Translating Ancient Hittite with MT

Machine Translation using NLLB

Abstract

Hittite is one of the oldest written languages, spoken by the ancient Hittites with records dating as far back as the 17th century B.C.E. in what is now modern-day Turkey. However, the language died out around the 13th century B.C.E and relatively few records of the language have been uncovered, making Hittite a low data language. The issue is training a language model to translate written Hittite into written English. A difficult task for two main reasons: as mentioned, there is a data scarcity when it comes to labeled Hittite to English translations. Possibly the bigger issue though is the lack of language models that support fine-tuning for new languages.

1 Introduction

1.1 Background:

1.11 Hittite as a Language:

Hittite is an ancient language spoken by the Hittite people in modern day Turkey, between the 17th and 13th centuries B.C.E.. The language died out shortly after the collapse of the Hittite kingdom at the end of the bronze age. Like most ancient languages, all of what is known about Hittite stems from its written form rather than its spoken form, and therefore the written form is what will be discussed throughout the paper. Hittite is not a phonetic language, it is a logographic language, which means rather they don't use an alphabet that includes all graphemes which represent pieces of a morpheme or morphemes themselves, see 'a book' in which 'a' is a unit of meaning. Instead, the Hittite language uses multiple morphemes to represent a single sound; For example, the phrase "I can see you" could be represented by a single pictograph and be pronounced with a single syllable. The meaning of a word may also change depending on its position in a sentence. To complicate things further, the stone tablets found which contain Hittite cuneiform, are often a composition of many writing styles and dialects, each of which contain their own rules surrounding what is and is not a morpheme.

1.12 Difficulties in Transliteration:

A large obstacle faced by our team was consistency in the datasets published online. In our research we found that many of the datasets published online used their own specialized rules to encode the Hittite cuneiform into a representation using the English alphabet. Obviously, this stems from their being no clear equivalents between much of the cuneiform system and the English system, specifically when representing a single morpheme of cuneiform as a single morpheme in English letters. This has led many researchers to attach many supplementary symbols such as "(TUS)" or "(WA)" to a Hittite representation with very little overlap between datasets. These data set themselves use complex utf encodings and inconsistent ordering to attempt to bridge the lexical gap. As will be further discussed later in the paper, this complexity and inconsistency even within a single dataset makes it next to impossible to scrape these datasets programmatically. Worse still, as more information about Hittite is uncovered many of the old datasets have been rendered incorrect, further eliminating sources of data for our group to train on. Largely, the datasets that could be scraped and were grammatically sound were restricted to individual word pairs that consist of a single Hittite word paired with a sentence in English describing its meaning.

1.2 Designing the Test:

Unfortunately, many of the tests used to assess even low data language models are predicated on the language model having been trained on if not paragraphs then at least sentences. As an example, the model could be asked to fill in a blank correctly in a sentence. These methods were not available to us as we had no grammatical information only individual word meanings; Therefore, the test we selected asked the fine-tuned NLLB model to process a Hittite word it had not seen before (to prevent overfitting) and accurately predict its translation in English. To prove that this task was meaningful and more importantly feasible, we performed a test on our dataset. Importantly, because the only information available to the language model was individual Hittite words, to guess the English translation the model must use only information encoded in one word, rather than any kind of grammatical information. If this task is possible then there necessarily must be a correlation between the "stems" In the Hittite word and its meaning. This is obvious in phonetic languages like English, where words like chronograph contain the stem "khronos" which relates to time and "graph" which means to write or draw. If someone has not seen the word chronograph but knew about English word stems, they could still reasonably guess the meaning of chronograph. However, because Hittite words do not follow the same word structure conventions as Latin languages and the Hittite words themselves are a product of transliteration from the original cuneiform pictographs, it is not immediately obvious if the necessary morphosemantic relationship exists in our dataset.

