A Hybrid System for Comprehensive and Consistent Automated MedDRA Coding of Adverse Drug Events

Abir Naskar[1], Liuliu Chen[1], Jemima Kang[1,2], Mike Conway[1]

[1]School of Computing and Information Systems, University of Melbourne, Australia
[2]Melbourne School of Psychological Sciences, University of Melbourne, Australia
{anaskar, liuliuc, jemima.kang}@student.unimelb.edu.au, mike.conway@unimelb.edu.au

Abstract

Normalization of Adverse Drug Events (ADEs), or linking adverse event mentions to standardized dictionary terms, is crucial for harmonizing diverse clinical and patient-reported descriptions, enabling reliable aggregation, accurate signal detection, and effective pharmacovigilance across heterogeneous data sources. The ALTA 2025 shared task focuses on mapping extracted ADEs from documents to a standardized list of MedDRA phrases. This paper presents a system that combines rule-based methods, zero-shot and fine-tuned large language models (LLMs), along with prompt-based approaches using the latest commercial LLMs to address this task. Our final system achieves an Accuracy@1 score of 0.3494, ranking second on the shared task leaderboard.

1 Introduction

Adverse Drug Events (ADEs) refer to harmful or unintended medical events associated with medication use. They represent a critical concern in healthcare, motivating ongoing pharmacovigilance efforts to improve drug safety and patient outcomes (Liu and Chen, 2015). Traditional pharmacovigilance systems capture only a fraction of ADEs due to under-reporting and inconsistent data, making timely detection crucial for patient safety.

With the growth of electronic health records and social media data, Natural Language Processing (NLP) is being increasingly utilized for large-scale ADE surveillance (Golder et al., 2025). Typical ADE pipelines involve three tasks: (1) detection, (2) extraction, and (3) normalization—mapping identified ADEs to standardized vocabularies such as Medical Dictionary for Regulatory Activities (MedDRA) (Morley, 2014), which is terminology that standardizes the description of medical conditions and events. While corpora like CADEC (Karimi et al., 2015) provide valuable resources for detection and extraction, normalization remains challenging due to synonymy and lexical variation. Early systems such as MagiCoder achieved promising results but highlighted the complexity of accurate MedDRA mapping (Combi et al., 2018). Recent research in biomedical entity linking, including contrastive pre-training (e.g., KRISS) (Zhang et al., 2022) and retrieval-augmented generation (Shlyk et al., 2024), shows strong potential for improving normalization.

The Australasian Language Technology Association (ALTA) 2025 Shared Task[1] (Mollá et al., 2025) targets ADE–MedDRA normalization, where participants generate ranked MedDRA terms for pre-identified ADE mentions, evaluated by Acc@1, Acc@5, and Acc@10. By establishing this benchmark, the ALTA Shared Task addresses the need for robust, standardized ADE-normalization frameworks capable of handling noisy, user-generated text. Complementary work such as MultiADE (Dai et al., 2024) and MCN (Luo et al., 2019) further underscores the importance of domain generalization and ambiguity resolution in building scalable, real-world pharmacovigilance NLP systems.

In this shared task, the extracted Adverse Drug Events (ADEs) from user-generated texts were provided to participants. The objective was to perform normalization, i.e., mapping the extracted ADE mentions to their corresponding MedDRA (Medical Dictionary for Regulatory Activities) identifiers. We propose an end-to-end normalization pipeline that integrates rule-based methods, supervised learning approaches, and Large Language Model (LLM) prompting. Using a weighted combination of these components, the system generates a ranked list of the top 10 most relevant MedDRA terms for each ADE mention within a given document, sorted in decreasing order of similarity to the ADE phrase.

[1] https://www.alta.asn.au/events/sharedtask2025/index.html

Proceedings of The 23rd Annual Workshop of the Australasian Language Technology Association, pages 216–223

To evaluate system performance, we computed three ranking-based accuracy metrics: Acc@1, Acc@5, and Acc@10, which indicate whether the gold-standard MedDRA label appears within the top 1, 5, or 10 ranked terms, respectively. Our team achieved second place on the final test set leader board.

