SuBERT *

Luca Martucci, Michael Menghi,
Samuele Nicolò Straccialini, Francesco Vinciguerra

Spring 2025

Abstract

Recent advances in computer vision and natural language processing open new possibilities for reading and translating ancient cuneiform texts: in this work, we tackle one of the most ancient and obscure languages, spoken by the Sumerians, a grandiose civilization that flourished in the Near East around 2000 BCE and vanished millennia ago, leaving us with only a few scant testimonies. We focus on two main tasks: first, we exploit image-classification techniques to isolate and recognize individual characters from high-resolution tablet images. Then, we attempt to translate Sumerian into English via Transformer-like architectures.
The complete code is available in the following GitHub repository: https://github.com/samusamu2/AI-project-SuBERT

1 Introduction

Deciphering ancient languages has always been a fundamental task in historical studies. The interest in such a problem is quite old, but serious decipherments did not take place until the 18th century, the most famous case being the one of Egyptian hieroglyphs. Their decipherment only took place in 1822, with Champollion's work on the Rosetta Stone, a trilingual inscription containing text in classical hieroglyphs, demotic, and Greek. Clearly, the comparison with a known language is considered to be a starting point for decipherment, allowing researchers to exploit heuristic techniques, based on philological and linguistic knowledge. Indeed, a large number of successful decipherments came from multilingual inscriptions, or more generally, by the study of languages close to the target one. Sumerian language, which is the focus of our project, makes no exception: its decipherment has been a long, multi-stage process, based on comparison with Akkadian language, which was itself understood only when compared to Old Persian.

With the rise of new technologies, computational approaches to decipherment are becoming a valid line of research. For instance, in [LCB19], a machine learning model was able to re-decipher Linear B and Ugaritic, exploiting data from two similar languages, respectively Greek and Hebrew. Nevertheless, automatic decipherment presents various difficulties, the lack

*We chose this name for our report in the spirit of the playful naming convention in the NLP community, where many language models include "BERT" in their names. Despite this, our analysis does not rely on the original BERT architecture for any of the main tasks.

of parallel data and the scarcity of ancient texts being the most evident. Moreover, as each decipherment requires a huge domain-specific knowledge, realizing a single model which works on multiple languages is absolutely non-trivial. That being said, computational approaches certainly offer a valid help, at the very least in speeding up tasks that are usually carried out in years of work, such as capturing linguistic relationships with known languages.

Our intent with this project is to "test" the power of AI models in this context by concentrating on a well-understood language. A notable example in such a sense is Google's Fabricius, which enables recognition and translation of Egyptian hieroglyphs. With this example in mind, we decided to focus on Sumerian cuneiform, a language which is widely studied, but probably less "popular" than hieroglyphic.

2 Dataset

In our project, we relied on the SumTablets datasets curated by Cole Simmons, Richard Diehl Martinez, and Prof. Dan Jurafsky. Their work, which can be found in their GitHub repository, distributed through HuggingFace and licensed under the Creative Commons Attribution 4.0 International, allowing for its adaptation and reuse with proper credit. We now stick to a basic description of the dataset, and refer the interested reader to the aforementioned GitHub repository and the related [SDJ24].

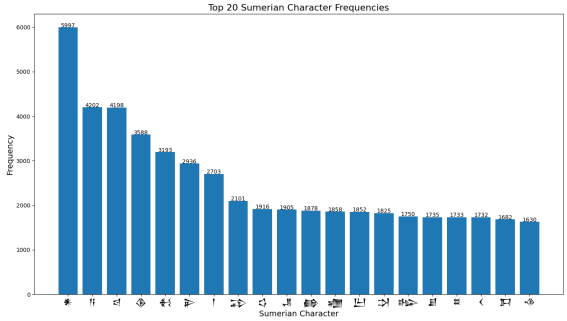

Figure 1: Top sumerian characters

2.1 Data Analysis

We performed data analysis on a dataset obtained by merging SumTablets and SumTablets.English, thus containing, for each glyph inscription: the period in which it was written, the genre (e.g., administrative, royal inscription etc.), the transliteration, and the English translation. We first compared the text length distribution in translation and transliteration, discovering a very similar distribution. We also visualized the most frequent Sumerian characters (see fig. 1), as well as most frequent English words in translation, and then we clustered the latter by topic (see fig. 2.1) via Latent Dirichlet Allocation.

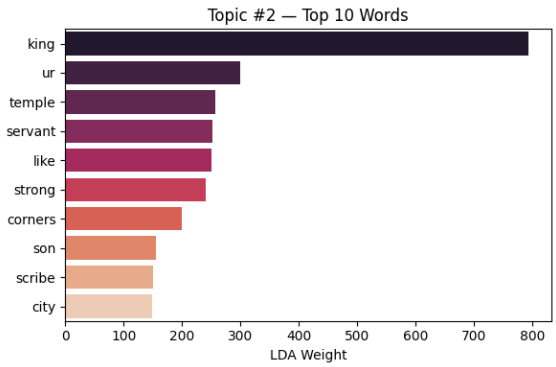

Figure 2: Top words per topic.

As a first assessment of the quality of translations in the dataset, we plotted the number of tokens in translation versus transliteration; graphically, the scaling seems approximately linear, and this relationship becomes stronger if the texts are subcategorized by their genre (see fig. 3).

