimpleText test set extends the 2025 setup with Cochrane reviews published since March 2025. It consists of 176 abstract-PLS pairs, of which 32 are highly aligned at the sentence level. These 32 better-aligned pairs are referred to as `simple_auto` and the full set as

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

j.bakker@uva.nl (J. Bakker); kamps@uva.nl (J. Kamps)

0009-0002-9085-8491 (J. Bakker); 0000-0002-6614-0087 (J. Kamps)

`simple_original`. The test set additionally includes reviews in 9 other languages and English abstracts covering all seven Cochrane sections rather than only two. The test set covering all seven sections is referred to as `simple_full`. We report all scores against the references used in Codabench[1].

Cochrane-sections. All our proposed systems use the Cochrane-sections[2] dataset [5], which provides 7.7K English abstract-PLS pairs derived from Cochrane systematic reviews and aligned at the section level. Unlike Cochrane-auto[3] [6], which covers only the *Main results* and *Authors' conclusions* sections, Cochrane-sections covers all seven sections of a Cochrane abstract, adding *Background*, *Objectives*, *Search methods*, *Selection criteria*, and *Data collection and analysis*.

We use the Cochrane-sections dataset in two ways. First, we use its sentence-level alignments to train a sentence-level planner. This planner predicts operation labels for the sentences in an abstract, which we then use to guide the LLM (runs denoted as PG-). Second, we use its section- and sentence-level alignments to provide topically related abstract-PLS section pairs as in-context examples, each split into rephrased and inserted sentences (runs denoted as RAG-).

2.2. Methods

Our systems are developed on Cochrane-sections and are therefore intended to simplify whole Cochrane abstracts that include all seven sections. Cochrane-auto and the SimpleText test set cover only a subset of these sections, but since they share the same abstract format, our systems can be applied to them without modification.

2.2.1. Setup

For generation we primarily use Qwen2.5-32B-Instruct [7, 8] (denoted as Q32), run with the vLLM library [9] for faster inference and lower memory use. One of our RAG variants also uses Llama-3.1-70B-Instruct [10] (denoted as L70). We run all models in bfloat16, and set the temperature to 0.2, top-p to 0.9, repetition penalty to 1.1, and max new tokens to 2048. All systems simplify the abstract sentence by sentence, using the surrounding section name and text as context.

Since we provide the section as context, we must assign a section label to every sentence in the test set. Some abstracts already contain section headers, which we map to one of the seven Cochrane sections. As there are more header variants than sections, several variants map to the same section (e.g. *synthesis of results* to *Main results*). For abstracts without headers, we classify each sentence into one of the seven sections using Q32.

The prompts for all our systems can be found in Appendix A. The baseline uses only the `<<HEADER>>` block, the plan-guided system adds the `<<PG INTRO>>` block, and the RAG systems the `<<RAG INTRO>>` block. The baseline is denoted as LLM_Q32.

2.2.2. Plan-guided LLM

Our plan-guided system follows the planner of Cripwell et al. [11], who used it in combination with a BART-based simplification model. We adopt only the planner and use it to predict an operation label for each sentence of the CLEF 2026 SimpleText test set. We then extend the baseline prompt with the `<<PG INTRO>>` block, which defines these labels, and ask Q32 to simplify each sentence according to its predicted operation.

We train the planner on the Cochrane-sections train set. Because the operation labels are highly imbalanced, we add a class-weighted cross-entropy loss that assigns greater weight to less frequent labels, improving performance on the minority classes. We denote this system as PG_PCW_Q32.

[1] https://www.codabench.org/competitions/16108/
[2] github.com/JanB100/cochrane-sections
[3] github.com/JanB100/cochrane-auto

Table 1
Evaluation against `simple_original` on CLEF 2026 SimpleText test set, sorted by SARI. Best results are in **bold**.

