Validator-Guided Hard Negative Mining for Masked Language Modeling in Low-Resource Ancient Languages

Andrei Voinea
Faculty of Mathematics and Computer Science
Babeș-Bolyai University
andrei.voinea@stud.ubbcluj.ro

Abstract

Masked language modeling for low-resource ancient languages remains challenging because pre-trained multilingual models lack exposure to these languages. We investigate rule-based linguistic constraints and hard negative mining for Sumerian, a language isolate not included in multilingual BERT's training data. We build a hierarchical validator capturing subword, word, and part-of-speech patterns from 4,545 annotated sequences, using it to filter candidates and identify hard negatives for fine-tuning. Vanilla mBERT achieves 18.0% hit@10 accuracy. The validator alone improves this to 72.8%, while hard negative fine-tuning reaches 78.3%. Combining both yields 86.7%, a 68.7 percentage point improvement. Temporal generalization evaluation on tablets from 600 years earlier shows that both the hard negative mining and the validator alone improve performance, but the combined approach underperforms due to the validator's period specific rules. These findings demonstrate that hard negative mining transfers across periods while explicit rule-based constraints provide strong in-domain improvements but limited cross-period generalization.

1 Introduction

Sumerian, a language isolate spoken in ancient Mesopotamia over 5,000 years ago, is one of humanity's earliest written languages and a critical window into ancient civilization (Pagé-Perron et al., 2017). Despite the existence of over 100,000 digitized Sumerian texts, computational processing of this language remains challenging due to severe resource constraints, complex agglutinative morphology, and the fragmentary nature of clay tablet inscriptions (Chiarcos et al., 2018). Reconstructing damaged or partially illegible Sumerian texts is essential for our understanding of ancient administrative, legal, and literary documents.

Recent advances in natural language processing have begun to address Sumerian text analysis, including machine translation systems (Pagé-Perron et al., 2017; Punia et al., 2020), morphological annotation frameworks (Chiarcos et al., 2018), and shared tasks on lemmatization and token prediction (Gordin et al., 2025). However, pre-trained multilingual models, while powerful for high-resource languages, perform poorly in this setting. Sumerian is not among the languages seen during pre-training, and crucially, it is a language isolate (no known related languages exist) which severely limits the cross-lingual transfer that multilingual models typically rely on (Wu and Dredze, 2020). On our Sumerian masked language modeling (MLM) task, vanilla multilingual BERT (mBERT) achieves only 18.0% hit@10 accuracy, far below the performance such models achieve on modern languages. This raises the question of whether explicit linguistic knowledge, extracted from limited annotations, can serve as an effective inductive bias to guide neural language models in such an extreme low-resource scenario.

In this paper, we investigate this question in the context of masked language modeling for Sumerian text reconstruction. We evaluate a pipeline that combines a hierarchical rule-based validator with hard negative mining to fine-tune mBERT. The validator learns subword, word-level, and part-of-speech patterns from a small POS-annotated corpus of 4,545 sequences (extracted from 296 tablets) and serves a dual role: it constrains the candidate space during prediction, and it identifies linguistically valid but semantically incorrect tokens as hard negatives for fine-tuning. Our experiments show that the two components are strongly complementary. The validator alone improves hit@10 accuracy from 18.0% to 72.8%, a 54.8 percentage point gain achieved purely through rule-based filtering. Hard negative fine-tuning brings mBERT to 78.3% without the validator. Combining both yields 86.7% hit@10 accuracy, a 68.7 percentage

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop), pages 779–790

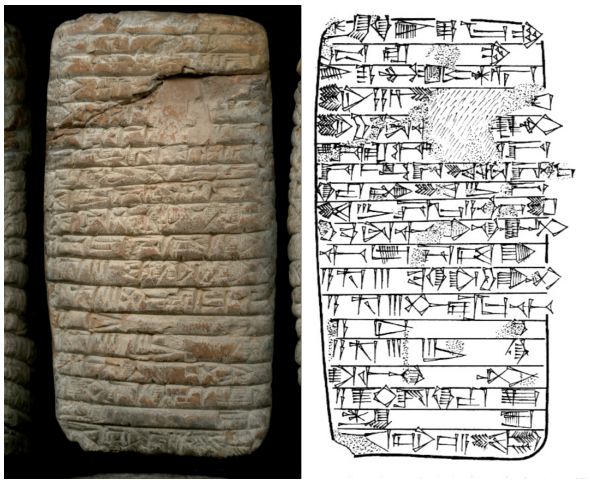

Figure 1: Sumerian tablet from CDLI and line art

point improvement over the vanilla baseline. Temporal generalization evaluation on tablets from 600 years earlier reveals that while hard negative mining maintains robust performance, the validator's morphosyntactic rules are period-specific, demonstrating domain-dependent trade-offs between optimization and generalization. These results suggest that explicit linguistic constraints remain a powerful tool for neural language modeling when annotation is scarce but domain expertise is available.

2 Data

We evaluate our approach on Sumerian cuneiform texts from the Cuneiform Digital Library Initiative (CDLI). This section describes the linguistic properties of Sumerian, the structure of the CDLI corpus, and the specific dataset used in our experiments.

2.1 Sumerian and the CDLI Corpus

Sumerian is a language isolate, no known related languages have been identified, spoken in ancient Mesopotamia from at least 3200 BCE until approximately 2000 BCE (Thomsen, 1984). The lack of related languages is particularly consequential for modern NLP: unlike low-resource languages with living relatives (e.g. Scots Gaelic and Irish), Sumerian cannot benefit from cross-lingual transfer learning, a key mechanism by which multilingual models like mBERT generalize (Wu and Dredze, 2020). Typologically, Sumerian is classified as an agglutinative language with split ergativity and subject-object-verb (SOV) word order (Thomsen, 1984).

