Sentence-level Scientific Text Simplification With Just a Pinch of Data

Notebook for the SimpleText Lab at CLEF 2025

Marvin M. Agüero-Torales[1,*], Carlos Rodríguez Abellán[1] and Carlos A. Castaño Moraga[1]

[1]*CoE of Data Intelligence, Fujitsu, Camino Cerro de los Gamos, 1, Pozuelo de Alarcón, 28224, Madrid, Spain*

Abstract

We present our *CLEF 2025 SimpleText Task 1.1* submission (*Lenguaje-Claro* team), demonstrating that competitive sentence simplification of scientific text can be achieved with only a 'pinch' of high quality data. Our approach uses `GPT-3.5-Turbo`, `o4-mini` and `T5-Efficient` with zero-shot and three-shot prompt on three sample sentences, complemented by a lightweight ensemble and rule-based model simplifiers. Then, a unified LLM-based judge selects or, if necessary, regenerates outputs below a quality threshold. Experiments show that `GPT-3.5-Turbo` with a three-shot prompt outperforms all other modules, establishing a good baseline in data-scarce settings.

Keywords

Text Simplification, Plain Language, Few-Shot Learning, Ensemble Methods, LLM-as-a-judge, Low-resource NLP

1. Introduction

Simplifying scientific sentences for general audiences is vital, yet challenging, especially when parallel corpora (simplified and original pair) are scarce [1]. We explore how 'just a pinch' of data (.i.e., three synthetic high quality sentence pairs) can bootstrap strong simplification performance. Our system integrates:

- Zero and three-shot prompting of `GPT-3.5-Turbo` [2], `o4-mini` [3] and `T5-Efficient-small` [4] models,
- A rule-based model,
- A lightweight ensemble (choosing the shorter simplified text) and,
- A unified Large Language Model (LLM) based auto-evaluation [5] and fallback generation mechanism of the best results of the aforementioned.

This pipeline achieves reasonable *SARI*[1] scores while minimizing data and computation.

Contribution Our main contribution in this summary is to report the results of different systems, models, strategies, and approaches in the test set when data is almost non-existent and the novel introduction of a unified LLM-based judge for text simplification and plain language.

2. Methodology

In this section, we describe our methodology in more detail about our participation in *SimpleText* shared-task [7] about scientific text simplification at sentence-level [8]. First, we describe the data

CLEF 2025 Working Notes, 9 – 12 September 2025, Madrid, Spain

*Corresponding author.

marvinmatias.aguerotorrales@fujitsu.com (M. M. Agüero-Torales); carlos.rodriguezabellan@fujitsu.com (C. Rodríguez Abellán); carlosalberto.castanomoraga@fujitsu.com (C. A. Castaño Moraga)

https://mmaguero.github.io/ (M. M. Agüero-Torales); https://www.linkedin.com/in/carlosrodriguezabellan/ (C. Rodríguez Abellán); http://www.linkedin.com/in/carlos-alberto-castaÃśo-moraga-887a5854 (C. A. Castaño Moraga)

0000-0002-0910-0310 (M. M. Agüero-Torales)

[1]The SARI score is the arithmetic mean of the accuracies and recoveries of n-grams for add, keep and delete. A higher SARI score indicates greater simplicity or readability [6].

used for few-shot prompt, the simplifiers, and then the ensembles and LLM-judges over the simplifiers' results.

2.1. Few-Shot Prompting with Pinch of Data

We curate three representative pairs of complex-simple sentence samples for a few-shot prompting with `Microsoft Copilot` tool [9]. These examples were synthetically created to cover three core phenomena rather than being chosen at random: (i) biomedical terminology, (ii) numerical information, and (iii) discourse marker splitting. Previous studies [10, 11] showed that high quality targeted samples outperform random sampling (in our pilots, more than 0.2 SARI), demonstrating that a small but carefully curated 'pinch' of data can reliably guide LLM simplification.

Each input to `GPT-3.5-Turbo` and `o4-mini` is prefixed with the prompt listed in Listing 1.

```
You are a helpful plain language assistant for clear, easy, plain,
   and simple text simplification.

Simplify the following scientific-style sentence into a concise,
   plain-English sentence that preserves meaning.
Example 1: Complex: "..." Simplified: "..."
Example 2: Complex: "..." Simplified: "..."
Example 3: Complex: "..." Simplified: "..."

Now simplify:
Complex: "..."
Simplified (limit to max. 45 characters):
```

Listing 1: Prompt for GPT-based models.