To assert that this relationship exists we performed a correlation test between the Levenshtein distance of two Hittite words and the cosine distance of their translations in English. The reasoning behind this test was that if two Hittite words had a low Levenshtein distance then they also contained similar sub words or morphemes. Furthermore, if their translations had a smaller cosine distance, then we can assume those translations are similar in meaning. To perform this calculation, we found the Levenshtein distance between each word and every other word in the dataset and then paired that number with cosine distance between the mean vectors of their translations. These mean vectors were found by finding the mean vector of every word in a Hittite word's English description. The initial Pearson correlation test yielded a correlation coefficient of .007 and a p-value extremely close to zero. This indicates a very small linear correlation between the Levenshtein distance and cosine distance of two Hittite words and English translations. The extremely small p-value indicates that though the correlation was small the result was statistically significant. To improve the results we included synonyms, words with a small cosine distance to the target word, of each word in the translation to the mean vectors. This resulted in a correlation coefficient of .02 and a similarly small p-value, a significant improvement. Next, we omitted words in the data set that were particularly problematic and contained uncommon utf encodings or other noise. This yielded a correlation coefficient of .039 and a close to 0 p-value, a similar improvement. We then ran a spearman correlation test to see if there was some nonlinear relationship present which returned a coefficient of .065 and a p-value of $5 * 10^{-34}$, our best result yet. We concluded that with the small size of the dataset as well as the previously mentioned translation issues, the final test results proved there was a small yet statistically significant correlation between the characters used to represent a Hittite word and that words English translation, which further proved that the test we planned to run on our fine-tuned language model had a reasonable chance of proving that the model was able to learn, in part, the meaning of a Hittite words sub words.

2 Data Collection

To properly approach this problem of translating Hittite to English we need to build a dataset that would be sufficient for training a language model. There were many sources that we looked at to gather data to build our optimal dataset. The Oriental Institute of the University of Chicago has a resource called the Chicago Hittite Dictionary (CHD) which is a comprehensive dictionary covering the lexicon based off published Hittite texts. The dictionaries were available in PDF format for the L-N, P, and S Hittite words along with their meanings and the information of their origin. Our first approach was to scrape the PDFs and extract the Hittite word and its associated translation in English. The task was simple, however there were many problems we encountered while trying to scrape the PDFs. First, the PDFs were not text-based but rather they were image-only PDFs meaning there were just scanned or photographed images of pages without a text layer. With the assistance of the Data Analysis tool in ChatGPT 4, I prompted the tool to extract a Hittite word and its corresponding English translation. To get around the image-only PDF problem the tool utilized Optical Character Recognition to convert the images back into text. Additionally, to help accurately scrape the data we need from the PDFs I prompted the tool a format of how the words and translations were in the PDF. Despite these instructions and careful prompting there were too many variations in the format of the Hittite word and its corresponding translation. For instance, one Hittite had multiple English translations based on the context of how the word was used as a noun, verb, adjective and more. To lessen the load of the OCR we attempted to feed the data analysis tool with a few pages of the dictionary at a time which contain one or two Hittite words per page. However, the tool was still not able to properly extract the Hittite word and its translation properly. Luckily, another source known as the Hittite Lexicon, which included almost 1500 Hittite words and their translations was scraped successfully and we had a small but decent dataset size. Another source of Hittite data we encountered was by the Linguistics Research Center at the University of Texas at Austin. The center has a series of 10 Lessons on Hittite based on important Hittite documents and texts the University found. Within the lessons were paragraphs of Hittite and their corresponding English translation. We again tried to devise a way to scrape the data and format the data with the Hittite word and its translation. However, there were too many variables in the organization of the paragraphs, words, and translations. When we generated a bit of data there was too much noise, and the data was filled with arbitrary characters and letters that were irrelevant to our dataset. Lastly while reviewing the Orient Institute of the University of Chicago's website, we were able to access more data through their electronic dictionary. The dictionary was in a java program which when successfully queried generated a list of all Hittite words they had along with their translations. Following this we were successfully able to add around 2000 more Hittite words to our dataset. Following this we attempted to clean the dataset.

The original lexicon data was very messy (containing lots of special characters in seemingly random places), so we used regex to parse and clean the data. Example regex used:

```
'(\w+[;]?[^.]+)[.]{1}'
```

Full regex code and Lexicon files can be seen on our main GitHub repo.