Tablets in CDLI are documented using the ASCII Transliteration Format (ATF), a standardized plain-text notation system that represents cuneiform signs using ASCII characters. Each tablet record includes transliteration, metadata (provenance, period, genre, collection), and when available, digital images and hand-drawn copies. Figure 1 shows an example tablet photograph alongside its hand-drawn lineart transcription. The ATF format example in A represents a record for tablet P100004. Key elements of the format include: tablet ID (`&P100004`), language declaration (`#atf: lang sux` for Sumerian), physical structure markers (`@tablet`, `@obverse`, `@reverse`), determinatives in curly braces (`{d}` for divinity, `{ki}` for place names)—semantic classifiers written in the original cuneiform but represented using curly brace notation in ATF by modern transcribers—collation corrections (`!`), editorial insertions (`<...>`), and uncertain readings (`?`).

2.2 Dataset Construction and Split

While the CDLI corpus contains over 143,000 Sumerian tablets, theoretically (CDLI Contributors, 2026) sufficient for establishing a large-scale training dataset, our approach requires tablets with Part-of-Speech (POS) tag sequences to train the validator component (described in Section 3.2). This requirement significantly constrains the available data: of the 143,078 Sumerian tablets in CDLI, only 370 include POS annotations at the time of the experiments.

We tokenize all 370 tablets using the BERT Fast Tokenizer from the `bert-base-multilingual-cased` model (Devlin et al., 2019). This produces subword tokens following BERT's WordPiece tokenization, where continuation tokens are marked with the `##` prefix. For example, the Sumerian phrase `mu {d} szu-{d} suen lugal` is tokenized as:

```
mu { d } sz ##u - { d } sue ##n lu ##gal
```

After tokenization, the 370 tablets yield 5,366 sequences (individual lines of text from the tablets). We split the data using an 85-10-5 train-validation-test. Test split *by tablet* (not by sequence) to prevent data leakage, all sequences from a given tablet appear in exactly one split. This produces:

- **Training set**: 314 tablets, 4,545 sequences
- **Validation set**: 37 tablets, 568 sequences
- **Test set**: 19 tablets, 253 sequences

The training set is used to extract validator patterns (Section 3.2) and to mine hard negatives for fine-tuning (Section 3.3). The validation set is used for hyperparameter selection (e.g., the weighting coefficient α for combining validator and BERT scores, and validator confidence thresholds). The test set is held out for final evaluation and is never seen during training or hyperparameter tuning.

Vocabulary Statistics The training corpus contains 1,847 unique tokens after BERT tokenization, with an average sequence length of 12.3 tokens (median: 10 tokens, 90th percentile: 22 tokens). The vocabulary includes both starter tokens (e.g., `mu`, `lugal`) and continuation tokens (e.g., `##gal`, `##n`). The mBERT vocabulary contains approximately 119,000 tokens total, meaning Sumerian-specific tokens represent less than 2% of the full vocabulary.

3 Method

We combine linguistic rule-based validation with neural language modeling through validator training, hard negative mining, and complementary prediction at inference. The validator learns token patterns from annotated data, guides mBERT fine-tuning through hard negatives, then constrains the inference search space while the fine-tuned model provides semantic ranking.

3.1 Overview

The pipeline consists of three phases. First, we train a hierarchical validator on POS-annotated data to learn the morphological and syntactic constraints. Second, we use this validator to mine hard negatives, tokens that are linguistically valid but semantically incorrect in context. Third, we fine-tune mBERT on these hard negatives to improve semantic disambiguation among grammatically acceptable alternatives. Figure 2 illustrates the cascading validator.

3.2 Hierarchical Validator

The validator implements three-level cascade filtering: subword, word, and POS levels. Each level assigns confidence scores to candidates based on patterns observed in training data.

Subword level examines token sequences at morpheme boundaries. For candidate c with context $(w_{\text{prev}}, w_{\text{next}})$, we verify token type compatibility (continuation tokens marked ## must follow other tokens) and compute confidence from learned

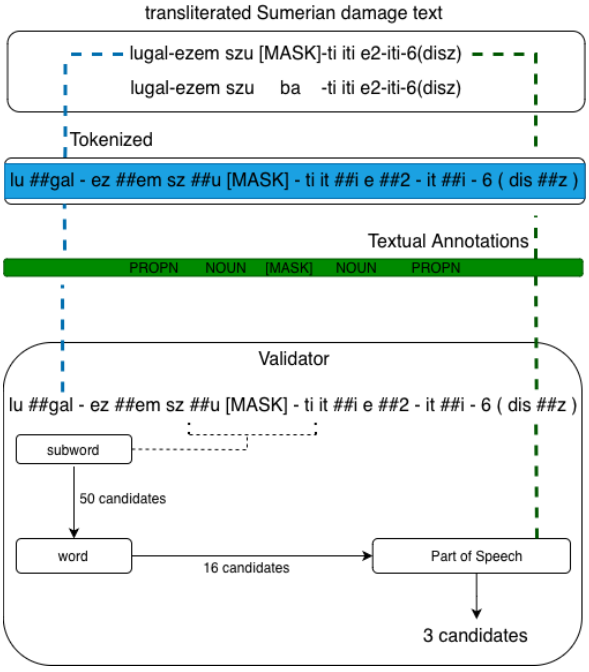

Figure 2: System architecture showing validator filtering for Sumerian text reconstruction.

bigram/trigram patterns. The subword score is:

$$s_{\text{sub}}(c) = \frac{1}{N}\left(\sum_{\text{bigrams}} s_{\text{bigram}} + 2.0 \cdot s_{\text{trigram}}\right) \quad (1)$$

where N is the number of applicable patterns. If a candidate appears in known transitions but lacks explicit patterns, we assign a default score of 0.1. If no valid context patterns exist, the candidate is rejected.