Run	BLEU ↑	SARI ↑	FKGL ↓	Cmprss. ratio	Sent splits	Lev sim	Exact copies	Add prop	Del prop	Lex cmplx score
RAG_BoN_k4_Q32L70	12.80	**47.57**	12.34	0.75	0.90	0.51	0.00	0.31	0.60	8.44
RAG_BoN_k4_Q32	11.07	46.67	11.00	0.58	0.82	0.50	0.00	0.19	0.64	8.52
PG_PCW_Q32	12.51	46.50	10.37	0.71	1.13	0.50	0.00	0.27	0.60	8.56
LLM_Q32	9.41	46.20	10.55	0.56	0.83	0.49	0.00	0.20	0.67	8.52
RAG_k4_Q32	9.90	46.11	10.81	0.54	0.78	0.49	0.00	0.17	0.66	8.53

2.2.3. Retrieval-augmented LLM

The Cochrane-sections dataset provides two alignments that we use in our RAG systems. First, each PLS sentence is classified into a section. Second, each PLS sentence is aligned to a complex counterpart in the abstract. A PLS sentence without a counterpart is treated as inserted content, present only in the PLS. At inference time, these alignments let us retrieve, for each section of the target abstract, the top-k most similar sections and split each into rephrased sentences (those with a complex counterpart) and inserted sentences (those without).

We build the RAG index by embedding each abstract section of the Cochrane-sections train set with the MedCPT article encoder [12]. At inference time, we embed each section of the target abstract with the MedCPT query encoder and retrieve the top-k most similar sections by cosine similarity. For each retrieved section we add to the prompt the complex abstract section together with its rephrased and inserted PLS sentences. This way, the LLM is shown section-level examples of both the rephrasing and the insertion behavior it should reproduce when writing the PLS for the target section.

All our RAG systems use $k = 4$ together with the `<<HEADER>>` and `<<RAG INTRO>>` prompts (Appendix A). The base RAG system is denoted RAG_k4_Q32. We additionally experiment with a best-of-N (BoN) method. For each sentence, we generate $N = 6$ candidates at temperature 0.6, drop the sentence if a majority of the candidates delete it, and otherwise keep the candidate with the highest token-overlap F1 against the retrieved PLS sentences. We denote the system that generates all 6 candidates with Q32 as RAG_BoN_k4_Q32, and the system that generates 3 with Q32 and 3 with L70, to exploit differences in the two models' behavior, as RAG_BoN_k4_Q32L70.

2.3. Results and Discussion

As discussed earlier, our systems are intended to simplify abstracts that cover all seven sections, which makes it harder to interpret an evaluation against only two of them (Tables 1 and 2). For `simple_original`, RAG_BoN_k4_Q32L70 leads the table on SARI but also has the highest FKGL. This is because L70 is stronger at deciding what to keep and add, which is reflected in the run also having the highest add proportion and compression ratio. It is also interesting that RAG_k4_Q32 ranks below our baseline LLM_Q32, but understandable as the baseline is already quite good at simplifying the *Main results* and *Authors' conclusions* sections, and the RAG systems gain the most in other sections.

If we look at the `simple_full` results, both per section and for the whole PLS (Tables 3 and 4), we see a clear advantage for the RAG variants, with all three runs occupying the top three. This advantage of having related sections as examples becomes especially clear for *Search methods* and *Selection criteria*, where the baseline LLM_Q32 and its plan-guided variant struggle to simplify these sections effectively. The system hierarchy in Table 4 is exactly as expected, with the plan-guided variant improving slightly over the baseline and the RAG variants performing better across the board. The BoN RAG variants generate each sentence multiple times, which introduces a trade-off between added performance and complexity.

Table 5 presents the average scores assigned by LENS and three small language models acting as judges to the simplification outputs generated by the five systems. Scores are reported on a 0–100 scale.

Table 2
Evaluation against `simple_auto` on CLEF 2026 SimpleText test set, sorted by SARI. Best results are in **bold**.

Run	BLEU ↑	SARI ↑	FKGL ↓	Cmprss. ratio	Sent splits	Lev sim	Exact copies	Add prop	Del prop	Lex cmplx score
RAG_BoN_k4_Q32	15.04	**46.81**	10.87	0.57	0.84	0.51	0.00	0.18	0.65	8.45
PG_PCW_Q32	13.02	46.61	10.36	0.79	1.29	0.52	0.00	0.31	0.57	8.50
RAG_BoN_k4_Q32L70	12.26	46.20	12.00	0.72	0.93	0.51	0.00	0.29	0.60	8.37
RAG_k4_Q32	14.89	46.09	10.76	0.52	0.79	0.49	0.00	0.16	0.67	8.47
LLM_Q32	13.03	45.65	10.50	0.52	0.84	0.48	0.00	0.19	0.69	8.48

Table 3
Evaluation against `simple_full`, per section on CLEF 2026 SimpleText test set, sorted by mean SARI. Best results are in **bold**.