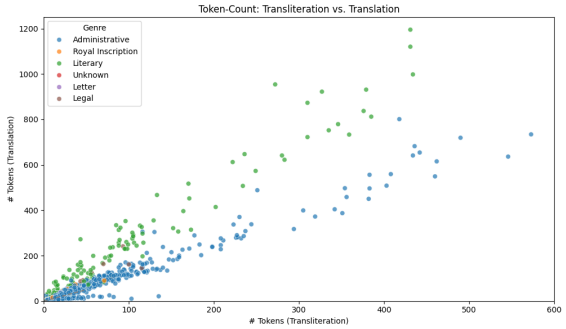

Figure 3: Token count.

This makes sense: clearly, an administrative document will be much more "neutral" in style than, say, a literary inscription, thus requiring a more "basic" translation for the same number of tokens in transliteration. A similar trend emerges if texts are subcategorized by time periods.

3 Vision task: recognizing glyphs

As a first step, we decided to translate raw tablet imagery into structured, machine-readable representations of individual glyphs. At a high level, we desired to have our pipeline:

- locate each discrete sign in a photograph or scan of a tablet,
- isolate it from surrounding noise (e.g., cracks, erosion, background texture),
- produce a normalized image patch or feature description amenable to downstream classification or transliteration.

Achieving these goals is nontrivial: cuneiform characters can be densely clustered, overlap slightly, or exhibit fragmentary strokes; lighting conditions, surface wear, and uneven clay coloration further complicate reliable sign detection. In practice, a variety of techniques, ranging from classical image-processing (thresholding, edge detection, connected-component analysis) to more modern learning-based methods (feature-extraction networks, region-proposal frameworks, semantic/instance segmentation), may be applied. The objective is not only to draw tight bounding boxes around each wedge but also to assign each glyph a consistent orientation and scale, even when parts of it are chipped away or obscured. By solving this "glyph-localization" problem in a robust manner, we hoped that subsequent recognition and transliteration stages operate on clean, well-aligned inputs. Key challenges we tackled included differentiating true cuneiform impressions from spurious cracks or shadows, accounting for different tablet styles (school texts versus administrative tablets, for example), where sign sizes and carving depths vary dramatically, and overcoming the limited availability of finely annotated training data, which often requires synthetic augmentation or careful data-curation to achieve high recall under real-world conditions.

3.1 YOLO

The core of our approach leverages a YOLO–based object detection framework to localize and classify individual cuneiform signs on photographed or scanned Sumerian tablets, effectively transforming each tablet image into a structured layout of bounding boxes corresponding to specific glyph types. YOLO's single-stage architecture makes this especially well-suited to the variability and clutter often encountered in ancient tablets, where signs can be densely inscribed, unevenly illuminated, and subject to erosion, because it simultaneously predicts objectness scores and class probabilities for a dense grid of image regions in one forward pass. By initializing from a lightweight, pre-trained YOLO11 backbone and fine-tuning with extensive syn-

thetic augmentations (e.g., mosaic blending, random jitter, and color/contrast perturbations), we encourage the model to learn robust features that distinguish subtle wedge shapes and orientations characteristic of cuneiform. Moreover, incorporating segmentation annotations alongside bounding-box labels further refines the network's capacity to discriminate overlapping or partially damaged signs by explicitly modeling pixel-wise foreground masks.

3.2 Faster R-CNN

Initially, we trained a YOLO-based detector on a large synthetic glyph dataset, increasing both the number of generated samples and the number of training epochs. Despite these efforts, YOLO's one-stage architecture struggled to precisely localize small, tightly clustered glyphs; even after extensive augmentation, its performance plateaued. In particular, overlapping symbols and minute details were often missed, and classification scores remained inconsistent. Recognizing the inherent limitations of a single-pass detector in our small-object context, we decided to adopt a two-stage approach. Faster R-CNN, which first proposes candidate regions via an RPN (Regional Proposed Network) and then refines them through a separate classification and bounding-box regression head, offered better localization accuracy at the cost of higher computational load. Because our hardware resources were limited but still sufficient for moderate two-stage training, we chose Faster R-CNN as the best compromise between detection quality and resource requirements.

4 Translation task: from Sumerian to English

4.1 Metrics

To evaluate the quality of the Sumerian–English translations, we employed several complementary metrics that capture different aspects of linguistic fidelity.

METEOR measures unigram overlap between the generated and reference English texts while incorporating synonymy and stemming; it penalizes word-order fragmentation and therefore rewards semantically accurate yet flexible phrase-level matches. Concretely, let

$$P = \frac{|\text{aligned unigrams}|}{|\text{hypothesis unigrams}|}$$

and

$$R = \frac{|\text{aligned unigrams}|}{|\text{reference unigrams}|}.$$

then their weighted F-score is

$$F_\alpha = \frac{P \cdot R}{\alpha\, P + (1-\alpha)\, R},$$

and the original paper [BL05] chose the harmonic mean with $\alpha = 0.9$. If $m = |\text{aligned unigrams}|$ and ch is the number of discontiguous "chunks" of aligned unigrams, then with penalty weights $\gamma = 0.5$ and $\beta = 3$ we define

$$\text{Penalty} = \gamma\big(\tfrac{\mathrm{ch}}{m}\big)^{\beta}.$$

Finally,

$$\text{METEOR} = F_\alpha \times \big(1 - \text{Penalty}\big).$$

chrF, a straightforward modification of METEOR proposed in [Pop15] computes an F-score over character-level n-grams, which is especially useful for morphologically rich or formulaic texts, as it captures subword and inflectional variations that word-based

metrics might miss. For each order $n = 1, 2, \ldots, N$, let C_n be the number of overlapping char-n-grams,