Word level reconstructs complete lexical units from subword sequences. Continuation tokens group with preceding starters to form words. We validate that the resulting word sequence appears in learned patterns and check word boundary transitions. The word-level score combines pattern confidence and boundary scores:

$$s_w(c) = \frac{1}{3}(s_p + s_{b^-} + s_{b^+}) \quad (2)$$

where s_p is the pattern score (exact match: full confidence; partial match: 0.3 weight), and s_{b^-}, s_{b^+} are the previous and next word boundary scores respectively. Each component is computed from observed frequencies in the training corpus.

POS level ensures candidates fit syntactic patterns. Since the candidate's POS tag is unknown at inference time—POS tags are available in the training corpus but the correct tag for a candidate

replacement at a masked position cannot be determined without knowing the correct token—we test all possible tags observed in training, checking if sequences (t_{before}, t) and (t, t_{after}) appear in learned patterns. For each possible POS tag t:

$$s_{\text{pos}}(t) = \frac{1}{3}(s_{b^-} + s_{b^+} + s_t) \tag{3}$$

where s_{b^-} and s_{b^+} are scores from previous and next bigram contexts, and s_t is the trigram-based score. We take the maximum across all valid tags:

$$s_{\text{pos}}(c) = \max_{t \in \mathcal{T}} s_{\text{pos}}(t) \tag{4}$$

where $\mathcal{T}$ is the set of possible POS tags observed in training.

Combined validation: The three levels operate as cascade filters. Candidates passing all levels receive a weighted average score:

$$s_v(c) = \frac{w_s \cdot s_{\text{sub}}(c) + w_w \cdot s_{\text{word}}(c) + w_p \cdot s_{\text{pos}}(c)}{w_s + w_w + w_p} \tag{5}$$

We use equal weights $w_s = w_w = w_p = 1.0$ by default. Candidates are ranked by their combined score, and only those exceeding threshold $\tau_{\text{conf}} = 0.01$ are retained. The validator reduces candidate space from ∼30,000 tokens to <20 per position (93.4% reduction) while maintaining 94.2% recall on validation data.

3.3 Hard Negative Mining

Standard MLM trains on random negatives that may be trivially distinguishable. We identify *hard negatives* that are linguistically valid according to the validator but semantically incorrect in context, forcing the model to learn fine-grained semantic distinctions.

For position i with correct token w_i, let V be tokens passing validation and B_5 be vanilla mBERT's top-5 predictions. We define hard negatives as $N_{\text{hard}} = V \setminus B_5 \setminus \{w_i\}$: tokens that are linguistically valid but not among the model's current top predictions. We use $|N_{\text{hard}}|$ as a measure of positional ambiguity. Positions where $|N_{\text{hard}}| \geq 15$ represent cases where many grammatically valid alternatives exist and the model has not yet learned to disambiguate them, these are the most informative positions for fine-tuning. We pre-compute this for all training positions by running the validator and vanilla mBERT on 4,545 sequences (masking 2 positions per sequence), yielding ∼9,000 candidate positions. We exclude 27% where $|N_{\text{hard}}| < 15$, retaining ∼6,600 positions where training signal is strongest.

3.4 mBERT Fine-tuning

We fine-tune `bert-base-multilingual-cased` (Devlin et al., 2019) using standard MLM with cross-entropy loss over the full vocabulary:

$$\mathcal{L} = -\log P(w_i \mid w_1, \ldots, [\text{MASK}]_{w_i}, \ldots, w_n) \tag{6}$$

The hard negative mining in Section 3.3 determines *which* positions are included in fine-tuning, not the loss function itself. By restricting training to positions with high ambiguity ($|N_{\text{hard}}| \geq 15$), we concentrate gradient updates on cases where multiple linguistically valid tokens compete, forcing the model to learn contextual semantic distinctions rather than trivial pattern matching.

3.5 Complementary Prediction

At inference, we combine validator and fine-tuned BERT complementarily. The validator filters the vocabulary to valid candidates $\mathcal{V}$ (typically <20 tokens). We score each $c \in \mathcal{V}$ using both components and combine:

$$s_{\text{combined}}(c) = \alpha \cdot s_{\text{BERT}}(c) + (1 - \alpha) \cdot s_{\text{val}}(c) \tag{7}$$

where $\alpha = 0.7$ (selected on validation set) emphasizes BERT's semantic judgments while incorporating validator confidence. If the validator produces zero candidates (<1% of cases), we use BERT's top-10 directly as fallback.

3.6 Implementation

We implement the system in Python using PyTorch (Paszke et al., 2019) and HuggingFace Transformers (Wolf et al., 2020). The validator uses hash-based data structures and processes the training corpus in <5 minutes. See Appendix B for complete implementation details.

4 Experimental Setup

4.1 Evaluation Protocol

For each test sequence, we randomly mask two tokens (excluding sequence boundaries) and measure whether the correct token appears in the model's top-k predictions for k ∈ {1, 5, 10}.

We report top-k accuracy rather than perplexity because our validator fundamentally changes the candidate space, making perplexity comparisons

between approaches invalid. Top-k accuracy directly measures the quality of predictions while accounting for the inherent ambiguity of token prediction in morphologically rich languages.