Run	Back-ground	Objec-tives	Search methods	Selection criteria	Data coll. & analysis	Main results	Authors' conclusions	Mean
RAG_BoN_k4_Q32L70	**44.83**	40.15	**37.59**	**40.16**	**34.53**	**44.22**	**44.99**	**40.93**
RAG_BoN_k4_Q32	44.07	**40.93**	33.85	39.09	33.77	44.00	43.94	39.95
RAG_k4_Q32	43.96	40.64	33.53	39.08	33.71	43.79	43.63	39.76
LLM_Q32	43.46	37.36	5.17	23.80	32.85	43.78	43.82	32.89
PG_PCW_Q32	43.99	36.62	7.32	23.57	12.77	43.92	43.26	30.21

Table 4
Evaluation against `simple_full` on CLEF 2026 SimpleText test set, sorted by SARI. Best results are in **bold**.

Run	SARI ↑
RAG_BoN_k4_Q32L70	**49.68**
RAG_BoN_k4_Q32	48.40
RAG_k4_Q32	47.98
PG_PCW_Q32	47.35
LLM_Q32	46.70

Table 5
System evaluation using LENS and LLM-as judge, on CLEF 2026 (only English), sorted by LENS.

Run	LENS μ	LENS σ	Llama-3.1-8B. μ	Phi-3-mini-4k. μ	Qwen2.5-7B. μ
RAG_BoN_k4_Q32L70	68.31	5.61	67.64	38.52	72.27
LLM_Q32	67.50	8.11	60.23	51.80	73.62
RAG_k4	65.71	7.45	62.97	46.04	72.61
RAG_BoN_k4_Q32	64.99	7.76	66.11	44.12	72.39
PG_PCW_Q32	57.39	9.20	49.31	43.90	69.33

Since LENS was trained exclusively on English data, only the English subset of the CLEF 2026 dataset was included in this evaluation. The experimental setup used the system-generated simplifications together with the corresponding reference simplifications for the 163 English sentences in the dataset. The three LLM judges were prompted using the same questionnaire employed in the original LENS evaluation framework [13]. Overall, the rankings produced by the LLM judges show varying degrees of agreement with LENS, with Qwen2.5-7B exhibiting the closest alignment to it.

The LENS scores only partially agree with our SARI results. Although the RAG_BoN_k4_Q32L70 variant leads the table, our baseline LLM_Q32 comes in a close second, despite ranking near the bottom on SARI. It is important to note that our evaluation in Table 5 is of narrower scope, as LENS is computed at the sentence level on the highly aligned subset, covering only two of the seven sections. Additionally,

Table 6
UvA submissions for SimpleText Task 2.2.

Run	Source	Model	Description
UvA_Task22a_GPT41Mini_nosource.zip	No	GPT-4.1-mini	Zero-shot no-source prompt
UvA_Task22a_GPT41Mini_PrecisionV2.zip	No	GPT-4.1-mini	Precision-oriented no-source prompt
UvA_Task22a_Qwen3_14B_nosource.zip	No	Qwen3-14B	Zero-shot open-weight no-source prompt
UvA_Task22b_GPT41Mini_GroundedV1.zip	Yes	GPT-4.1-mini	Zero-shot source-aware prompt

Table 7
Official CodaBench scores for the UvA Task 2.2 submissions.

Run	Doc. F1	Prec.	Rec.	Sent. F1	Cat. Acc.
UvA_Task22a_GPT41Mini_nosource.zip	0.4291	0.3220	0.6428	0.3980	0.4178
UvA_Task22a_GPT41Mini_PrecisionV2.zip	0.4339	0.3236	0.6584	0.4113	0.4454
UvA_Task22a_Qwen3_14B_nosource.zip	0.4056	0.2718	0.7989	0.3351	0.3463
UvA_Task22b_GPT41Mini_GroundedV1.zip	0.4061	0.2758	0.7695	0.3117	0.5032

LENS was trained on more general-domain data instead of biomedical text, which could explain why it rewards the more conservative LLM_Q32 baseline.

3. Task 2: Controlled Creativity

Task 2 asks systems to identify overgeneration in simplified biomedical abstracts. The input consists of a source Cochrane abstract and a generated simplified version, already split into sentences. Each sentence must be labeled as either non-overgenerated or as one of the overgeneration categories. Task 2.1 evaluates binary detection, while Task 2.2 evaluates the full multi-class taxonomy. Both tasks have a no-source setting, where only the generated sentences are available, and a with-source setting, where the source abstract is also provided. We submitted runs only for Task 2.2.