$$H_n \;=\; |\text{char-}n\text{-grams in hypothesis}|,$$

and

$$R_n \;=\; |\text{char-}n\text{-grams in reference}|.$$

Define

$$P_n \;=\; \frac{C_n}{H_n}, \quad R_n \;=\; \frac{C_n}{R_n}.$$

Let $P = \sum_{n=1}^{N} P_n$ and $R = \sum_{n=1}^{N} R_n$. With $\beta^2 = 1/2$, chrF is

$$\text{chrF} \;=\; \frac{(1+\beta^2)\, P \cdot R}{\beta^2\, P \;+\; R}.$$

Cosine Similarity (Embedding-based) assesses semantic equivalence beyond surface form. If $\mathbf{h}, \mathbf{r} \in \mathbb{R}^d$ are the d-dimensional embeddings of hypothesis and reference, then

$$\cos(\mathbf{h}, \mathbf{r}) \;=\; \frac{\mathbf{h}\cdot\mathbf{r}}{\|\mathbf{h}\|\,\|\mathbf{r}\|}.$$

This score is very basic and clearly lies in $[0, 1]$ for sentence embeddings, with 1 indicating near-identical semantic content.

Labeled Attachment Score (LAS, see [NF17]) quantifies syntactic preservation using dependency parses of the English reference and generated output. Suppose both sentences have N tokens aligned by position, and for each token i we denote: $h_r^{(i)} = \text{head}_{\text{ref}}(i)$, $d_r^{(i)} = \text{dep}_{\text{ref}}(i)$, $h_h^{(i)} = \text{head}_{\text{hyp}}(i)$, and $d_h^{(i)} = \text{dep}_{\text{hyp}}(i)$. Define the indicator

$$\mathbb{1}_i \;=\; \begin{cases} 1, & \text{if } \left(h_h^{(i)} = h_r^{(i)}\right) \wedge \left(d_h^{(i)} = d_r^{(i)}\right), \\ 0, & \text{otherwise.} \end{cases}$$

Then

$$\text{LAS} \;=\; \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_i.$$

High LAS indicates that the hypothesis preserved the reference's syntactic structure (low syntactic drift).

4.2 GPT2

At its core, the Sumerian-to-English translation pipeline reframes transliteration as a conditional text-generation problem, leveraging a pretrained autoregressive transformer (in this case, a GPT-2 "medium" model, consisting of 355M parameters) to learn the mapping from Sumerian sequences written in Latin-letter transliteration to fluent English renderings. Rather than building a bespoke encoder–decoder, our approach simply concatenates each training pair into a single 'Sumerian + English' string, and then fine-tunes GPT-2 to predict whatever follows "English:" whenever it sees a new Sumerian prompt. By initializing from a large, unsupervised language model that already captures general world and linguistic knowledge, we dramatically reduce the amount of parallel data needed to learn accurate translations of cuneiform contents; the model's transformer layers, multiheaded self-attention, and deep residual stacks are adept at modeling long-range dependencies, which is crucial when Sumerian clauses can be elliptical, 'poetic', or contextually dense.

During training, each transliteration/English pair is tokenized into subword units, padded or truncated to a uniform length (here, 528 tokens) to fit in a consistent batch, and fed through the causal-language-model objective so that GPT-2 learns to assign high probability to the correct English words given the preceding Sumerian prefix. A small validation split helps guard against overfitting, particularly important since our corpus of tablet-level translations is relatively small, and standard language-generation metrics, which we described above, are computed by comparing the raw generated English strings against ground-truth references.

Nonetheless, this methodology introduces a few domain-specific challenges. First, Sumerian transliterations often employ sign names, determinatives, and glosses that do not appear in GPT-2's original vocabulary; these are handled via open-vocabulary (Byte-Pair Encoding) tokenization, but rare transliteration tokens can still be split awkwardly or lead to out-of-vocabulary subtleties. Second, the dataset itself is relatively small – only a few thousand tablet sentences at most – so the model risks memorizing frequent admin texts (e.g., offering lists) rather than learning robust generalization to literary, legal, or poetic genres; aggressive regularization (weight decay, warmup schedules, gradient checkpointing) and careful length filtering (capping sequences at ~ 528 tokens) help mitigate catastrophic overfitting. Finally, our metrics can only partially capture the nuance of translation, especially for a language with flexible word order and agglutinative morphology, so human evaluation remains essential. Even so, by fine-tuning GPT-2 on concatenated bilingual "Sumerian → English" pairs, we harness a powerful conditional generation engine that can produce fluent, context-aware English translations from previously unseen Sumerian transliterated inputs.

4.3 BART

For our Sumerian-to-English pipeline, we also employed BART (Bidirectional and Auto-Regressive Transformers), a pretrained sequence-to-sequence model originally introduced by [Lew+20], combining a bidirectional (i.e., "masked") encoder, similar to BERT, with an autoregressive (i.e., "left-to-right") decoder, similar to GPT, into a single denoising autoencoder framework. During pretraining, BART corrupts input text with various noise functions (e.g., token masking, sentence permutation, text infilling) and trains the model to reconstruct the original sequence. As a result, the encoder learns to build rich contextualized embeddings over the entire input (both left and right contexts), while the decoder learns to generate coherent sequences conditioned on those embeddings. Because BART is trained with a cross-entropy objective that reconstructs full sentences from corrupted inputs, its weights capture both understanding the source structure and generating fluent text. The "base" version has approximately 139 million parameters, whereas the "large" version has approximately 406 million parameters, differing primarily in layer depth and hidden-size dimensions.