All experiments use 5 random seeds for robustness. The test set contains 253 sequences with 2 masked positions each, yielding 472 predictions per seed. We report mean and standard deviation across seeds. While this test set is small, it represents the complete held-out portion of our 370 annotated tablets, reflecting the constraints of working with extremely low-resource languages.

4.2 Baselines and Comparisons

We compare six approaches:

mBERT Alone: We run the base model without any help on the testing data to establish the baseline.

mBERT + Validator: Pre-trained mBERT (without fine-tuning) combined with the validator using the same complementary prediction strategy as our full approach. This isolates the contribution of hard negative fine-tuning.

Standard MLM fine-tuning: We fine-tune mBERT on the proposed dataset on standard masked language model strategy to establish a comparison between fine tuning approaches.

MLM fine-tuning + Validator: Utilize the fine-tuned mBERT combined with the validator for a clear comparison between approaches.

Hard Negative Mining Fine-tuning: Fine-tuned mBERT using our hard negative mining strategy, making predictions directly from its output distribution without validator filtering. This isolates the contribution of semantic learning from hard negatives.

Combined (Ours): Fine-tuned mBERT on hard negatives with validator using complementary prediction. This represents our proposed system.

5 Results and Discussion

5.1 Main Results

Table 1 presents the accuracy of all approaches on the test set. Our combined approach achieves 86.7 ± 1.1% top-10 accuracy, outperforming both BERT alone (78.3 ± 1.1%), the validator alone (72.8 ± 1.6%), and standard MLM fine-tuning combined with the validator (82.3 ± 0.7%). The improvement is consistent across all k values, with top-1 accuracy of 68.3 ± 1.4% compared to 51.0 ± 1.8% for hard negative BERT alone.

These results demonstrate that hard negative fine-tuning successfully teaches BERT to distinguish among linguistically valid alternatives. The fine-tuned model alone achieves 78.3 ± 1.1% top-10 accuracy, substantially higher than vanilla BERT (18.0 ± 0.6%) and standard MLM fine-tuning (75.0 ± 0.9%). This validates that mining linguistically valid but semantically incorrect examples provides stronger training signal than random masking.

The validator alone achieves 72.8 ± 1.6% top-10 accuracy through pure rule-based filtering, demonstrating that linguistic constraints extracted from limited annotations can substantially improve over the vanilla baseline. However, the validator's performance plateaus below both fine-tuning approaches, reflecting its high precision but limited ability to handle rare or novel morphological contexts unseen in the 4,545 training sequences.

5.2 Complementarity Analysis

The key finding is that combining the fine-tuned BERT with the validator yields the best performance. While BERT alone achieves strong results (78.3%), adding validator filtering provides an additional 8.4 percentage point improvement to 86.7%.

Crucially, this improvement cannot be attributed solely to candidate space reduction. If the validator's contribution were limited to narrowing the search space, then the quality of the model within that space would be irrelevant, yet fine-tuned BERT combined with the validator (86.7%) substantially outperforms vanilla BERT combined with the same validator (72.8%), a 13.9 percentage point gap demonstrating that the fine-tuned model has learned genuine semantic distinctions among the validator's candidates.

The validator's contribution is most pronounced at top-1, where it improves fine-tuned BERT's accuracy from 51.0% to 68.3% (+17.3 points). This suggests that validator filtering is particularly effective at eliminating semantically plausible but linguistically invalid top predictions from BERT. At top-10, the gain is smaller (+8.4 points), indicating that BERT already places many correct tokens within its top predictions, and the validator primarily helps with final ranking.

5.3 Comparison to Standard MLM

We compare our hard negative mining approach to standard MLM fine-tuning (random masking without validator-guided mining). Standard MLM achieves 75.0 ± 0.9% hit@10 alone and 82.3 ±

Model	Method	Validator	Hit@1	Hit@3	Hit@5	Hit@10
mBERT	None	No	6.9 ± 0.4	10.6 ± 0.6	13.5 ± 0.6	18.0 ± 0.6
mBERT	None	Yes	32.1 ± 1.2	51.8 ± 2.6	63.4 ± 2.0	72.8 ± 1.6
mBERT Fine-Tuned	Hard Negatives	No	51.0 ± 1.8	67.0 ± 1.2	72.7 ± 1.5	78.3 ± 1.1
mBERT Fine-Tuned	Hard Negatives	Yes	**68.3 ± 1.4**	**79.2 ± 1.0**	**82.7 ± 0.9**	**86.7 ± 1.1**
mBERT Fine-Tuned	Standard MLM	No	43.7 ± 1.7	57.7 ± 2.0	64.5 ± 1.5	75.0 ± 0.9
mBERT Fine-Tuned	Standard MLM	Yes	53.2 ± 0.5	68.8 ± 0.8	75.6 ± 1.1	82.3 ± 0.7
Improvement (Hard Neg + Val vs Vanilla)			*+61.4*	*+68.6*	*+69.2*	*+68.7*

Table 1: Accuracy (%) on masked token prediction. Results are averaged over 5 random seeds with 472 predictions per seed (253 test sequences with 2 masked positions each, where lines containing less than 3 tokens had to be removed). Our combined approach (hard negative fine-tuning with validator) achieves 86.7 ± 1.1% hit@10 accuracy, a 68.7 percentage point improvement over vanilla mBERT. The validator alone brings vanilla mBERT from 18.0% to 72.8%, demonstrating the strong contribution of linguistic constraints. Hard negative fine-tuning outperforms standard MLM both alone (78.3% vs 75.0%) and with the validator (86.7% vs 82.3%), validating our mining strategy.

