The Deep Past Challenge: Translating Akkadian with Found Data

Series: Kaggle Competitions **Date**: January 2026 **Role**: Data Scientist & NLP Engineer **Tech Stack**: PyTorch, Hugging Face, OCR, Beautiful Soup

1. The Challenge: Resurrecting a Dead Language

The **Deep Past Challenge** presented a unique NLP problem: translating **Old Assyrian** (a dialect of Akkadian) into English. The source texts were 4,000-year-old clay tablets documenting trade, family law, and debt in Bronze Age Mesopotamia.

The Constraint

This was a classic **Low-Resource Machine Translation (NMT)** problem.

- **Training Data**: Only **1,561** document-level translations.
- **Morphology**: Highly complex agglutinative language (median word expansion ratio of 1.48x).
- **Domain**: Specific vocabulary (minerals, textiles, ancient cities) that modern pre-trained models (LLMs) hallucinate on.

The baseline models (mBART, NLLB) failed significantly because they had never seen this specific dialect.

2. Hypothesis: Data Quantity > Model Architecture

My primary hypothesis was that **model architecture matters less than data quantity** in this regime. No amount of hyperparameter tuning on 1,500 samples would generalize well. I needed to "find" more training data from the available unlabelled corpus.

The Hunt for "Found Data"

I analyzed the provided `publications.csv` (PDFs of academic papers) and `published_texts.csv` (metadata). I realized many of these contained the translations we needed, just buried in unstructured formats.

Experiment 1: The AICC Scraper Instead of trying to OCR complex layouts, I reverse-engineered the API of a known digital archive (AICC) mentioned in the metadata.

- **Action**: Wrote a focused scraper (`scripts/aicc_scraper.py`) targeting the JSON API endpoints.
- **Result**: Successfully retrieved **4,755** new high-quality translations.
- **Impact**: **400% increase** in training data size (1,561 → 6,316 samples).

---

3. Architecture: Multi-Level Curriculum Learning

With the expanded dataset, I faced a new problem: **Context Length**.

- Some tablets were short (receipts).
- Some were long treaties.
- Training on full documents caused OOM errors or poor convergence on long sequences.

The Strategy: Sentence vs. Document

I implemented a **Multi-Level Training Strategy** (`scripts/multi_level_trainer.py`).

1. **Phase 1: Sentence Alignment** I used `scripts/sentence_alignment.py` to map 1,213 aligned sentence pairs from the training set. This created a high-quality, short-context dataset.
2. **Phase 2: Curriculum Training**
    - **Step A**: Train on Sentence Pairs (Easy, high gradient signal).
    - **Step B**: Fine-tune on Full Documents (Hard, learns discourse structure).

Custom Transformer

Instead of just fine-tuning, I built a custom Transformer from scratch (`scripts/custom_transformer.py`) featuring:

- **ALiBi (Attention with Linear Biases)**: To handle variable length tablets better than sinusoidal pos embedding.
- **Noam Scheduler**: Standard for transformer convergence.
- **Gradient Accumulation**: To simulate large batch sizes on consumer hardware.

---

4. Evaluation & Error Analysis

A key part of the process was moving beyond a single loss number. I implemented a robust evaluation suite (`scripts/evaluate.py`) tracking:

- **BLEU & chrF++**: Standard MT metrics.
- **Geometric Mean**: The competition's specific metric.

Error Analysis Taxonomy

I didn't just look at the score; I categorized *where* the model failed:

- **Named Entities**: Did it mangle names like "Ashur" or "Kanesh"?
- **Numbers**: Were quantities of silver/tin preserved?
- **Omissions**: Did it drop sentences entirely?

This analysis revealed that the model struggled most with **Named Entities**, leading to a lexicon-based post-processing strategy using a 13,000-entry dictionary I compiled (`Phase 1.3`).

---

5. Key Takeaways

1. **Data Archaeology Pays Off**: The 4x data boost from scraping/matching outperformed any architectural tweak I tried. In low-resource NMT, **data engineering is the modeling**.

2. **Curriculum Learning Works**: Starting with simple sentences and moving to full documents stabilized training significantly compared to shoving full documents in from epoch 0.
3. **Specific Metrics Drive Progress**: Breaking down errors by type (Entities vs. Grammar) gave actionable insights that a flat BLEU score never could.