Beyond Base Predictors: Using LLMs to Resolve Ambiguities in Akkadian Lemmatization

Frederick Riemenschneider
Dept. of Computational Linguistics
Heidelberg University, Germany
riemenschneider@cl.uni-heidelberg.de

Abstract

We present a hybrid approach for Akkadian lemmatization in the EvaCun 2025 Shared Task that combines traditional NLP techniques with large language models (LLMs). Our system employs three Base Predictors–a dictionary lookup and two T5 models–to establish initial lemma candidates. For cases where these predictors disagree (18.72% of instances), we implement an LLM Resolution module, enhanced with direct access to the electronic Babylonian Library (eBL) dictionary entries. This module includes a Predictor component that generates initial lemma predictions based on dictionary information, and a Validator component that refines these predictions through contextual reasoning. Error analysis reveals that the system struggles most with small differences (like capitalization) and certain ambiguous logograms (like BI). Our work demonstrates the benefits of combining traditional NLP approaches with the reasoning capabilities of LLMs when provided with appropriate domain knowledge.

1 Introduction

Akkadian lemmatization presents significant challenges due to complex morphology, logographic elements, and varying scholarly conventions. Despite significant advances in neural lemmatization approaches (Sahala and Lindén, 2023), ambiguities continue to resist automated resolution. The EvaCun 2025 shared task explores how LLMs can be integrated into lemmatization workflows for Akkadian texts. In this paper, we present our hybrid approach that strategically combines traditional NLP techniques with LLM capabilities. Our system is motivated by the observation that while most Akkadian forms can be reliably lemmatized through conventional methods, a small but significant percentage requires deeper analysis. We leverage this finding by directing our LLM resources specifically toward resolving these difficult cases.

2 Related Work

Prior work on Akkadian Lemmatization and neighboring tasks primarily relies on rule-based systems. For instance, Kataja and Koskenniemi (1988) use finite-state transducers to analyze Akkadian morphology, while Macks (2002) employs a Prolog Definite Clause Grammar. Particularly significant in practice is L2 (Tinney, 2019), a dictionary-based tool that has been used to annotate the Open Richly Annotated Cuneiform Corpus (Oracc).[1]

More recently, Sahala et al. (2020) explore finite-state approaches to Ancient Babylonian through their BabyFST model, highlighting the challenges posed by word form ambiguity. The field has since advanced to neural approaches with BabyLemmatizer (Sahala et al., 2022) and its successor BabyLemmatizer 2.0 (Sahala and Lindén, 2023), which represent the current state of the art.

Beyond the Akkadian-focused systems described above, broader lemmatization research in recent shared tasks has established sequence-to-sequence modeling as an effective approach across multiple languages (Wróbel and Nowak, 2022; Yangarber et al., 2023; Riemenschneider and Krahn, 2024). We incorporate this proven methodology while investigating how LLMs can extend and enhance these techniques for Akkadian specifically.

3 System Architecture

In this section, we present our lemmatization system architecture, illustrated in Figure 1. Our approach implements a hierarchical approach wherein base predictors handle standard cases, while LLM components address more challenging instances.

3.1 Input

The input layer of our lemmatization pipeline accepts two primary data elements: the token of interest to be lemmatized and the full fragment in

[1] https://oracc.museum.upenn.edu/index.html.

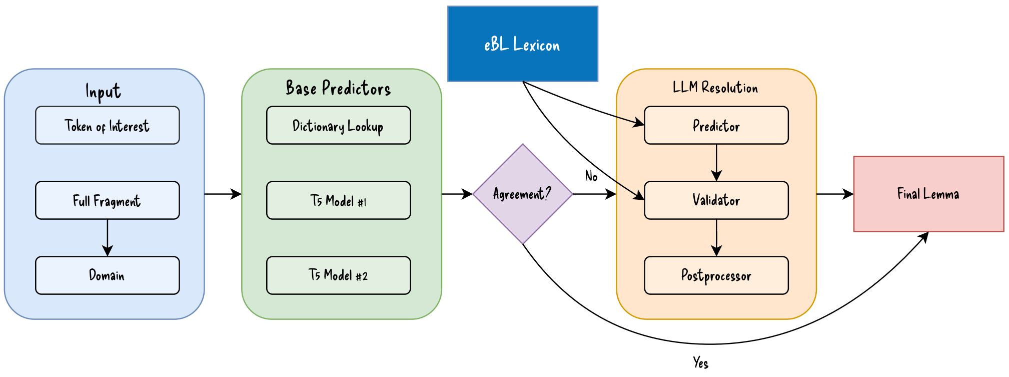

Figure 1: Overview of our system architecture.

which this token appears. In addition to providing contextual information, the full fragment serves an important technical purpose: it enables our system to detect which lemmatization convention should be applied to the output.