0.7% when combined with the validator. Our hard negative approach outperforms standard MLM by 3.3 percentage points without the validator (78.3% vs 75.0%) and 4.4 percentage points with the validator (86.7% vs 82.3%). This validates our hypothesis that mining linguistically valid but semantically incorrect examples provides stronger training signal than random negatives.

5.4 Error Analysis

We manually analyzed errors made by the combined system to identify failure modes. Errors fall into two main categories:

Rare morphemes: The correct token involves a morphological form rarely observed in training, causing both the validator to filter it and BERT to rank it low. These errors reflect incomplete coverage of Sumerian's morphological space in our 4,545 training sequences (see Appendix D)

Homophony: Multiple tokens are linguistically and semantically valid in context due to ambiguity. The model chooses a plausible alternative that differs from the ground truth. This is expected and a good sign that the approach is not only just memorizing patterns.

5.5 Validator Architecture Ablation

We compare our full three-level validator (subword, word, and POS patterns) against a simpler subword-only baseline to assess the contribution of word and POS-level filtering. Table 2 shows results on the Ur III test set.

The full three-level validator achieves 86.7% compared to 85.6% for the subword-only variant, a 1.1 percentage point improvement. While this gain is modest, it demonstrates that word-level pattern reconstruction and POS-level syntactic constraints provide complementary signal beyond subword transitions.

Approach	Validator	Hit@1	Hit@10
Our	Subword-only	66.9 ± 1.1	85.6 ± 1.0
Our	**Full (3-level)**	**68.3 ± 1.4**	**86.7 ± 1.1**

Table 2: Comparison of validator architectures on Ur III test set.

5.6 Temporal Generalization

We evaluate temporal generalization by testing on tablets from the Early Dynastic IIIa period (ca. 2600-2500 BC), approximately 600 years before our Ur III training data. This out-of-domain evaluation uses 2,800 masked predictions from ED IIIa tablets with no overlap with the training set. Table 3 presents results.

The results reveal a critical asymmetry in how the validator and fine-tuned BERT generalize across time periods. On Ur III, the validator contributes +8.4pp at hit@10 (78.3% $\rightarrow$ 86.7%). However, on ED IIIa, this contribution reverses to -4.5pp (63.8% $\rightarrow$ 59.3%), with the validator actively degrading performance.

This pattern becomes more pronounced at higher k values. At hit@1, the validator still helps on ED IIIa (+5.8pp), but by hit@10, it actively harms performance (-4.5pp). This suggests the validator successfully filters truly invalid tokens even out-of-domain, but increasingly rejects period-appropriate tokens that violate Ur III-specific patterns as the candidate list expands.

Model	Method	Validator	Top-1	Top-3	Top-5	Top-10
mBERT	None	No	4.3 ± 0.1	7.1 ± 0.2	9.5 ± 0.2	13.5 ± 0.2
mBERT	None	Yes	23.6 ± 0.5	40.5 ± 0.4	47.0 ± 0.4	55.3 ± 0.5
mBERT Fine-tuned	Hard Negatives	No	30.1 ± 0.5	50.7 ± 0.3	**57.1 ± 0.5**	**63.8 ± 0.4**
mBERT Fine-tuned	Hard Negatives	Yes	**35.9 ± 0.2**	**52.4 ± 0.3**	55.7 ± 0.2	59.3 ± 0.2
Validator Contribution						
Ur III (In-Domain)			+17.3	+12.2	+10.0	+8.4
ED IIIa (Out-of-Domain)			+5.8	+1.7	−1.4	−4.5

Table 3: ED IIIa (out-of-domain, 600 years earlier). The validator improves in-domain performance across all metrics but degrades out-of-domain performance at top-5 and top-10, demonstrating period-specific pattern learning. Hard negative fine-tuning alone (without validator) achieves the best out-of-domain results (63.8% hit@10), showing that learned semantic representations transfer better than explicit morphosyntactic rules across temporal boundaries.

Fine-tuned BERT without the validator achieves the best out-of-domain performance (63.8% hit@10), dropping only 14.5pp from its in-domain performance (18.5% relative drop). In contrast, the combined system drops 27.4pp (31.6% relative drop). This demonstrates that BERT's learned semantic representations transfer more robustly across periods than the validator's explicit morphosyntactic rules.

These findings demonstrate that the complementarity between rule-based and neural components is fundamentally domain-dependent. For in-domain deployment with period-specific annotations, the full system (86.7%) is optimal. For cross-period application, fine-tuned BERT alone (63.8%) provides better generalization by avoiding the validator's period-specific brittleness on extended candidate selection.

6 Related Work

Our work intersects three research areas: neural methods for ancient language restoration, hard negative mining for language models, and low-resource multilingual NLP.

6.1 Neural Approaches to Ancient Language Processing

The application of neural methods to ancient cuneiform languages has seen growing interest in recent years. Fetaya et al. (2020) pioneered the use of recurrent neural networks for restoring fragmentary Babylonian texts, training LSTM models on administrative documents from the Achaemenid period. Their approach achieved up to 95% success rate for top-10 word predictions on Late Babylonian archival texts with highly structured syntax. However, their work focused on Akkadian rather than Sumerian and used sequence-to-sequence generation rather than masked language modeling.

Building on this foundation, Lazar et al. (2021) formulated ancient text restoration as a masked language modeling task, explicitly connecting it to modern pre-training objectives. They developed BERT-based architectures for Akkadian that achieved 89% hit@5 accuracy on missing token prediction with approximately 1M tokens of training data.