By fine-tuning BART on parallel Sumerian–English pairs, where each example is represented as a "source: $\langle$*Sumerian transliteration*$\rangle$" and "target: $\langle$*English translation*$\rangle$" pair, tokenized to fixed input/output lengths (e.g., 512 tokens each), the model learns to align agglutinative Sumerian morphemes, logograms, and determinatives with their corresponding English renderings. In practice, each training step consists of:

1. encoding the entire Sumerian transliteration with the bidirectional encoder, thereby capturing context from preceding and following signs (e.g., disambiguating case endings or sign compounds);

2. shifting the target English tokens by one position to create decoder inputs (teacher forcing), so that the decoder is trained to predict the next English token given both previously generated tokens and full Sumerian context;

3. computing cross-entropy loss against the true English sequence (labels), then backpropagating gradients through both encoder and decoder to update all transformer weights.

At inference time, the encoder first builds a contextual representation of the input Sumerian text; then an autoregressive beam-search decoder (e.g., five beams, no-repeat-n-gram constraints) generates the most likely English translation without relying on ground-truth "next tokens."
The theoretical justifications for choosing BART in this domain are the following:

- BART's encoder attends to the *entire* Sumerian input sequence (both left and right contexts). This is crucial for dealing with Sumerians' flexible word order and agglutinative morphology, where determinatives or case markers may only be fully interpretable when adjacent signs are considered simultaneously. A unidirectional encoder (or a purely autoregressive model) would not capture long-range dependencies as effectively.

- Dedicated Decoder Fluency: the autoregressive decoder produces fluent, coherent English syntax even when the source uses conventions foreign to English. By decoupling "understand source" (encoder) from "generate target" (decoder), BART naturally aligns morphological and lexical units in Sumerian with idiomatic English phrases, without intermixing source and target tokens in a single stack (as in GPT-only approaches).

- Data Efficiency and Robustness: ecause BART was pretrained on vast corpora with diverse denoising objectives, its weights already encode a strong prior over language structure. Fine-tuning on a relatively small Sumerian–English corpus (on the order of a few thousand sentences) therefore benefits from faster convergence and better generalization. Regularization techniques—weight decay, label smoothing (0.1), cosine-annealing warmup, mixed-precision (fp16)—help prevent overfitting in administrative or formulaic texts.

Nonetheless, domain-specific challenges accompany this choice, most notably: tokenization breakdowns (Sumerian transliterations often contain logographic sign names and determinatives) and the default BART vocabulary may split these into multiple subword pieces, potentially diluting semantic coherence (custom preprocessing – harmonizing determinative notation, adding special tokens for multi-character sign names – can mitigate but not entirely eliminate token fragmentation); limited parallel data (our dataset does not approach the billions of tokens BART saw during pretraining; if care is not taken, large BART can overfit peculiar sentence patterns and fail to generalize to literary or legal genres) but early-stopping callbacks based on validation METEOR scores and a modest epoch count (e.g., 100 epochs with validation monitoring) address this risk; sequence-length constraints (Sumerian sentences with extensive glosses or commentary can exceed 512 tokens) sometimes discards critical context, leading to mistranslations. In practice, setting truncation thresholds carefully and experimenting with longer maximum lengths (when GPU memory allows) is necessary.

5 Results

5.1 YOLO and FR-CNN

The result of our analysis with YOLO is a detector that achieves rapid inference, critical for batch processing large corpora and high recall on small, intricate glyphs, thereby providing reliable sign localization as the first step toward downstream recognition and transliteration into Sumerian lexica.

Model	Mean Similarity	Std Dev	Sample Size
BART_base	0.7749	0.2276	86
BART_large	0.7529	0.2391	86
GPT_2:temp=0.2	0.4783	0.2776	86
GPT_2:temp=0.7	0.4750	0.2729	86
GPT_2:temp=0.5	0.4745	0.2717	86
GPT_2:temp=1.0	0.4714	0.2744	86

Table 1: Model Comparison Summary.

Despite using fewer epochs and a smaller dataset than our YOLO experiments, Faster R-CNN achieved comparable overall accuracy and provided notably improved localization of overlapping or tiny glyphs. Some glyphs remained undetected or were misclassified, but the two-stage approach demonstrated clear benefits in handling densely packed symbols. The training time for 10.000 images over 20 epochs on the virtual machine was approximately three hours. Given additional computational resources, more extensive anchor-box tuning to match glyph aspect ratios, and possibly mixed-precision or multi-GPU training, we expect a significant reduction in missed detections and misclassifications. These improvements would ultimately lead to a more robust glyph detection pipeline.
In general, the two models behaved well on synthetic sentences and entire synthetic tablets; however, they showed poorer results on real images. To obtain more accurate results, we would need a complete labeling of a number of images of Sumerian tablets, which is out of the scope of this project. Therefore, we preferred to concentrate on the Transformers architecture.

5.2 Transformers

Despite these hurdles we described, properly fine-tuned, BART achieved robust alignment between Sumerian determinatives and English glosses, and produced coherent English clauses even when the Sumerian source omits verbs or subjects (a common feature of elliptical tablet texts). Quantitatively, base and large BART yield higher METEOR and chrF scores than simpler encoder-only or decoder-only approaches, especially on validation sets containing literary excerpts. Qualitatively, generated translations are more fluent and less prone to hallucination, in particular when compared with those generated by GPT2, which was dense with translation errors and low embedding scores.