Domain Detection. An initial analysis of the training data reveals a critical distinction between two different lemmatization conventions that impact how a word should be lemmatized. The dataset contains a mixture of texts following either the Archibab (Charpin, 2014)[2] or the eBL[3] format. For example, "lord" appears as "bêlum" in Archibab (with circumflex accent and mimation) but as "bēlu I" in eBL (with macron, no mimation, and Roman indexing). The transliterated text follows different standards as well: while the data belonging to eBL uses curly brackets for determinatives, Archibab employs parentheses. Moreover, while eBL consistently uses lowerscript indices to distinguish homophones, Archibab uses the acute and grave accents for the indices 2 and 3 (e.g., "u two" is written as "u_2" in eBL and as "ú" in Archibab).

To address this challenge, our first processing step determines whether to follow the eBL or Archibab lemmatization format by analyzing the full fragment context. We develop a rule-based domain detection method that determines the target format by analyzing the input format. For example, a transliterated fragment containing parenthesized determinatives and acute accents (e.g., "[...] i-na-an-na ma-ar-ta be-lí-ia (I)-(munus)-ki-ru-ú ša [...]" would be identified as Archibab-style, indicating that "be-lí-ia" should be lemmatized as "bêlum". Conversely, a fragment with subscript numbers and curly braces (e.g., "[...] šu-pi $ša_2$ be-li_2-ia lu ṭa-a-bi [...]") would be classified as eBL-style, signaling that "be-li_2-ia" should be lemmatized as "bēlu I".

[2]https://www.archibab.fr/home.
[3]https://www.ebl.lmu.de/.

3.2 Base Predictors

Our system is built on the observation that lemmatization difficulty varies across tokens, with only a subset requiring complex disambiguation. We therefore leverage a dictionary lookup and T5 models to handle most of the predictions.

Dictionary Lookup. When splitting the training data into 95% train and 5% validation data, a simple dictionary lookup already achieves 77.63% accuracy on the validation set. This baseline approach is further improved with a domain-aware dictionary lookup, which reaches 82.63% accuracy. Our domain-aware implementation maintains separate dictionaries for eBL and Archibab; when a token cannot be found in the domain-specific dictionary, the system falls back to a merged dictionary containing entries from both domains.

T5 Models. In recent shared tasks, treating lemmatization as a sequence-to-sequence task has proven successful (Wróbel and Nowak, 2022; Yangarber et al., 2023; Riemenschneider and Krahn, 2024). Following this established approach, we pre-train our own Akkadian T5 model on the transliterated texts provided by the task organizers, as additional data was not permitted under the competition rules. Specifically, we train a $T5_{base}$ model for 100 epochs using nanoT5 (Nawrot, 2023). We fine-tune this pre-trained model twice, each time with a different 5% held-out validation split, continuing until we observe no improvements in lemmatization accuracy for five consecutive evaluation runs.

The input format for both models consists of a domain token followed by a window of three tokens before and after the target, with special tokens delimiting the target: For instance, "[DOMAIN=eBL] BI ip-pal-si-hu ina [special_token_0] MUL.MUL [special_token_1] u {d}30 IGI-šu$_2$-nu-ti-ma" should be lemmatized as "zappu I". Despite this contextualized representation, our two T5 models achieve only 86.96% and 88.25% validation accuracy respectively, barely outperforming the dictionary-based approach and showing limited generalization to unseen forms.

We utilize model disagreement as a signal for identifying challenging lemmatization decisions. When at least two of our three base models produce different predictions for a given token, we invoke a LLM resolution strategy. This targeted approach allows us to process the majority of cases efficiently: 81.28% of the test set was lemmatized using only our base predictors, while the more sophisticated LLM resolution was reserved for the remaining 18.72% of challenging cases.

3.3 LLM Resolution

The difficult cases requiring LLM intervention present a dual challenge: on one hand, they require reasoning to determine the correct lemma. On the other hand, we are simultaneously confronted with arbitrary lexicographic conventions that cannot be derived through reasoning alone. For instance, "kasāpu" is a homonym that can mean either "to break (into bits)" or "to make funerary offering", but even after correctly determining the lemma and its meaning, the assignment of Roman indices (I for "break" versus II for "funerary offering") follows arbitrary conventions rather than linguistic principles.