Our work achieves comparable final performance (82.7% hit@5, 86.7% hit@10) despite fundamentally different conditions: Sumerian's language isolate status precludes the cross-lingual transfer from related Semitic languages that Lazar et al. leveraged, and our training set is orders of magnitude smaller (4,545 sequences vs. 1M tokens). While these differences prevent direct comparison, they suggest that validator-guided hard negative mining offers an alternative path to strong performance when neither related languages nor large corpora are available.

Beyond cuneiform, neural approaches have been successfully applied to other ancient languages. Assael et al. (2022) developed Ithaca for ancient Greek inscriptions, combining textual restoration with geographical and chronological attribution. Their work emphasized human-AI collaboration, showing that historians using Ithaca improved their restoration accuracy from 25% to 72%. While we do not address geographical or temporal attribution, we share the goal of creating tools that augment rather than replace human expertise.

6.2 Hard Negative Mining for Language Models

Hard negative mining has proven effective in various NLP tasks, particularly for contrastive learning and retrieval. The technique originated in computer

vision but has been widely adopted for text embedding models. In the context of dense retrieval, methods like DPR Karpukhin et al. (2020) and RocketQA Qu et al. (2021) use BM25 or trained bi-encoders to mine hard negatives that are semantically similar to queries but not correct answers.

The effectiveness of hard negatives depends critically on their quality. Training on false negatives can harm performance Moreira et al. (2024).

6.3 Multilingual BERT for Low-Resource Languages

Multilingual BERT Devlin et al. (2019) has demonstrated surprising cross-lingual transfer capabilities despite being trained on Wikipedia text for 104 languages without explicit cross-lingual supervision. However, its effectiveness varies dramatically across languages. Wu and Dredze (2020) showed that mBERT performs much worse on low-resource languages in the bottom 30% of its training distribution, often failing to match even simple baseline models on tasks like named entity recognition.

Several approaches have been proposed to improve mBERT for low-resource languages. Wang et al. (2020) extended mBERT's vocabulary with language-specific tokens, achieving 23% F1 improvement on Named Entity Recognition for languages not in mBERT's training data. Pfeiffer et al. (2020) introduced adapter-based methods that enable efficient cross-lingual transfer. However, these approaches generally assume access to monolingual corpora for vocabulary extension or adapter training.

7 Conclusion

Our results demonstrate that validator-guided hard negative mining successfully combines linguistic knowledge with neural language modeling for low-resource ancient languages. The key insights are:

First, training position selection matters. Standard MLM masks positions uniformly, including many where the correct token is trivially identifiable. Our validator-guided mining identifies positions where multiple linguistically valid alternatives exist and the model does not yet rank the correct token highly, concentrating fine-tuning on the most informative cases. This is validated by our comparison to standard MLM: validator-guided position selection outperforms uniform masking by 4.4 percentage points (86.7% vs 82.3% combined with validator).

Second, the validator and BERT are truly complementary. The validator excels at filtering based on morphosyntactic patterns but cannot select among semantically similar candidates. BERT provides semantic ranking but may propose linguistically invalid tokens. Combining them yields 86.7% accuracy, better than either alone (78.3% BERT, 72.8% validator). However, this complementarity is domain-dependent: temporal generalization evaluation (Section 5.6) reveals that the validator's period-specific rules degrade performance on tablets from 600 years earlier, while BERT's semantic representations transfer more robustly.

Third, this approach is particularly well-suited to low-resource scenarios. The validator requires only POS annotations (370 tablets), not full semantic parsing. The hard negatives allow efficient use of limited fine-tuning data by focusing on difficult cases. The combination enables strong performance (86.7%) despite having orders of magnitude less data than modern NLP benchmarks.

Fourth, the approach generalizes beyond Sumerian in principle. The validator requires only POS annotations, which exist for other ancient languages including Akkadian (Lazar et al., 2021), Hittite, and Ancient Egyptian. The position selection strategy is language-agnostic: it requires only a validator capable of identifying linguistically valid candidates and a pre-trained model whose predictions can be evaluated against them. For low-resource languages that are not isolates, cross-lingual transfer from related languages would provide a stronger baseline, and our method would stack on top of that transfer by concentrating fine-tuning on positions where the model still struggles. We expect the validator's contribution to scale with morphological complexity, languages with rich agglutinative morphology like Sumerian present more ambiguous positions, offering more opportunities for targeted training.

8 Limitations

While our approach demonstrates strong performance on Sumerian masked language modeling, several limitations warrant discussion.

Model Selection. We evaluate exclusively with mBERT and do not compare to other multilingual models such as XLM-R (Conneau et al., 2020). While our approach is tokenizer-agnostic, different tokenizers (e.g., XLM-R's SentencePiece vs mBERT's WordPiece) produce fundamentally dif-

ferent segmentations that complicate direct comparison. Since neither mBERT nor XLM-R include Sumerian in their pre-training, we expect similar baseline performance and comparable improvements from our method. However, empirical validation of cross-tokenizer generalization remains future work.

Annotation Requirements. Our validator requires POS-annotated training data, limiting applicability to the 370 tablets (0.3% of CDLI's Sumerian corpus) with complete annotations. While this is orders of magnitude less than typical NLP datasets, it still represents a significant annotation burden for ancient languages. Extending our approach to fully unannotated settings through unsupervised pattern mining or cross-lingual projection from related languages could broaden applicability.