A big gap between the base/large BART and GPT models was further highlighted in the semantic similarity distributions (see fig. 5, 6), very skewed towards high scores for BART and almost uniform, or even bimodal, for GPT2 for every choice of temperature in the schedule $\{0.2, 0.5, 0.7, 1.0\}$.

However, even the models performing the worst semantically, were able to preserve the syntactic structure, as measured by the LAS score (see fig. 7, 8), quelling the drift in the generated translation, although the overall performance for both BART and GPT was centered around 2.0. For this, the complexity and highly intricate syntactic structure of Sumerian should be held accountable: even the translations from human researchers are some of the times unintelligible to the reader, but, fixed that one, the generated translation was never too dissimilar.

6 Conclusion and Further Work

In conclusion, our work is quite satisfactory given the complexity of the task. However, several improvements can be implemented, in particular:

- to create a full pipeline, one would have to fill the gap between image recognition and translation, namely, the detection of glyphs directly from photos of real-world artifacts and their subsequent transliteration. Our project successfully explored the expressive power of YOLO and Faster R-CNN architectures: to achieve a better performance, we would need a labeled dataset of tablets;
- BART models have shown a great adaptability to the texts and are a better fit for the translation task. The "base" model performs better than the "large" ones, which may be due to the scarcity in data for the true translations: in fact the large model training stopped earlier than the base one albeit achieving similar results (fig 4);
- GPT2 is highly influenced by the variations in the input length. In fact, usually, one would have to cap the number of output tokens. This threshold heavily depends on the input sentence, and impacted very badly our translation task, with sentences often getting truncated, too long or failing to be generated (see figures 9 and 10).

Future works can concentrate on improving the aforementioned challenges and experimenting with more models, such as mT5 encoder-decoder and Gemma decoder-only architecture, which we attempted but failed to train due to computational limitations.

References

[BL05] Satanjeev Banerjee and Alon Lavie. "METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments". In: *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*. Ed. by Jade Goldstein et al. Ann Arbor, Michigan: Association for Computational Linguistics, June 2005, pp. 65–72. URL: https://aclanthology.org/W05-0909/.

[LCB19] Jiaming Luo, Yuan Cao, and Regina Barzilay. "Neural Decipherment via Minimum-Cost Flow: From Ugaritic to Linear B". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Ed. by Anna Korhonen, David Traum, and Lluís Màrquez. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 3146–3155. DOI: 10.18653/v1/P19-1303. URL: https://aclanthology.org/P19-1303/.

[Lew+20] Mike Lewis et al. "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by Dan Jurafsky et al. Online: Association for Computational Linguistics, July 2020, pp. 7871–7880. DOI: 10.18653/v1/2020.acl-main.703. URL: https://aclanthology.org/2020.acl-main.703/.

[NF17] Joakim Nivre and Chiao-Ting Fang. "Universal Dependency Evaluation". In: *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*. Ed. by Marie-Catherine de Marneffe, Joakim Nivre, and Sebastian Schuster. Gothenburg, Sweden: Association for Computational Linguistics, May 2017, pp. 86–95. URL: https://aclanthology.org/W17-0411/.

[Pop15] Maja Popović. "chrF: character n-gram F-score for automatic MT evaluation". In: *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Ed. by Ondřej Bojar et al. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 392–395. DOI: 10.18653/v1/W15-3049. URL: https://aclanthology.org/W15-3049/.

[SDJ24] Cole Simmons, Richard Diehl Martinez, and Dan Jurafsky. "SumTablets: A Transliteration Dataset of Sumerian Tablets". In: *Proceedings of the 1st Workshop on Machine Learning for Ancient Languages (ML4AL 2024)*. Ed. by John Pavlopoulos et al. Hybrid in Bangkok, Thailand and online: Association for Computational Linguistics, Aug. 2024, pp. 192–202. DOI: 10.18653/v1/2024.ml4al-1.20. URL: https://aclanthology.org/2024.ml4al-1.20/.

Appendix: visualization of test translation results

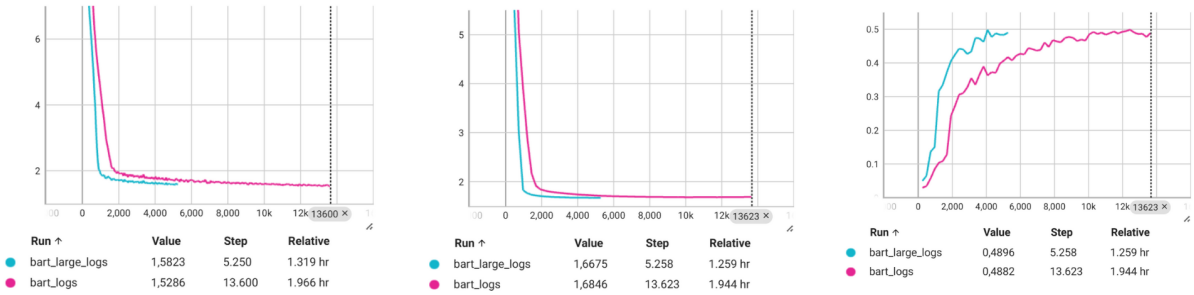

Figure 4: Training and validation losses of BART models, and METEOR results on validation data.