2 Methods

The end-to-end ADE-to-MedDRA phrase linking process is illustrated in Figure 1. Three inputs were used for this task: the complete list of MedDRA medical phrase–ID pairs (containing 74,359 unique IDs), the document text, and the extracted ADE mentions within each document that require normalization. Since the evaluation metrics include Acc@1, Acc@5, and Acc@10, our system generates the top 10 MedDRA phrases for each ADE mention, ranked in decreasing order of similarity.

We compute methods A–F to obtain the Phase 1 output, as detailed below. Subsequently, the Phase 2 output is generated based on the Phase 1 results, and the Phase 3 output builds upon the Phase 2 results. A weighted sorting approach is applied at each stage to combine and refine the ranked lists, as described in the corresponding sections.

Two types of distance functions are utilized in our approach — set-level and embedding-level — to capture both lexical and semantic similarity. To calculate the similarity between tokens, we employ a set-based similarity metric that compares two sets of tokens — the Target (T) and the Source (S). Which is, $Overlap(T, S) = \frac{|T \cap S|}{|T \cup S|}$, where, $|S|$ is cardinality of the set S. To evaluate embedding-level similarity, cosine similarity is employed to quantify the closeness between vector representations.

2.1 Finetuning LLM and prediction (Method A)

In the provided dataset, three subsets were available: training, development (dev), and test. Our automated method was developed and optimized using the training and development sets, and subsequently evaluated on the final test set. Each entry in the training and development data consists of a text document (the source text from which ADEs were extracted), the corresponding ADE mentions, and the associated MedDRA concepts. The training set contains 773 documents with a total of 4,200 ADE-to-MedDRA concept mappings, while the development set includes 161 documents and 849 ADE-to-MedDRA mappings.

Our first approach involves fine-tuning a Large Language Model (LLM) to predict the corresponding MedDRA phrase given a document and an ADE mention as input. For this purpose, we utilize MedGemma (Sellergren et al., 2025), an open medical LLM built on Gemma 3, designed for medical text and video comprehension. The prompt format used for training is presented in Table 4 in the Appendix. After the model generates a candidate phrase, we identify the closest MedDRA phrase by computing the Levenshtein distance (Yujian and Bo, 2007) between the generated phrase and all MedDRA entries, assigning a similarity score of 1 to the best-matched term.

2.2 Edit distance (Method B)

In this method, we start with the ADE phrase and compute its Levenshtein distance against all MedDRA phrases. Following this, the similarity score of the two phrases, P_1 and P_2 is calculated as: $L\_Sim(P_1, P_2) = \frac{1}{1+Levenshtein\_distance(P_1,P_2)}$. We then sort all MedDRA phrases in decreasing order based on their similarity scores and select the top 30 phrases along with their corresponding scores for further processing.

2.3 Stemmed token overlap (Method C)

This method is divided into two parts. Similar to the previous approach, our objective is to identify the top MedDRA phrases most similar to the given ADE. We first check whether two phrases, P_1 and P_2, match after converting both to lowercase and removing punctuation. If an exact match is found, we retain those phrases with a similarity value of 1. Next, we remove the terms "NOS" and "NEC" (if present) from both phrases and again check for an exact match; any matches found at this stage are also assigned a similarity value of 1.

After filtering out these exact matches, we tokenize both phrases P_1 and P_2, apply stemming to each token, and construct two sets of tokens, S_{P1} and S_{P2}. We then compute the similarity between the two sets using the $Overlap(S_{P1}, S_{P2})$ metric as defined earlier. Finally, we select the top 30 most similar MedDRA phrases based on the overlap similarity, scaled by a factor of 0.95. Next, we remove stopword tokens from both phrases and recalculate the overlap similarity, retaining the top 30 phrases based on the updated similarity values scaled by 0.9. The two sets of selected phrases