To address this challenge, we prepare the LLM input text with comprehensive contextual information: the full textual fragment, the target token, predictions from all base models, and–importantly–the corresponding dictionary definitions retrieved from the eBL lexicon. Additionally, we augment this information with up to three representative examples from the same textual domain, selected first from exact matches of the target token and then supplemented with similar forms (based on Levenshtein distance) when necessary. We ensure diversity in our examples by including only one example per lemma, thereby presenting three different lemmata to the model. This approach provides information on how lemmatization conventions are structured within the specific domain.

The enriched context supports a two-stage LLM resolution process. First, an LLM Predictor analyzes the available information to generate an initial lemma prediction. Then, a second LLM instance serves as a Validator, reviewing both the original context and the predictor's reasoning to make the final determination. In our implementation, we use Anthropic's `claude-3-7-sonnet-20250219`[4] as both Predictor and Validator.

LLM Predictor. The Predictor component leverages the LLM's reasoning abilities while bridging the gap to arbitrary conventions through the selected contextual information. By analyzing the full fragment, the dictionary definitions, and considering domain-specific examples, the Predictor generates both a reasoned explanation and a lemma prediction.

LLM Validator. The Validator component receives all the information provided to the Predictor, along with the Predictor's reasoning and lemma choice, as well as the dictionary definition of the Predictor's proposed lemma. This second-stage verification ensures that even when the correct lemma was not present in the base model predictions, it can still be properly evaluated against the eBL lexicon. The Validator considers all available evidence to make the final determination, serving as a safeguard against potential errors in the Predictor's analysis.

Postprocessing. The resulting LLM-based prediction system usually follows the domain-specific standards, but exhibits a bias toward eBL conventions due to our reliance on the eBL lexicon. To ensure domain-appropriate outputs, we apply targeted post-processing rules that adapt the lemmata to their domains. For instance, in the Archibab texts, we remove Roman indices and replace macrons with circumflexes (converting ā, ē, ī, ū to â, ê, î, û). Conversely, for eBL texts, we verify the presence of required Roman indices, falling back to the closest base model prediction when necessary. This post-processing ensures that our final lemmata adhere to the expected conventional standards of each domain.

4 Error Analysis

Base Model Performance. Our system architecture enables a systematic analysis of error patterns at each stage of the prediction pipeline. We begin

[4]https://www.anthropic.com/news/claude-3-7-sonnet.

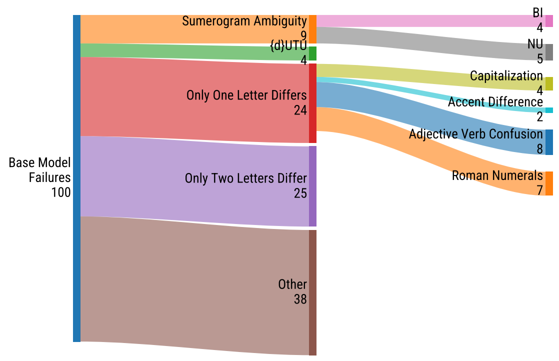

Figure 2: Base Model Failures Despite Agreement (%).

by examining the performance of our base predictors in cases where they unanimously agree, thereby bypassing the need for LLM intervention. Across the two independent train/validation splits, we observe that when both the dictionary lookup and T5 model agree on a lemma prediction, their combined accuracy reaches 97.2% ± 0.04%. This analysis validates our architectural decision to avoid LLM processing when base predictors reach consensus, as the error rate in these cases consistently remains below 3%. Although we cannot directly measure the error rate when all three systems (dictionary and both T5 models) agree due to training data overlap, we can reasonably expect this error to be even lower than the observed 2.8%.

We present an analysis of the 2.8% of cases where base models fail despite agreement in Figure 2. The logogram NU represents the most frequently mispredicted form (5% of all errors), where the system consistently predicts "lā I" instead of the contextually appropriate "lu I" or "ṣalmu II". This pattern reveals a fundamental limitation in the models' ability to disambiguate logogram based on context, highlighting a specific area where targeted improvements could yield significant gains in overall system performance. (ii) Similarly, the logogram BI (3.65% of errors) demonstrates limitations in grammatical reasoning, with the system defaulting to "šū I" instead of contextually appropriate alternatives like "šuāti I" or "šī I".