	BART_base	BART_large	GPT_2:temp=0.2	GPT_2:temp=0.5	GPT_2:temp=0.7	GPT_2:temp=1.0
count	86.000	86.000	86.000	86.000	86.000	86.000
mean	0.475	0.446	0.216	0.213	0.215	0.213
std	0.318	0.310	0.227	0.222	0.218	0.205
min	0.000	0.000	0.000	0.000	0.000	0.000
25%	0.174	0.136	0.000	0.000	0.000	0.000
50%	0.489	0.440	0.145	0.146	0.168	0.178
75%	0.785	0.706	0.367	0.373	0.352	0.356
max	1.000	0.999	0.836	0.811	0.811	0.816

Table 2: Summary of METEOR scores

	BART_base	BART_large	GPT_2:temp=0.2	GPT_2:temp=0.5	GPT_2:temp=0.7	GPT_2:temp=1.0
count	86.000	86.000	86.000	86.000	86.000	86.000
mean	0.571	0.541	0.324	0.318	0.327	0.323
std	0.274	0.279	0.263	0.259	0.262	0.255
min	0.035	0.019	0.001	0.001	0.001	0.001
25%	0.317	0.279	0.037	0.033	0.037	0.037
50%	0.589	0.573	0.269	0.277	0.276	0.285
75%	0.815	0.759	0.534	0.542	0.544	0.524
max	1.000	1.000	0.791	0.791	0.791	0.791

Table 3: Summary of chRF scores

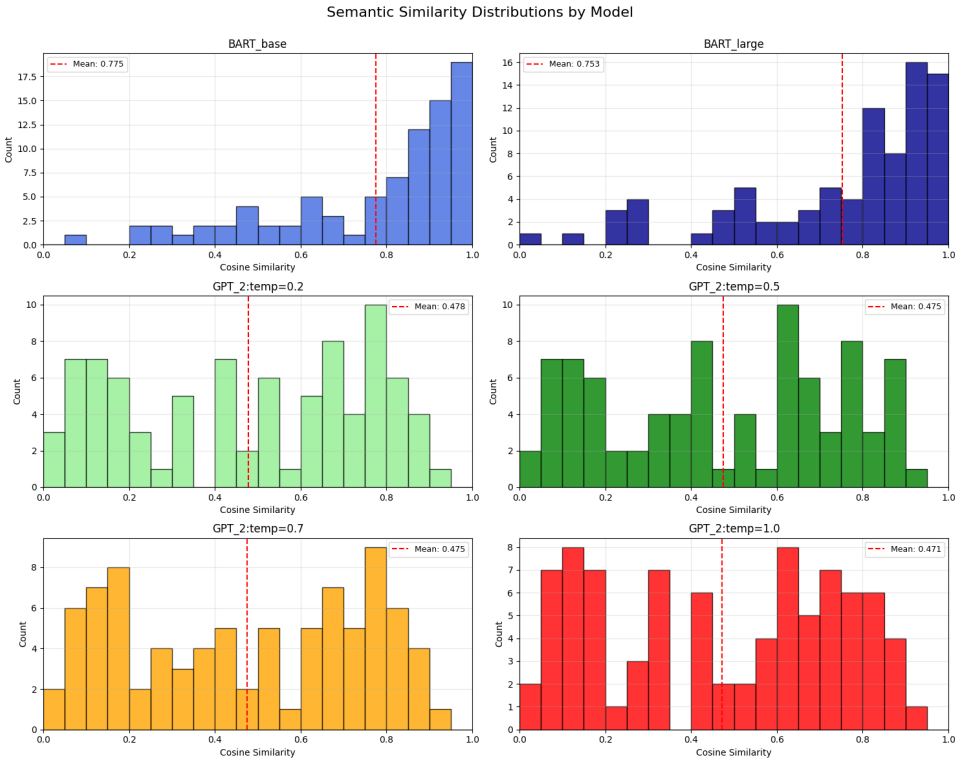

Figure 5: Semantic similarity distribution for each model.

Model	Mean LAS	Std Dev	Sample Size
BART_base	0.2762	0.3026	86
BART_large	0.2587	0.2937	86
GPT_2:temp=1.0	0.1842	0.2164	86
GPT_2:temp=0.5	0.1821	0.2131	86
GPT_2:temp=0.2	0.1813	0.2173	86
GPT_2:temp=0.7	0.1676	0.1973	86

Table 4: Syntactic Preservation Comparison

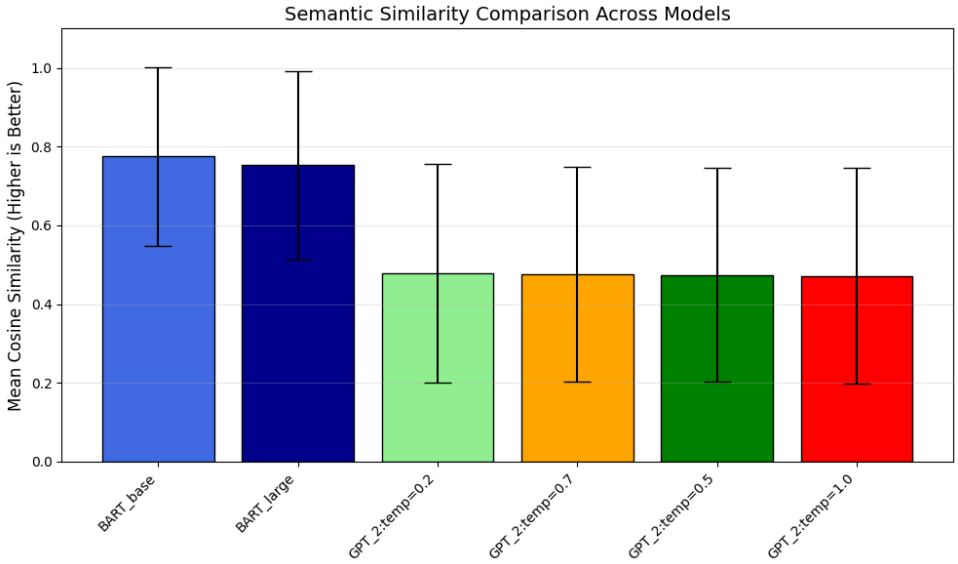

Figure 6: Semantic similarity comparison across models.