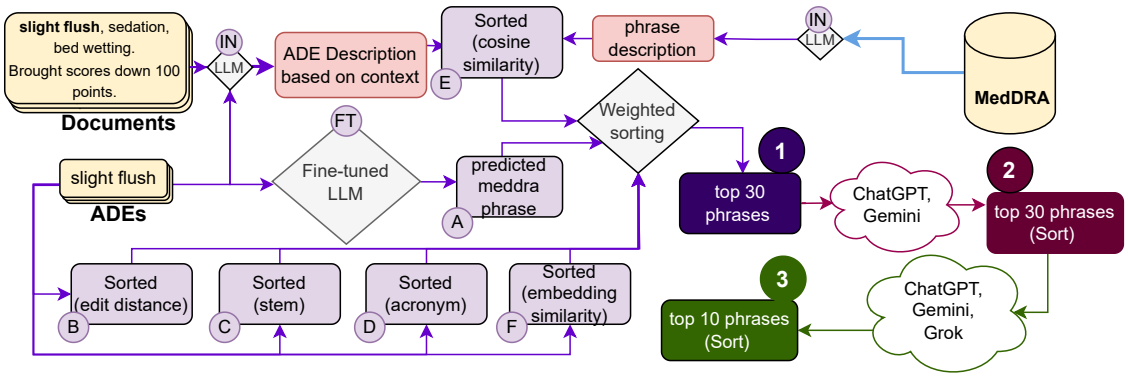

Figure 1: The full pipeline of the entire process. The detailed description is provided in Section 2.
FT: finetune, IN: inference.

are then combined, and from the merged list, we retain the final top 30 unique phrases, ensuring that each phrase appears only once. In cases of duplication, the phrase with the highest similarity score is preserved.

2.4 Acronym detection (Method D)

For acronym detection, the system identifies text segments in the source file that are written entirely in uppercase letters, with a minimum length of two and a maximum of six alphabetic characters. These acronyms may appear as standalone words, within parentheses, or adjacent to punctuation, but are not immediately preceded or followed by alphabetic characters. For example, if the source text contains "TIA's", the system extracts only "TIA". Once an acronym is identified, the system performs a match check by comparing it with the initials of MedDRA phrases that have the same number of characters as the extracted acronym. For instance, for the MedDRA phrase "Transient ischaemic attack", the corresponding acronym TIA is generated; similarly, for "Acute transient ischaemic attack", both "ATI" and "TIA" are considered. If no generated acronym matches the extracted one, the system discards it. Otherwise, it computes an overlap score based on token-level similarity, treating the acronym as a single token. For example, for the MedDRA phrase "Acute transient ischaemic attack", the tokens are ["Acute", "TIA"], and for the source phrase "TIA's", the tokens are ["TIA", "'s"]. This comparison allows the model to recognize acronym-based mentions and accurately link them to their corresponding MedDRA concepts.

2.5 Similarity of descriptions (Method E)

In this method, we first generate a short contextual description of each ADE within the document using LLM prompting, following the template provided in Table 2 in the Appendix. Similarly, we create short descriptions for each MedDRA phrase. However, since there are nearly 75,000 MedDRA phrases, generating descriptions individually would be computationally expensive. To address this, we batch the phrases (50 at a time) using the prompt shown in Table 3. This batching strategy keeps the total output length under 2,000 tokens for better generation quality, while reducing the total number of LLM calls to approximately 1,500.

After generating descriptions for both ADEs and MedDRA phrases, we compute sentence embeddings using sentence-transformers (Reimers and Gurevych, 2019). Instead of directly comparing the phrases, we calculate cosine similarity between the corresponding description embeddings. Finally, we select the top 30 most similar MedDRA phrases, scaling their similarity values by a factor of 0.9.

2.6 Similarity of embedding (Method F)

In this method, we compute the cosine similarity between the embeddings of the ADE and each MedDRA phrase. The top 30 most similar phrases are then selected based on their similarity scores, scaled by a factor of 0.9.

2.7 Phase 1 output

In this stage, we combine the outputs from Methods A to F, assigning equal weights to all. For phrases appearing in multiple methods, only the instance with the highest similarity score is retained. The results are then sorted in descending order of

similarity, and the top 30 phrases are selected as the final output of this stage.