Temporal Generalization. Our validator learns period-specific morphosyntactic patterns that degrade performance on tablets from different historical periods. While fine-tuned BERT generalizes reasonably across 600 years (18.5% relative drop), the validator's contribution reverses from +8.4pp in-domain to -4.5pp out-of-domain. This limits deployment flexibility: period-specific validators must be retrained for each target era, requiring annotated data from multiple periods.

Error Analysis Scope. Our manual error analysis examines a sample of failures rather than exhaustive categorization of all errors. The three identified failure modes (rare morphemes, homophony, long-range dependencies) likely capture the majority of systematic errors, but additional patterns may exist in unexamined cases.

These error patterns suggest that further improvements would require either larger annotated datasets (to cover rare morphemes) or architectural changes to model longer contexts.

Evaluation Dataset Size. Despite using 5 random seeds for robustness, our test set contains only 253 sequences (472 predictions per seed). This reflects the extreme scarcity of annotated Sumerian data but limits statistical power for detecting small performance differences. The low variance across seeds (std $\leq 2.0\%$) partially mitigates this concern.

Single Language and Evaluation Scope. We evaluate only on Sumerian, a language isolate with no known relatives. While this demonstrates our approach works without cross-lingual transfer, it leaves open questions about generalization to other low-resource languages with different linguistic properties or availability of related languages for transfer learning. Additionally, we evaluate only on masked token prediction, an intrinsic metric that does not directly assess performance on practical restoration tasks. End-to-end evaluation on full tablet reconstruction, particularly in collaboration with Assyriologists, remains important future work.

Ethical Considerations

This work uses publicly available cuneiform texts from the Cuneiform Digital Library Initiative. No human subjects or private data are involved. We note that automated text reconstruction tools should augment, not replace, expert philological judgment.

Acknowledgements

We thank Dr. Virgilius-Aurelian Minuţă for supervising the thesis that inspired this work. We also thank the anonymous ACL 2026 SRW reviewers for their constructive feedback, which substantially improved this paper.

References

Yannis Assael, Thea Sommerschield, Brendan Shillingford, Mahyar Bordbar, John Pavlopoulos, Marita Chatzipanagiotou, Ion Androutsopoulos, Jonathan Prag, and Nando de Freitas. 2022. Restoring and attributing ancient texts using deep neural networks. *Nature*, 603(7900):280–283.

CDLI Contributors. 2026. Cuneiform digital library initiative. Accessed: 2026-02-19.

Christian Chiarcos, Émilie Pagé-Perron, Ilya Khait, Niko Schenk, and Lucas Reckling. 2018. Annotating a low-resource language with LLOD technology: Sumerian morphology and syntax. *Information*, 9(11).

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ethan Fetaya, Yonatan Lifshitz, Elad Aaron, and Shai Gordin. 2020. Restoration of fragmentary babylonian texts using recurrent neural networks. *Proceedings of the National Academy of Sciences*, 117(37):22743–22751.

Shai Gordin, Aleksi Sahala, Shahar Spencer, and Stav Klein. 2025. EvaCun 2025 shared task: Lemmatization and token prediction in Akkadian and Sumerian using LLMs. In *Proceedings of the Second Workshop on Ancient Language Processing*, pages 242–250, The Albuquerque Convention Center, Laguna. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Koren Lazar, Benny Saret, Asaf Yehudai, Wayne Horowitz, Nathan Wasserman, and Gabriel Stanovsky. 2021. Filling the gaps in Ancient Akkadian texts: A masked language modelling approach. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4682–4691, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Gabriel de Souza P Moreira, Radek Osmulski, Mengyao Xu, Ronay Ak, Benedikt Schifferer, and Even Oldridge. 2024. Nv-retriever: Improving text embedding models with effective hard-negative mining. *arXiv preprint arXiv:2407.15831*.

Émilie Pagé-Perron, Maria Sukhareva, Ilya Khait, and Christian Chiarcos. 2017. Machine translation and automated analysis of the Sumerian language. In *Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 10–16, Vancouver, Canada. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An adapter-based framework for multi-task cross-lingual transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Ravneet Punia, Niko Schenk, Christian Chiarcos, and Émilie Pagé-Perron. 2020. Towards the first machine translation system for Sumerian transliterations. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3454–3460, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5835–5847, Online. Association for Computational Linguistics.

Marie-Louise Thomsen. 1984. *The Sumerian Language: An Introduction to its History and Grammatical Structure*. Akademisk Forlag.

Zihan Wang, Karthikeyan K, Stephen Mayhew, and Dan Roth. 2020. Extending multilingual BERT to low-resource languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2649–2656, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Shijie Wu and Mark Dredze. 2020. Are all languages created equal in multilingual BERT? In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 120–130, Online. Association for Computational Linguistics.

A ATF Format Example

```
&P100004 = AAS 016
#atf: lang sux
@tablet
@obverse
1. 2(disz) geme2 ki lugal-si!-NE!-e-ta
2. iti sig4-{gesz}i3-<szub>-ga2-ra-ta
3. u4 6(disz) ib2-ta-zal
4. ur-ab-ba lunga?
5. i3-dab5
@reverse
$ blank space
1. iti sig4-{gesz}i3-szub-ga2-ra
2. mu si-mu-ru-um{ki} ba-hul
```

B Method Details

B.1 Mining Strategy Details

B.1.1 Hard Negative Definition

Formally, for position i with correct token w_i:

$$\mathcal{V}(i) = \{c \mid s_v(c) > \tau_{\text{conf}}\} \tag{8}$$

$$\mathcal{B}_5(i) = \text{top-5}(P_{\text{mBERT}}(c \mid \text{context})) \tag{9}$$

$$\mathcal{N}_{\text{hard}}(i) = \mathcal{V}(i) \setminus \mathcal{B}_5(i) \setminus \{w_i\} \tag{10}$$

B.1.2 Sampling Procedure

For each training sequence:

1. Sample 2 positions uniformly (excluding boundaries)
2. For each position i:
   (a) Run validator to obtain $V(i)$
   (b) Run vanilla mBERT to obtain $B_5(i)$
   (c) Compute $N_{\text{hard}}(i)$
   (d) If $|N_{\text{hard}}(i)| \geq 15$: include this position in the fine-tuning set.
   (e) Else: skip this position (insufficient ambiguity for informative training).