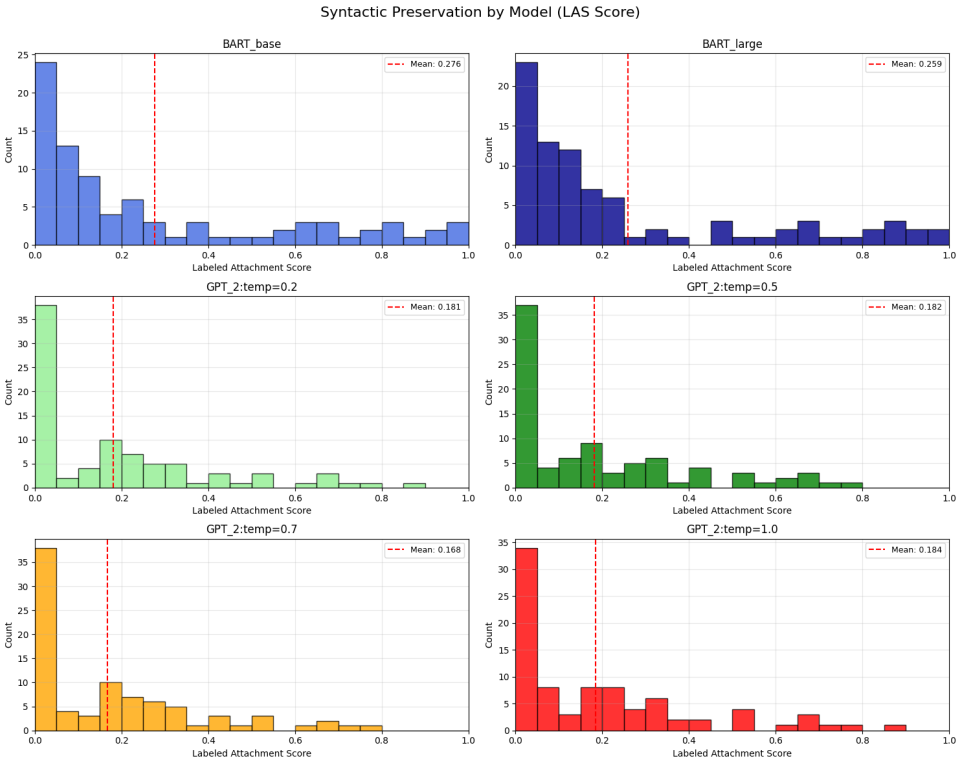

Figure 7: LAS score for each model.

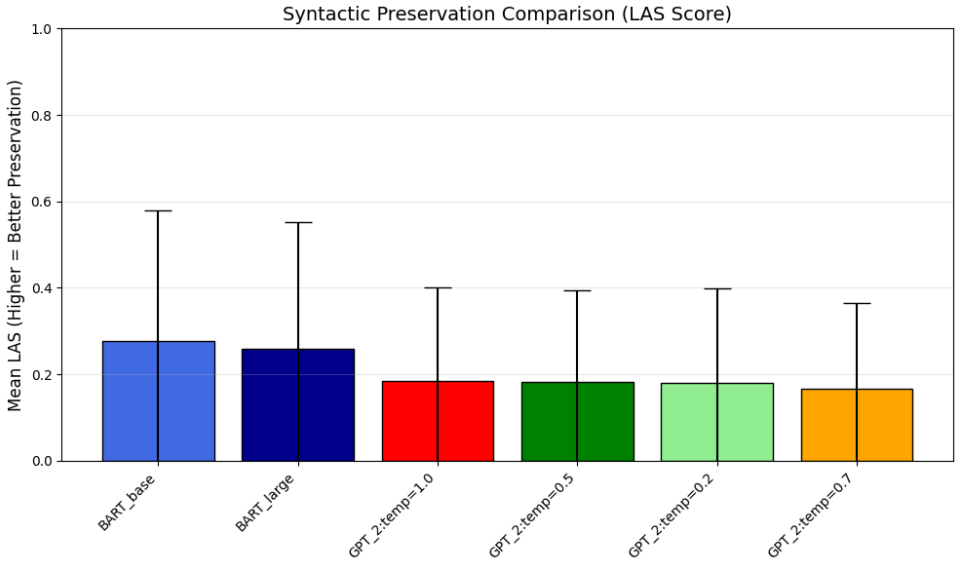

Figure 8: LAS score comparison across models.