2.8 Phase 2 output

For Phase 2, we begin by selecting the top 30 phrases from the Stage 1 output. The similarity scores are removed while preserving the original ranking order. These ranked lists, along with the corresponding document and ADE phrase, are then provided to ChatGPT and Gemini using the prompt shown in Table 5 in the Appendix. Both models return their own sorted lists. We assign positional values (like: 1, 2, 3, ...) to the elements in each of the three lists.

To combine multiple ranked outputs, we employ a rank-fusion approach. This function takes three ranked lists, a set of corresponding weights, and a default rank value (set to 30) as inputs. It first collects all unique items across the lists, then computes an average rank for each item by using its actual position if present, or the default rank if absent. The mean rank serves as the final score, and items are sorted in ascending order of this value. This consensus-based fusion promotes candidates that consistently appear higher across lists while penalizing those that are missing or ranked lower, resulting in a balanced and robust final ranking.

2.9 Phase 3 output

In Phase 3, we repeat the rank-fusion process described previously, but this time using outputs from three commercial LLMs — ChatGPT, Gemini, and Grok. The fusion function in this stage considers only the sorted lists generated by these models. The prompt is slightly modified to provide the top 15 phrases from the previous stage and to instruct the models to return the top 10 most similar MedDRA phrases in decreasing order of relevance. The aggregated and re-ranked output from this stage constitutes our final prediction.

2.10 Experiment setup

For commercial LLMs, we use GPT-4o mini[2] for prompting through ChatGPT. For prompting with Gemini, we use the free version Gemini 2.5 Flash[3]. For Grok, we employ the Grok-4-Fast-Reasoning[4] model.

For fine-tuning and inference with local LLMs, we use the MedGemma-27B Instruct-Tuned model[5] using Unsloth. The fine-tuning configuration includes a learning rate of 2e-4, batch size = 2, number of training epochs = 5, and LoRA parameters of $r = 8$, $\alpha = 8$, and dropout = 0. For inference on MedGemma, we set the temperature to 0.7.

For embedding generation, we use the Sentence-Transformers model all-MiniLM-L6-v2, a lightweight yet high-performing model widely used for computing sentence-level embeddings efficiently.

For weighted sorting, we assign equal weights (1.0) to all methods in Groups A–F to generate the Stage 1 results. In Stage 2, we combine the equally weighted sorted outputs from ChatGPT, Gemini, and the Stage 1 results (each assigned a weight of 1.0). Finally, in Stage 3, we apply equal weighting to the sorted lists produced by Gemini, ChatGPT, and Grok to obtain the final results.

In the training and development sets, a total of 4,200 and 849 connections were provided, respectively. All of these were used to fine-tune our MedGemma LLM for predicting the final test set, which contains 83 connections to be identified.

3 Results

In this task, we were allowed a maximum of three submissions. We developed three stages of results, as discussed in the previous sections. The performance metrics — Acc@1, Acc@5, and Acc@10 — were calculated on the final submission corresponding to the test set and are presented in the table below. Among the three stages, Stage 3 produced the best results. The results can be viewed on the Codabench page[6] under the team name NoviceTrio.

	Acc@1	Acc@5	Acc@10
Stage-1	0.1928	0.3976	0.4699
Stage-2	0.2891	0.6265	0.6988
Stage-3	**0.3494**	**0.6747**	**0.7229**

Table 1: Final test set result of three stages

[2] https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/
[3] https://deepmind.google/models/gemini/flash/
[4] https://x.ai/news/grok-4-fast
[5] https://huggingface.co/unsloth/medgemma-27b-text-it-unsloth-bnb-4bit
[6] https://www.codabench.org/competitions/9717/#/results-tab

4 Discussion

Due to the limited number of allowed submissions, as well as time and resource constraints, we were unable to conduct additional experiments that may have improved our system's performance. From the development phase results Table 6 in Appendix, we observed that incorporating predictions from the fine-tuned MedGemma model significantly enhanced overall accuracy (Acc@1 from 0.27 to 0.77). In this task, we applied all rule-based methods (B–F) on the provided ADE inputs; however, in future work, we plan to explore applying these same methods to the fine-tuned MedGemma (method A) outputs to potentially achieve better results. We also did not conduct an ablation study to determine which of the methods mentioned above contributed most to the performance improvement. This analysis will be explored in future work.