This procedure yielded 9,000 training samples from 4,545 sequences, with 27% of positions excluded.

B.2 Training Configuration

B.2.1 Hyperparameters

Parameter	Value
Validator	
Frequency threshold (τ_{freq})	2
Confidence threshold (τ_{conf})	0.01
Weights (w_s, w_w, w_p)	1.0 each
Hard Negative Mining	
Minimum hard negatives threshold	15
Masking positions per sequence	2
Excluded positions	27%
mBERT Fine-tuning	
Learning rate	2×10^{-5}
Batch size (effective)	8
Gradient accumulation steps	8
Gradient clipping	1.0
Epochs	3
Selected checkpoint	Epoch 2
Complementary Prediction	
BERT weight (α)	0.7
Validator weight ($1 - \alpha$)	0.3

Table 4: Complete hyperparameter configuration.

B.2.2 Hardware and Timing

All experiments run on a single NVIDIA RTX 4060 GPU with 8GB memory. Timing breakdown:

- Validator training: $<$5 minutes
- Hard negative mining: $\sim$1 hour (includes validator + vanilla mBERT inference)
- mBERT fine-tuning: $\sim$1 hour for 3 epochs

Implementation uses PyTorch 2.0 and HuggingFace Transformers 4.30.

B.3 Implementation Details

B.3.1 Validator Data Structures

The validator uses Python dictionaries to store:

- Subword patterns: `Dict[(token, token) -> float]` for bigrams
- Word patterns: `Dict[tuple[token, ...] -> float]` for word sequences
- POS patterns: `Dict[(POS, POS) -> float]` for tag bigrams

Pattern frequencies are computed during training and normalized to confidence scores in $[0, 1]$.

B.3.2 Fallback Mechanism

The fallback strategy activates when $|\mathcal{V}| = 0$. In this case:

1. Run BERT on full vocabulary
2. Return top-10 predictions directly
3. Log the fallback event for analysis

This occurs in $<$1% of test positions, typically for rare morphemes or unseen contexts.

C Reconstruction Examples

C.1 Example 1: Challenging Case

Original: ugu2 da-da ba-a-gar

Masked: ['ug', '##u', '##2', '[MASK]', '-', 'da', 'ba', '-', 'a', '-', 'gar']

Correct token: da

Model	Rank	Top-1 (Score)
Vanilla mBERT	>10	##u (0.128)
Vanilla + Validator	>10	a (0.043)
Hard Negative	**3**	mu (0.266)
Hard Negative + Validator	**3**	mu (0.198)
Standard MLM	>10	ba (0.313)
Standard MLM+ Validator	>10	ba (0.231)

Table 5: Example 1 predictions. Hard Negative achieves rank 3; both the baseline and standard MLM model fail on this case.

C.2 Example 2: Successful Reconstruction

Original: kiszib3 ensi2

Masked: ['kis', '##zi', '##b', '##3', '[MASK]', '##2']

Correct token: ensi

Model	Rank	Top-1 (Score)
Vanilla mBERT	>10	##b (0.241)
Vanilla + Validator	>10	i (0.029)
Hard Negative	**1**	ensi **(0.728)**
Hard Negative + Validator	**1**	ensi **(0.538)**
Standard MLM	**1**	ensi (0.187)
Standard MLM + Validator	**1**	ensi (0.159)

Table 6: Example 2 predictions. Both fine-tuned models achieve rank 1 with correct prediction.

C.3 Example 3: Validator-Only Success

Original: dumu puzur4-{d}nanna

Masked: ['dum', '##u', 'pu', '##zur', '[MASK]', '-', '{', 'd', '}', 'nan', '##na']

Correct token: ##4

Model	Rank	Top-1 (Score)
Vanilla mBERT	>10	##u (0.251)
Vanilla + Validator	**1**	##4 (0.023)
Standard MLM	>10	ur (0.252)
Standard MLM + Validator	**1**	##4 (0.031)
Hard Negative	>10	##a (0.156)
Hard Negative + Validator	**1**	##4 (0.030)

Table 7: Validator filtering essential: all models fail without it, succeed with it.

D Error Analysis examples

Most of the missed tokens by both the validator's and fine-tuned mBERT's predictions fall in one of these scenarios:

D.1 Multiple tokens are semantically valid

In multiple texts, tokens follow the same pattern, that results in uncertainty while selecting candidates from both the model and the validator.

Original Text: mu hu-uh2-nu-ri{ki} ba-hul **Masked position 9:** mu hu - u ##h ##2 - nu - **[MASK]** { ki } ba - hul

All the methods that work with the validator predict plausible tokens in the presented context. Although the pattern "**ri{ki}**" appears in training, the number of occurrences is rather low: 8 compared to 319 that follow the same "token {ki}" pattern.

D.2 Token out of training corpus

In rare cases, tokens are out of the validator's rules. While this is taken into account, the validator could have incorrect tokens that match the patterns, resulting in not falling into the fallback strategy.