3.1. Task and Data

The test file contains 6,728 examples and 105,001 generated sentences. Each example contains an identifier, the source biomedical abstract, the generated simplified sentences, and placeholder labels. The primary leaderboard metric is document-level F1: sentence labels are first converted to binary, and a document is positive if at least one sentence is labeled as overgenerated.

3.2. Proposed Methods

Table 6 lists our submitted runs. All runs use zero-shot prompting with an output-format example; no model was trained or fine-tuned. The prompt asked the model to return one label per sentence index using the official Task 2.2 taxonomy. The GPT runs used `gpt-4.1-mini` through an OpenAI-compatible chat API with temperature 0. The Qwen run used `Qwen3-14B` on Google Colab with greedy decoding and a batch size of 8. The `PrecisionV2` run is a conservative no-source prompt variant, introduced after observing low precision in the initial no-source run.

3.3. Results and Discussion

Table 7 shows the official CodaBench scores. The best document-level F1 is obtained by the precision-oriented GPT-4.1-mini no-source run, although the gain over the initial GPT no-source prompt is small. Qwen3-14B obtains high recall but lower precision, indicating over-flagging. The source-aware GPT run achieves the highest category accuracy but a lower document-level F1 due to many false-positive documents. Overall, the no-source runs show that zero-shot LLM prompting can detect some

Table 8
CLEF 2026 SimpleText Task 3 Train Data Distribution

Discipline Area		Field Area	
Descriptor	**Number**	**Descriptor**	**Number**
Natural Sciences	9,409	*Physics*	4,995
		Mathematics	4,001
		Chemistry	240
		Geosciences	173
Engineering Sciences	1,601	*Computer Science*	1,254
		Electrical Engineering and Information Technology	207
		Mechanical and Industrial Engineering	69
		Systems Engineering	50
		Civil Engineering	12
		Materials Science	7
		Process Engineering	2
Life Sciences	907	*Biology*	696
		Medicine	211
		Veterinary Medicine	0
Social Sciences	361	*Social and Behavioural Sciences*	194
		Psychology	91
		Education	76
		Social Sciences	0
Humanities	23	*Linguistics*	15
		Archeology	5
		Literary Studies	3
		Anthropology	0
		Humanities	0

obvious overgeneration without access to the source, but precision remains the main bottleneck. This is particularly relevant for the no-source setting, where the model cannot verify factual support and must instead identify visible signs of overgeneration from the generated abstract alone.

4. Task 3: Research Area Classification

We continue with Task 3, asking for classification of scientific articles by research area.

4.1. Task and Data

Task 3 asks to identify the `Discipline Area` and the `Field Area` given a publication record containing an `ID`, a `Title`, and an `Abstract`. For details, we refer to the track overview [1] and task overview [4] papers.

Table 8 shows the train data distribution. We note that the taxonomy used is limited, with only a very few distinct disciplines and research areas. We also note that the sample of bibliographic records to be classified is highly imbalanced, with a clear overrepresentation of the sciences and some specific fields within those.

4.2. Proposed Methods

Table 9 details the submissions. Noting the skewed data distribution, our first submission tested the evaluation setup in CodaBench with a simple majority-class baseline. Our second submission was a zero-shot prompt of a closed-source LLM (Gemini 3.5 Flash), feeding the full taxonomy, and without any awareness of the extreme class imbalance.

Table 9
CLEF 2026 SimpleText Task 3 Submissions, official run names start with UAmsterdam_Task3_...

Task	Run	Description
3	UAmsterdam_1	Majority class baseline (Natural Sciences, Physics).
3	UAmsterdam_2	Zero-shot LLM, Gemini 3.5 Flash (each test title and abstract individually, with the full taxonomy).

Table 10
Analysis of SimpleText Task 3: simplify scientific text

Team Name	Discipline Area			Field Area		
	EM	SD	ED	EM	SD	ED
UAmsterdam_2	0.6122	0.7035	0.7759	0.7033	0.8279	0.8472
UAmsterdam_1	0.0625	0.2281	0.4099	0.2000	0.6225	0.6659

4.3. Results and Discussion

Table 10 details the results. The majority class baseline still performs reasonably on Exact Match scores and is surprisingly effective on String Distance and Embedding Distance measures. The performance of the majority-class baseline indicates that the test set has a similar extreme class imbalance to that of the training data. This suggests using macro-averages rather than plain accuracy and F1 scores, which are dominated by the majority class.