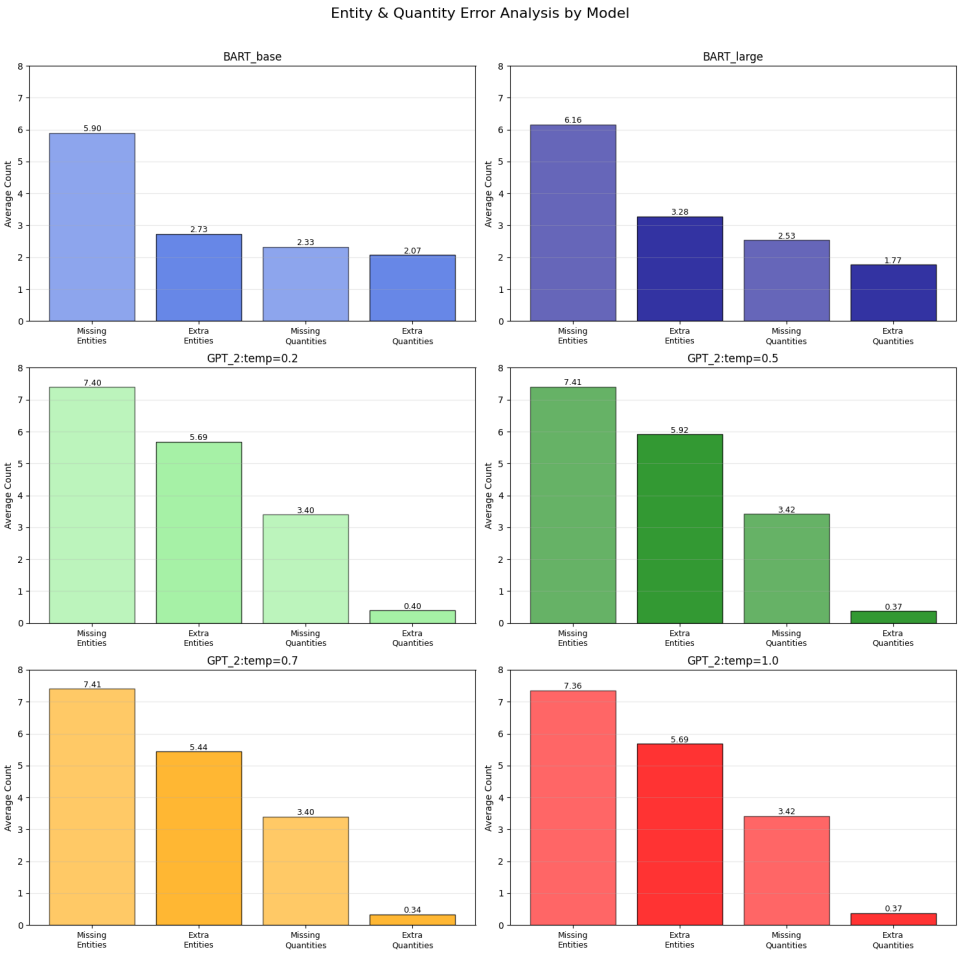

Figure 9: Missing and extra quantities for each model.

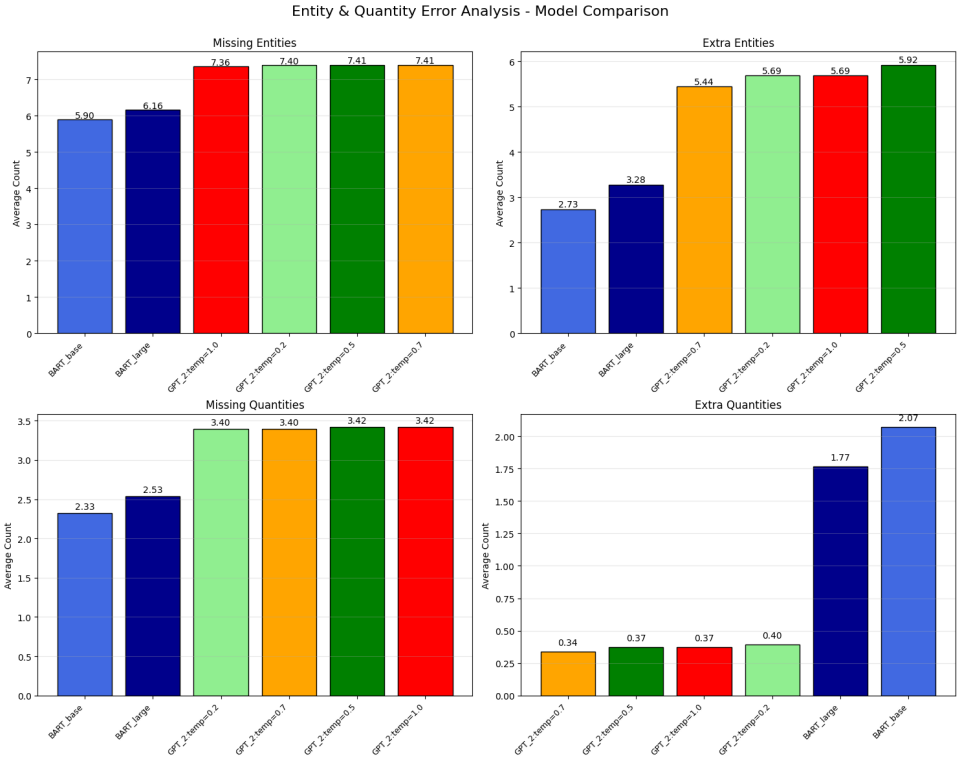

Figure 10: Missing and extra quantities comparison across models.

Model	Missing Entities	Extra Entities	Missing Quantities	Extra Quantities	Sample Size
BART_base	5.90	2.73	2.33	2.07	86
BART_large	6.16	3.28	2.53	1.77	86
GPT_2:temp=0.2	7.40	5.69	3.40	0.40	86
GPT_2:temp=0.5	7.41	5.92	3.42	0.37	86
GPT_2:temp=0.7	7.41	5.44	3.40	0.34	86
GPT_2:temp=1.0	7.36	5.69	3.42	0.37	86

Table 5: Entity & Quantity Error Comparison