(iii) The divine name {d}UTU represents the second most frequently mispredicted form (4.2% of errors), where our system consistently predicts "Šamaš I" while the gold standard sometimes requires "šamšu I". This distinction is particularly subtle, as it is often the case that both lemmata are valid, depending on contextual interpretation.

(iv) The largest category of systematically identifiable errors (24.22%) involves cases where predictions differ from gold standards by only a single letter. Within this category, capitalization accounts for 4% of all errors, where the system correctly identifies the lemma but uses incorrect capitalization (e.g., "šarru I" vs. "Šarru I"). Accent differences account for 1.92% of errors. Arguably, lemmata derived from the same root, e.g., "ṣalāmu I" and "ṣalmu I" both being valid lemmata for GE$_6$, also belong in this category as they represent minor variations of the same lexical concept. These "near-miss" errors suggest that a substantial portion of the system's failures involve formatting variations rather than fundamental misunderstandings of the underlying lexical items. Roman index errors represent a more significant issue (7.32% of all errors), where the system identifies the correct lemma but assigns an incorrect index.

LLM Resolution Performance. To analyze the performance of our LLM Resolution approach, we examine 500 instances where Base Predictors disagree on the validation data. The overall accuracy on these challenging cases is 77.4%. The LLM Predictor contributes significantly to this performance, correctly resolving 73.6% of cases. The Validator module further improves results by correctly resolving an additional 4% of cases, though it occasionally introduces errors (0.2% of cases). Post-editing rules correct another 1.4% of cases.

Figure 3 presents an error analysis of the remaining cases where our approach fails. The analysis reveals that Roman Indices are particularly well-handled through the LLM's reasoning capabilities, as the model can effectively leverage dictionary entries to reach correct conclusions. Similarly, almost all cases involving the logogram NU are successfully resolved. The logogram BI remains more challenging, with our system failing in 36.37% of all cases involving this logogram.

We hypothesize that these cases are not a limitation of the LLM but rather due to a peculiarity in Akkadian lemmatization, where case and gender variations may require distinct lemma entries (e.g., "šī I" or "šuāti I"). Given that personal pronouns and common logograms like BI appear frequently in the corpus, improving the model's handling of these cases could enhance overall performance. Future improvements could include explicitly instructing the model about the specific lemmatization conventions for these special cases.

Additionally, our analysis reveals several previ-

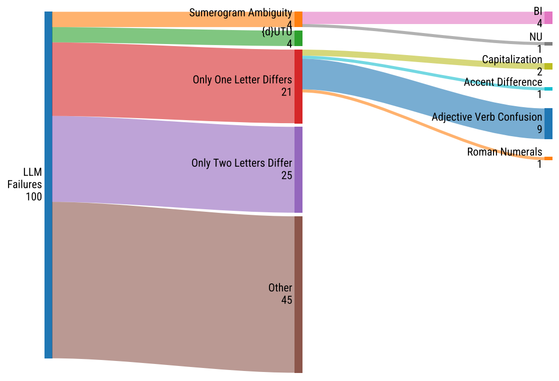

Figure 3: LLM Model Failures (%).

ously unclassified cases of Sumerogram Ambiguity that become apparent as other error types are resolved. These cases are categorized as "Other" in Figure 3 and are beyond the scope of our current qualitative analysis.

5 Outlook

Our system was designed specifically for the EvaCun 2025 Shared Task, utilizing only the data provided by the task organizers. In a real-world implementation scenario, several enhancements could substantially improve performance.

A straightforward improvement involves implementing more powerful base predictors. For instance, adapting the BabyLemmatizer or pretraining a more comprehensive T5 model on the full corpus of available digitized Akkadian texts–rather than being limited to the task-provided data–would establish a stronger foundation for the entire system. The BabyLemmatizer has demonstrated accuracy rates of approximately 95% (albeit on different datasets), indicating significant potential improvements for base lemmatizers that could be seamlessly integrated into our architecture.

Another limitation of our current approach stems from working with an LLM with closed weights. As it is not possible to fine-tune Claude, we invested considerable effort in designing prompts that would guide the model to produce appropriately formatted output, creating computational overhead during inference. Parameter-efficient fine-tuning methods such as LoRA (Hu et al., 2022) could potentially streamline this process by teaching the model expected output formats directly. Unfortunately, our initial experiments with Qwen (Yang et al., 2024) and Gemma (Riviere et al., 2024)–models offering accessible weights–yielded suboptimal results, likely due to their limited capabilities with Akkadian language processing.