The zero-shot LLM performs significantly better than the majority-class baseline. Even though the Field Area is a harder prediction problem due to the higher number of classes, it performs better than the Discipline Area. This may be due to the classification of multidisciplinary records. An informal inspection suggests that these are not classified according to standard human cataloging principles, with the systematic design of the controlled vocabulary taken into account. Rather, these seem to resemble superficial, non-expert classification, such as that obtained from crowdsourced workers or LLMs unaware of the classification systems used by experts for cataloging. It would be of interest to evaluate the task by taking the taxonomic relations (as well as associative related class relations) of the used classification system, and quantifying proper taxonomic distance measures.

5. Discussion and Conclusions

This paper detailed the University of Amsterdam's participation in the CLEF 2026 SimpleText track. We conducted a range of experiments for the different tasks of the track.

Our primary focus was on the core Task 1 on *Text Simplification*, where we evaluated multiple approaches to scientific text simplification. For this task, we submitted a baseline, a plan-guided variant, and a retrieval-augmented method. We found that the RAG method, which provides abstract-PLS section pairs as in-context examples to the simplification model, outperforms both our baseline and our plan-guided variant, with the largest gains on sections the other methods simplify poorly (*Search methods* and *Selection criteria*). Our experiments for Task 2 on *Controlled Creativity* focused on baseline approaches, in particular, to analyze the effectiveness of detecting overgeneration without access to the sources. Finally, we included a zero-shot baseline approach for Task 3 on *Research Area Classification*.

Acknowledgments

This research was conducted as part of final research projects for the Master's in Artificial Intelligence program and the Master's of Logic program at the University of Amsterdam.

We thank the track and task organizers for their amazing service and effort in making realistic benchmarks for scientific text simplification available.

Jan Bakker and Jaap Kamps are partly funded by the Netherlands Organization for Scientific Research (NWO NWA # 1518.22.105). Jaap Kamps is further supported by the University of Amsterdam (AI4FinTech program) and ICAI (AI for Open Government Lab). Views expressed in this paper are not necessarily shared or endorsed by those funding the research.

Declaration on Generative AI

During the preparation of this work, the authors used *Grammarly* in order to: **Grammar and spelling check**. After using these tools/services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] L. Ermakova, H. Azarbonyad, J. Bakker, B. Chakar, G. K. Shahi, J. Kamps, Overview of the CLEF 2026 SimpleText track: Simplify scientific texts (and nothing more), 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, Springer, 2026.

[2] J. Bakker, L. Ermakova, J. Kamps, Overview of the CLEF 2026 SimpleText Task 1: Simplify Scientific Text, in: [14], 2026.

[3] B. Chakar, J. Bakker, L. Ermakova, J. Kamps, Overview of the CLEF 2026 SimpleText Task 2: Identify and Avoid Hallucination, in: [14], 2026.

[4] G. K. Shahi, L. Ermakova, J. Kamps, Overview of the CLEF 2026 SimpleText Task 3: Research Area Classification, in: [14], 2026.

[5] J. Bakker, J. Kamps, Section-level simplification of biomedical abstracts, in: Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, 2025, pp. 13830–13844.

[6] J. Bakker, J. Kamps, Cochrane-auto: An aligned dataset for the simplification of biomedical abstracts, in: Proceedings of the Third Workshop on Text Simplification, Accessibility and Readability (TSAR 2024), 2024, pp. 41–51.

[7] Q. Team, Qwen2.5: A party of foundation models, 2024. URL: https://qwenlm.github.io/blog/qwen2.5/.

[8] A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, C. Li, C. Li, D. Liu, F. Huang, G. Dong, H. Wei, H. Lin, J. Tang, J. Wang, J. Yang, J. Tu, J. Zhang, J. Ma, J. Xu, J. Zhou, J. Bai, J. He, J. Lin, K. Dang, K. Lu, K. Chen, K. Yang, M. Li, M. Xue, N. Ni, P. Zhang, P. Wang, R. Peng, R. Men, R. Gao, R. Lin, S. Wang, S. Bai, S. Tan, T. Zhu, T. Li, T. Liu, W. Ge, X. Deng, X. Zhou, X. Ren, X. Zhang, X. Wei, X. Ren, Y. Fan, Y. Yao, Y. Zhang, Y. Wan, Y. Chu, Y. Liu, Z. Cui, Z. Zhang, Z. Fan, Qwen2 technical report, arXiv preprint arXiv:2407.10671 (2024).

[9] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, I. Stoica, Efficient memory management for large language model serving with pagedattention, in: Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles, 2023.