We observe a noticeable drop in performance during the test phase compared to the Stage 1 scores of the development phase. We suspect that the drop in performance is due to the distributional differences between the test sets of the development and test phases. As shown in Table 7, the ADE phrases in the development phase test set are quite similar to those in the training set, whereas Table 8 shows their comparison with MedDRA phrases. Although the difference is small, the development phase still has a slight advantage. From the statistics presented in Table 9 and Table 10, we can observe that the training and test sets in the development phase have quite similar characteristics. However, there is a noticeable difference between the statistics of the training set in the development phase and the test set in the test phase. Also after removing stopwords and applying stemming, 73.32% of the words in ADE phrases overlap between the training and test sets of the development phase, whereas 58.01% of the words overlap between the training set of the development phase and the test set of the test phase.

To mitigate this, we plan to use enterprise-level LLMs such as ChatGPT, Gemini, and Grok, which have stronger knowledge of MedDRA terms, ADE phrases, and clinical concepts. These models can help generate initial MedDRA phrase candidates for each ADE, potentially improving performance in Phase 1 and beyond. The domain specific databases with proper knowledge of the MedDRA phrases and medical conditions could help further as that can be retrieved and help to identify the target phrase corresponding to given by checking the relevant information or description provided in the database.

For the initial phase, we selected the top 30 most similar phrases as our starting point. To improve coverage and accuracy, future work will broaden this selection by considering the top 50 most similar phrases during and before Phase 1. These top 50 candidates will then be provided as input to Phase 2, while the top 30 most similar phrases will be used as input to Phase 3.

One of the key challenges we encountered was distinguishing between highly similar MedDRA phrases. For instance, without prior medical expertise, terms such as "Night sweats" and "Night sweat" appear almost identical, making it difficult to select the correct label. This highlights the need for greater domain insight and possibly expert-in-the-loop validation in future iterations.

Additionally, for commercial LLM-based approaches, incorporating a broader range of large language models and combining their outputs could yield a more comprehensive and robust result. We also aim to experiment with different prompting strategies such as few-shot prompting and chain-of-thought reasoning, where explicit reasoning patterns can be introduced to guide the model's understanding. These enhancements are expected to further improve the overall accuracy and interpretability of the system.

References

Carlo Combi, Margherita Zorzi, Gabriele Pozzani, Ugo Moretti, and Elena Arzenton. 2018. From narrative descriptions to meddra: automagically encoding adverse drug reactions. *Journal of Biomedical Informatics*, 84:184–199.

Xiang Dai, Sarvnaz Karimi, Abeed Sarker, Ben Hachey, and Cecile Paris. 2024. Multiade: a multi-domain benchmark for adverse drug event extraction. *Journal of Biomedical Informatics*, 160:104744.

Su Golder, Dongfang Xu, Karen O'Connor, Yunwen Wang, Mahak Batra, and Graciela Gonzalez Hernandez. 2025. Leveraging natural language processing and machine learning methods for adverse drug event detection in electronic health/medical records: a scoping review. *Drug Safety*, 48(4):321–337.

Sarvnaz Karimi, Alejandro Metke-Jimenez, Madonna Kemp, and Chen Wang. 2015. Cadec: A corpus of adverse drug event annotations. *Journal of biomedical informatics*, 55:73–81.

Xiao Liu and Hsinchun Chen. 2015. A research framework for pharmacovigilance in health social media: identification and evaluation of patient adverse drug event reports. *Journal of biomedical informatics*, 58:268–279.

Yen-Fu Luo, Weiyi Sun, and Anna Rumshisky. 2019. Mcn: a comprehensive corpus for medical concept normalization. *Journal of biomedical informatics*, 92:103132.

Diego Mollá, Xiang Dai, Sarvnaz Karimi, and Cécile Paris. 2025. Overview of the 2025 ALTA Shared Task: Normalise adverse drug events. In *Proceedings of the 2025 Australasian Language Technology Workshop (ALTA 2025)*, Sydney, Australia.