On the other hand, for the `T5-Efficient-small` model we use a more simple prompt (see Listing 2). These minimal settings ensure that the model learns key simplification patterns without extensive fine-tuning.

```
Simplify:
Input: "..." Output: "..."
Input: "..." Output: "..."
Input: "..." Output: "..."

Simplify:
Input: "..."
Output (limit to max. 45 characters):
```

Listing 2: `T5-Efficient-small` prompt.

2.2. Modular Simplifiers

In addition to the aforementioned models, we implement a rule-based simplifier and ensemble model.

Rule-based simplifier The rule-based simplifier removes parentheticals and splits at discourse markers. We used these markers {and, but, or, so, because} for the *simple* approach, and then add these for the *complex* approach {although, however, therefore, moreover, meanwhile, nevertheless, nonetheless, yet, still, then, thus, consequently}.

Ensemble simplifier To improve robustness and ensure output quality in low-data scenarios, we implemented a lightweight heuristic ensemble strategy that selects the best simplification among available candidates. Specifically, we collect output from the rule-based, T5-based, and optionally GPT-based modules, discard any empty or whitespace-only strings, and then choose the *shortest* nonempty output inspired by readability studies linking brevity to simplicity. In a quantitative analysis, 68% of the shortest candidates retained the full factual content while removing peripheral clauses, justifying the length as a proxy for simplicity [12].

In cases of ties, we apply a fixed priority: `GPT > T5 > Rule`. This simple yet effective selection mechanism favors brevity and leverages the higher performance of GPT-based simplifications when available. Our ensembler uses the logic shown in Appendix A.

2.3. Unified Evaluation and Ensemble

The outputs with the three highest SARI metric scores (see the results in Table 1) are passed as candidates to our LLM-as-a-judge `Unified Evaluator`, which scores each simplification candidate in terms of **fluency**, **adequacy**, and **simplicity** on a scale from 1 to 5. This scoring is based on a consistent prompting format in which the original and simplified sentences are provided, and the model is asked to return three comma-separated numbers (one per criterion).

The evaluator is designed to be model-agnostic and supports both local Hugging Face models (e.g., LLaMA, Gemma, etc.) and remote deployments via `Azure OpenAI` [13] (e.g., `GPT-3.5-Turbo`) through the `ChatCompletion` API [14]. If the average score of all three criteria for the top candidate falls below a configurable threshold (2.5 in our setting), the evaluator triggers a fallback regeneration process using `GPT-3.5-Turbo`. This regeneration prompt differs slightly from Listing 1 and is more concise (see Listing 3).

```
Please simplify this scientific sentence into concise, plain English
    (limit to max. 45 characters): "...".
```

Listing 3: `Unified Evaluator`'s fallback prompt.

The freshly generated simplification replaces the weaker candidates. The final selection always favors the highest-rated candidate or the regenerated version when necessary.

This unified evaluation pipeline enables automatic quality assessment and controlled generation in a consistent and scalable manner across different model settings.

3. Experiments

We evaluated our systems and models on the official *Task 1.1* test set. Additionally, we performed experiments with the text length of complex sentences, in order to define the adequate length of the simplified sentences (`Truncation` in Table 1). We set the optimal length to 45 characters. Table 1 summarizes the performance.

Although absolute differences are modest, `GPT-3.5-Turbo` with three-shot prompting consistently surpasses all other modules by $+1.8$ SARI average. Fallback generation recovers 0.3 SARI when initial candidates fail.

4. Conclusion

We demonstrate that high quality sentence simplification can be obtained with minimal and synthetic data via strategic prompt engineering and a modular ensemble. `GPT-3.5-Turbo` with three-shot prompting sets a new low-resource baseline for *CLEF SimpleText Task 1.1*. Future work will explore automated example selection, use no-synthetic few-shot data, try more setups, and domain adaptation.

Table 1
Test set results. The best system/model for every experiment setting are underlined. Three-shot `GPT-3.5-Turbo` (in bold) achieves the best performance.

System/Model	Approach	SARI Score ↑
Truncation	20 char length	36.01
	30 char length	36.68
	40 char length	36.89
	45 char length	36.92
	50 char length	36.84
	60 char length	36.49
	90 char length	34.51
Rule-based model	Simple	34.00
	Complex	34.13
T5Efficient	Zero-shot	36.55
	3-shot	33.89
GPT-3.5-Turbo	Zero-shot	38.49
	3-shot	**38.84**
o4-mini	Zero-shot	37.82
	3-shot*	38.20
Ensemble	T5Efficient (zero-shot) + Rule-based model (complex)	34.48
	GPT-3.5-Turbo (3-shot) + T5Efficient (zero-shot) + Rule-based model (complex)	38.55
Unified Judge	with Fallback (3 best results)	38.54
	without Fallback (3 best results)	38.50

* Post-Competition submission.