[10] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al., The llama 3 herd of models, arXiv preprint arXiv:2407.21783 (2024).

[11] L. Cripwell, J. Legrand, C. Gardent, Document-level planning for text simplification, in: Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, 2023, pp. 993–1006.

[12] Q. Jin, W. Kim, Q. Chen, D. C. Comeau, L. Yeganova, W. J. Wilbur, Z. Lu, Medcpt: Contrastive pre-trained transformers with large-scale pubmed search logs for zero-shot biomedical information retrieval, Bioinformatics 39 (2023) btad651.

[13] M. Maddela, Y. Dou, D. Heineman, W. Xu, LENS: A learnable evaluation metric for text simplification, in: A. Rogers, J. Boyd-Graber, N. Okazaki (Eds.), Proceedings of the 61st An-

nual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Toronto, Canada, 2023, pp. 16383–16408. URL: https://aclanthology.org/2023.acl-long.905/. doi:10.18653/v1/2023.acl-long.905.
[14] E. Sánchez Salido, A. Barrón Cedeño, A. García Seco de Herrera, S. MacAvaney, J. M. Struß (Eds.), Working Notes of CLEF 2026: Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, CEUR-WS.org, 2026.

A. Task 1.2 Prompts

A.1. Baseline Text Simplification

Our baseline prompt is as follows:

```
<<HEADER>>

You are rewriting sentences from a Cochrane biomedical review abstract as they would appear
↪  in a Cochrane plain-language summary (PLS). A PLS is written for a general reader with
↪  no medical training.

You work one sentence at a time. You receive the section name, the section text for context,
↪  and the current sentence. Decide whether the sentence belongs in the PLS and rewrite
↪  it, or drop it. Use the section name to judge how much is likely to stay: Search
↪  methods, Selection criteria, and Data collection and analysis are almost always
↪  dropped; Main results and Authors' conclusions are mostly rewritten; Background and
↪  Objectives sit in between.

Plain-language style
- Short, direct sentences.
- Replace clinical jargon with everyday words ("randomised controlled trial" -> "study";
↪  "participants" -> "people"; "adverse events" -> "side effects").
- Remove effect sizes, confidence intervals, p-values, statistical tests, and methodology.
↪  Report findings qualitatively ("more effective than", "no clear difference").
- Drop procedural detail (database names, search dates, risk-of-bias tools, inclusion
↪  criteria wording).
- Use active voice where natural. Refer to the review as "this review" or "we".

When to drop a sentence
If the sentence is purely procedural (search strategy, statistical method, risk-of-bias
↪  assessment, inclusion criteria wording, trial registration detail) or reports only
↪  numeric results without a clinically meaningful finding, output exactly: [DELETE]

Output
- Output only the rewritten sentence or [DELETE].
- No explanations, no prefixes, no markdown.
```

A.2. Plan-Guided Text Simplification

Our plan-guided text simplification system extends the baseline prompt with the following prompt:

```
<<PG INTRO>>

Operation label
The current sentence is labeled with an operation that specifies how to edit it. Apply the
↪  operation:
- COPY: Output the sentence exactly as written. Used only when the sentence is already in
↪  plain language.
- REPHRASE: Rewrite the sentence in plain-language style. Replace technical terms and
↪  jargon that a non-expert would not understand.
```

```
- SPLIT: Rewrite the sentence into two plain-language sentences, each should contain one
↪  piece of the original content.
- MERGE: Combine this sentence with the following NONE-labeled sentence into a single
↪  plain-language sentence.
- NONE: Part of the preceding MERGE group. Output exactly: [NONE]
- DELETE: This sentence does not appear in the PLS. Output exactly: [DELETE]
```

A.3. Retrieval-Augmented Text Simplification

Our retrieval-augmented text simplification system extends the baseline prompt with the following prompt:

```
<<RAG INTRO>>

Below are {k} examples of how the {section_name} section of a Cochrane PLS is typically
↪  written for reviews similar in topic. For each example we show three things: the
↪  original abstract section, the PLS sentences that are rewrites of kept abstract
↪  sentences, and the PLS sentences that the reviewer added without a counterpart in the
↪  abstract. Use them to guide STYLE, PHRASING, LENGTH, the KIND OF CONTENT that belongs
↪  in this section, and the KIND OF EXPLANATORY ADDITIONS PLS authors typically introduce.
↪  Do NOT copy specific findings, drug names, or numbers from them; those belong to other
↪  reviews.
```