Greg Morley. 2014. Adverse event reporting: a brief overview of meddra. *Medical Writing*, 23(2):113–116.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Andrew Sellergren, Sahar Kazemzadeh, Tiam Jaroensri, Atilla Kiraly, Madeleine Traverse, Timo Kohlberger, Shawn Xu, Fayaz Jamil, Cían Hughes, Charles Lau, Justin Chen, Fereshteh Mahvar, Liron Yatziv, Tiffany Chen, Bram Sterling, Stefanie Anna Baby, Susanna Maria Baby, Jeremy Lai, Samuel Schmidgall, and 62 others. 2025. Medgemma technical report. *Preprint*, arXiv:2507.05201.

Darya Shlyk, Tudor Groza, Marco Mesiti, Stefano Montanelli, and Emanuele Cavalleri. 2024. REAL: A retrieval-augmented entity linking approach for biomedical concept recognition. In *Proceedings of the 23rd Workshop on Biomedical Natural Language Processing*, pages 380–389, Bangkok, Thailand. Association for Computational Linguistics.

Kaiyang Wan, Honglin Mu, Rui Hao, Haoran Luo, Tianle Gu, and Xiuying Chen. 2025. A cognitive writing perspective for constrained long-form text generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 9832–9844, Vienna, Austria. Association for Computational Linguistics.

Li Yujian and Liu Bo. 2007. A normalized levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1091–1095.

Sheng Zhang, Hao Cheng, Shikhar Vashishth, Cliff Wong, Jinfeng Xiao, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2022. Knowledge-rich self-supervision for biomedical entity linking. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 868–880, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

A Appendix

A.1 Prompt for description generation

Table 2 presents the prompt used for generating the description of an ADE phrase in relation to its context (i.e., the document from which the ADE was extracted). This prompt is utilized in MedGemma.

Prompt for description generation
<start_of_turn>user You are a helpful medical assistant. Read the following document and provide a short, medically accurate one-line description for the phrase **{phrase}** in the context of this document. Document: **{document}** Description: <end_of_turn> <start_of_turn>model

Table 2: A zero-shot prompt that asks MedGemma to produce a concise, context-aware description of a medical-condition phrase appearing in a document.

A.2 Prompt for generating MedDRA phrase description

Table 3 presents the prompt we use to generate descriptions for each MedDRA phrase. Unlike the previous prompt, this one does not include any context and relies solely on the MedDRA phrases. The same MedGemma model is used for prompting. Given that there are approximately 75,000 phrases, calling the model individually for each phrase would be extremely time-consuming and costly. To mitigate this, we process phrases in batches of 50, reducing the overall time by a factor of 50. Since generating very long text can degrade the quality of output (Wan et al., 2025), we set a maximum output token limit of 2,000 and instruct the LLM to keep each description concise (12–15 words).

Prompt for generating MedDRA phrase description
<start_of_turn>user You are a helpful medical assistant. For each of the following MedDRA phrases, provide a one-line short description. Keep each answer concise (max 12–15 words) and medically accurate. Example: 1. Hypertension: Persistently high blood pressure. 2. Migraine: Recurrent severe headache often with nausea or light sensitivity. 3. Asthma: Chronic airway inflammation causing wheezing and breathlessness. Now answer for these phrases:**{phrase_list}** Format strictly as: 1. phrase: description 2. phrase: description ... and so on. <end_of_turn> <start_of_turn>model

Table 3: Few-shot prompt for Generating description of MedDRA phrases in batch mode. In {phrase_list} we are providing the list of 50 MedDRA phrases.

A.3 Prompt for finetuning LLM

Table 4 presents the prompt used for fine-tuning and inference with MedGemma. The objective is to generate the target MedDRA phrase corresponding to the ADE phrase related to the context of the provided document.