Acknowledgments

This research was supported by cloud credits from *Fujitsu*'s *Microsoft Azure* subscription. The authors thank the organizers of *CLEF 2025* and *SimpleText Lab 2025* for creating the perfect ecosystem for the shared task.

Declaration on Generative AI

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

References

[1] B. Ondov, K. Attal, D. Demner-Fushman, A survey of automated methods for biomedical text simplification, Journal of the American Medical Informatics Association 29 (2022) 1976–1988. doi:10.1093/jamia/ocac149.

[2] OpenAI, Model - openai api - gpt-3.5-turbo, 2025. https://platform.openai.com/docs/models/gpt-3.5-turbo [Date: 2025-06-16].

[3] OpenAI, Model - openai api - o4-mini, 2025. https://platform.openai.com/docs/models/o4-mini [Date: 2025-06-16].

[4] Y. Tay, M. Dehghani, J. Rao, W. Fedus, S. Abnar, H. W. Chung, S. Narang, D. Yogatama, A. Vaswani, D. Metzler, Scale efficiently: Insights from pretraining and finetuning transformers, in: International Conference on Learning Representations, 2022. URL: https://openreview.net/forum?id=f2OYVDyfIB.

[5] J. Liu, Y. Nam, X. Cui, S. Swayamdipta, Evaluation under imperfect benchmarks and ratings: A case study in text simplification, 2025. URL: https://arxiv.org/abs/2504.09394. arXiv:2504.09394.
[6] W. Xu, C. Napoles, E. Pavlick, Q. Chen, C. Callison-Burch, Optimizing statistical machine translation for text simplification, Transactions of the Association for Computational Linguistics 4 (2016) 401–415. URL: https://aclanthology.org/Q16-1029/. doi:10.1162/tacl_a_00107.
[7] L. Ermakova, H. Azarbonyad, J. Bakker, B. Vendeville, J. Kamps, Overview of the CLEF 2025 SimpleText track: Simplify scientific texts (and nothing more), in: J. Carillo de Albornoz, et al. (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Fifteenth International Conference of the CLEF Association (CLEF 2025), LNCS, Springer-Verlag, 2025.
[8] J. Bakker, B. Vendeville, L. Ermakova, J. Kamps, Overview of the clef 2025 simpletext task 1: Simplify scientific text, in: G. Faggioli, et al. (Eds.), Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2025), CEUR Workshop Proceedings, CEUR-WS.org, 2025.
[9] Microsoft, What is microsoft 365 copilot?, 2025. https://learn.microsoft.com/en-us/copilot/microsoft-365/microsoft-365-copilot-overview [Date: 2025-06-18].
[10] T. Schick, H. Schütze, Few-shot text generation with natural language instructions, in: M.-F. Moens, X. Huang, L. Specia, S. W.-t. Yih (Eds.), Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 2021, pp. 390–402. URL: https://aclanthology.org/2021.emnlp-main.32/. doi:10.18653/v1/2021.emnlp-main.32.
[11] Y. Su, Z. Meng, S. Baker, N. Collier, Few-shot table-to-text generation with prototype memory, in: M.-F. Moens, X. Huang, L. Specia, S. W.-t. Yih (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2021, Association for Computational Linguistics, Punta Cana, Dominican Republic, 2021, pp. 910–917. URL: https://aclanthology.org/2021.findings-emnlp.77/. doi:10.18653/v1/2021.findings-emnlp.77.
[12] A. Siddharthan, Syntactic simplification and text cohesion, Research on Language and Computation 4 (2006) 77–109. doi:10.1007/s11168-006-9011-1.
[13] Microsoft Azure, Azure openai in foundry models, 2025. https://azure.microsoft.com/en-us/products/ai-services/openai-service [Date: 2025-07-07].
[14] Microsoft Learn, Work with chat completion models - azure openai in azure ai foundry models, 2025. https://learn.microsoft.com/en-us/azure/ai-foundry/openai/how-to/chatgpt [Date: 2025-07-07].

A. Ensemble logic

The corresponding logic is implemented in the ensemble component as shown below in Python language:

```
"""
select shortest of non-empty outputs,
tie-break by priority ['gpt','t5','rule']
"""

candidates = { 'rule': r_out, 't5': t5_out, 'gpt': gpt_out }
filtered = {k:v for k,v in candidates.items() if v.strip()}
if not filtered: return r_out

pick shortest, then by priority

best = min(
filtered.items(), key=lambda x:(len(x[1]),
```

```
                        ['gpt','t5','rule'].index(x[0]))
            )[1]
    return best
```