6 Conclusion

We present our system for the EvaCun 2025 Shared Task on lemmatization, which combines traditional NLP approaches with LLM capabilities. Our approach consists of three Base Predictors–a dictionary lookup and two T5 models–augmented with an LLM module that resolves difficult cases, directly accessing the eBL dictionary entries.

In our analysis, we demonstrate that while the Base Predictors achieve already high performance, the LLM Module significantly enhances results by resolving challenging cases. The error analysis reveals that our approach is particularly effective at handling Roman Indices, where the LLM's ability to reason over dictionary entries proves valuable, as it can effectively disambiguate between lemmata by leveraging contextual clues and domain knowledge.

Our work highlights an important methodological consideration in applying AI to specialized linguistic tasks: the trade-off between fine-tuning and prompting approaches. Fine-tuning language models offers the advantage of domain adaptation but risks overfitting to biases present in limited training data. In contrast, prompting LLMs as demonstrated in our approach preserves their general reasoning capabilities but presents challenges in precisely controlling output format and applying domain-specific conventions. While our system successfully augmented the LLM with demonstrations and dictionary entries, future work could benefit from more structured guidance through explicit instructions about lemmatization conventions, particularly for edge cases involving case variations, Sumerian logograms, and personal pronouns.

Acknowledgements

We thank the anonymous reviewers for their valuable suggestions.

References

Dominique Charpin. 2014. The Assyriologist and the Computer. The "Archibab" Project. *Hebrew Bible and Ancient Israel*, 3(1):137–153.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Laura Kataja and Kimmo Koskenniemi. 1988. Finite-state description of Semitic morphology: A case study of ancient Accadian. In *Coling Budapest 1988 Volume 1: International Conference on Computational Linguistics*.

Aaron Macks. 2002. Parsing Akkadian verbs with Prolog. In *Proceedings of the ACL-02 Workshop on Computational Approaches to Semitic Languages*, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Piotr Nawrot. 2023. nanoT5: Fast & simple pre-training and fine-tuning of t5 models with limited resources. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, pages 95–101, Singapore. Association for Computational Linguistics.

Frederick Riemenschneider and Kevin Krahn. 2024. Heidelberg-boston @ SIGTYP 2024 shared task: Enhancing low-resource language analysis with character-aware hierarchical transformers. In *Proceedings of the 6th Workshop on Research in Computational Linguistic Typology and Multilingual NLP*, pages 131–141, St. Julian's, Malta. Association for Computational Linguistics.

Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Aleksi Sahala, Tero Alstola, Jonathan Valk, and Krister Linden. 2022. Babylemmatizer: A lemmatizer and pos-tagger for akkadian. In *CLARIN Annual Conference Proceedings, 2022*, CLARIN Annual Conference Proceedings, page 14–18, Netherlands. CLARIN ERIC. CLARIN Annual Conference ; Conference date: 10-10-2022 Through 12-10-2022.

Aleksi Sahala and Krister Lindén. 2023. A neural pipeline for POS-tagging and lemmatizing cuneiform languages. In *Proceedings of the Ancient Language Processing Workshop*, pages 203–212, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

Aleksi Sahala, Miikka Silfverberg, Antti Arppe, and Krister Lindén. 2020. BabyFST - towards a finite-state based computational model of ancient Babylonian. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 3886–3894, Marseille, France. European Language Resources Association.

Steve Tinney. 2019. L2: How it works. Oracc: The Open Richly Annotated Cuneiform Corpus.

Krzysztof Wróbel and Krzysztof Nowak. 2022. Transformer-based part-of-speech tagging and lemmatization for Latin. In *Proceedings of the Second Workshop on Language Technologies for Historical and Ancient Languages*, pages 193–197, Marseille, France. European Language Resources Association.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Roman Yangarber, Jakub Piskorski, Anna Dmitrieva, Michał Marcińczuk, Pavel Přibáň, Piotr Rybak, and Josef Steinberger. 2023. Slav-NER: the 4th cross-lingual challenge on recognition, normalization, classification, and linking of named entities across Slavic languages. In *Proceedings of the 9th Workshop on Slavic Natural Language Processing 2023 (SlavicNLP 2023)*, pages 179–189, Dubrovnik, Croatia. Association for Computational Linguistics.