Prompt for finetuning LLM
<start_of_turn>user You are a helpful medical assistant. Which MedDRA phrase best match with, **{phrase}** In the context of the following document, **{doc}** Only return that MedDRA phrase. <end_of_turn> <start_of_turn>model

Table 4: Prompt for fine-tuning MedGemma to predict MedDRA phrases.

A.4 Prompt for sorting phrases using commercial LLMs

We use ChatGPT, Grok, and Gemini to rank a given list of MedDRA phrases based on their similarity to an ADE phrase within the context of a document. The prompt used for this task is provided in Table 5.

A.5 Results Obtained During the Development Phase

Table 6 presents the results obtained during the development phase. In Phase 1, two main approaches were employed: (i) Methods B–F, as described in the Methods section, which do not utilize the given target MedDRA phrase; and (ii) a fine-tuning-based approach that leverages labeled ADE–MedDRA pairs. For ranking results, the first approach corresponds to the Stage 1 method, which excludes fine-tuning.

A.6 Comparison of different similarity during development and test

In Table 7 and Table 8, we present similarity values comparing ADE phrases from the test sets of the development and test phases with the ADE phrases from the training set and the MedDRA phrase set. Four different metrics are used to compute the similarity. The first metric is exact match, which indicates whether the phrases are identical; we report the percentage of exact matches. The second metric, edit distance similarity, represents

Prompt for sorting phrases using commercial LLMs
You are given a document (D) and a medical condition phrase (P). Your task is to sort the given list of MedDRA phrases based on their similarity to the phrase (P) in the context of document (D). Sort the phrases in decreasing order of similarity. Input: Document (D): {document} Phrase (P): {phrase} List (condition, id): {list of phrases} Output: Return only the sorted phrases in Python list format.

Table 5: Prompt for sorting list of MedDRA phrases based on similarity with a given document and a target condition phrase.

	Acc@1	Acc@5	Acc@10
Method B-F	0.2717	0.4033	0.4634
Stage 1	0.7722	0.7996	0.8077

Table 6: Development phase results: (1) Method B–F, (2) Stage 1, mentioned in Section 2

	Development phase	Test phase
Exact	0.312	0
Edit-distance	0.785	0.453
Embedding	0.905	0.605
Lexical overlap	0.447	0.065

Table 7: Comparison of ADE phrases in the test sets of the development and test phases with the training ADE phrase set

	Development phase	Test phase
Exact	0.053	0
Edit-distance	0.493	0.436
Embedding	0.442	0.306
Lexical overlap	0.079	0.012

Table 8: Comparison of ADE phrases in the test sets of the development and test phases with the MedDRA phrase set

the mean Levenshtein ratio, a measure that quantifies the similarity between two strings based on their Levenshtein distance, computed for the best-matching pairs. The third metric uses sentence embeddings generated by the sentence-transformers model "all-MiniLM-L6-v2" to identify the best phrase matches and then calculates the mean cosine similarity across all matched pairs. The fourth metric is lexical overlap, computed as the word overlap after removing stopwords and applying stemming. Each similarity value is computed using the best individual match, and the mean value is then reported.

In Table 7, the given ADE phrases are compared in two ways: (1) between the test and training sets of the development phase (first column), and (2) between the training set of the development phase and the test set of the test phase (second column). Table 8 presents the same comparisons, except that the training set of the development phase is replaced with the complete MedDRA phrase set.

A.7 Document and ADE phrase statistics

Table 9 presents the document-level statistics, including the average number of words and characters per document. Similarly, Table 10 provides the corresponding statistics for ADE phrases. In these tables, $Train_{dev}$ refers to the training set of the development phase, $Test_{dev}$ denotes the test set of the development phase, and $Test_{test}$ represents the test set of the test phase.

	Avg words per doc	**Avg chars per doc**
$Train_{dev}$	118.839048	659.328810
$Test_{dev}$	116.762621	648.512352
$Test_{test}$	147.807229	803.590361

Table 9: Statistics of documents

	Avg words per ADE	**Avg chars per ADE**
$Train_{dev}$	2.595952	15.34952
$Test_{dev}$	2.58754	15.64877
$Test_{test}$	4.084337	23.73494

Table 10: Statistics of ADE phrases