3 The No Language Left Behind Model (NLLB)

3.1 Model Introduction

The attempt of machine translation is done by leveraging Meta AI's No language Left behind (NLLB) MT model. NLLB is an open-source model capable of translating between 200 languages. Some of these languages are even low-resource languages- like Asturian and Luganda, meaning that the training was done on relatively small dataset sizes for those languages. Meta uses this technology for their translation interfaces on Facebook and Instagram. It also shows use on Wikipedia for content translation.

Specifically, we use the 600M parameter version of their model (aka distil-Nllb200) that's available on huggingface.co. This model was chosen because it is one of the most powerful open-source MT models on the market. Even more enticing is the fact that it was trained on multiple low-resource languages.

3.2 Model Implementation

Our implementation emphasizes the two main aspects for which we selected this model:

training on low resource datasets and leveraging the source code data to create new functionalities with the model.

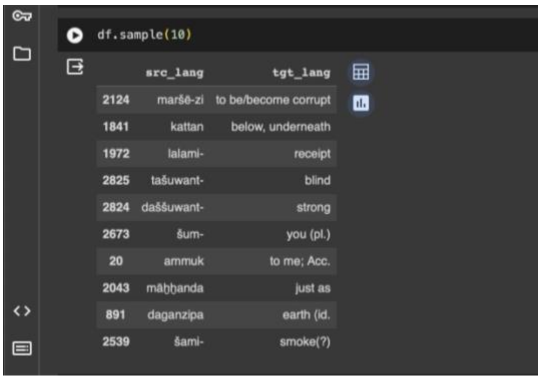
Calling it *our* implantation, however, is a little bit audacious. The google colab in which we implement our experiment is an adaptation of a similar problem, solved by David Dale from Meta AI itself. In his original colab, he tackles the issue of creating a new language tag for the NLLB model (a task that involves slightly modifying the source data) to add a new low-resource language called Tyvan (a Russian dialect). Many of his issues faced paralleled our problem, making this the perfect example to base our implementation off.

4 Experiment/Results

4.1 Data Splits

The model is trained on a labeled dataset of Hittite words and English translations.

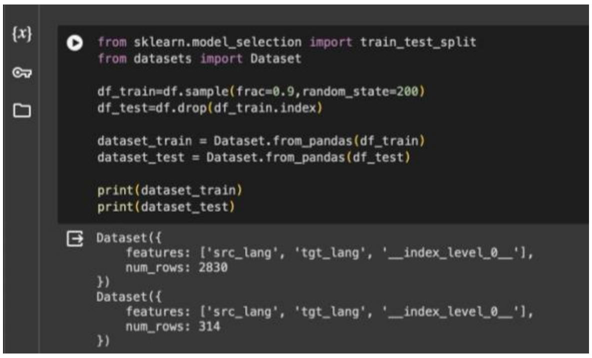
Figure 1: Hittite words with (English) Translated definitions as labels.



```
df.sample(10)
```

	src_lang	tgt_lang
2124	maršē-zi	to be/become corrupt
1841	kattan	below, underneath
1972	lalami-	receipt
2825	tašuwant-	blind
2824	daššuwant-	strong
2673	šum-	you (pl.)
20	ammuk	to me; Acc.
2043	māḫḫanda	just as
891	daganzipa	earth (id.
2539	šami-	smoke(?)

We use pandas to create our data frame objects and our splits.



```
from sklearn.model_selection import train_test_split
from datasets import Dataset

df_train=df.sample(frac=0.9,random_state=200)
df_test=df.drop(df_train.index)

dataset_train = Dataset.from_pandas(df_train)
dataset_test = Dataset.from_pandas(df_test)

print(dataset_train)
print(dataset_test)

Dataset({
    features: ['src_lang', 'tgt_lang', '__index_level_0__'],
    num_rows: 2830
})
Dataset({
    features: ['src_lang', 'tgt_lang', '__index_level_0__'],
    num_rows: 314
})
```

Figure 2: Data is split into a train and test set with pandas:

We tested with two main dataset splits:
90% train/ 10% test, with unrandomized split.
90% train/ 10% test, with randomized split.
(Smaller training set splits were not used due to the small amount of data available.)

4.2 Tokenizer Metrics

We test the NLLB tokenizer to see how well it does at tokenizing our data. Specifically, because the NLLB tokenizer was trained on normalized text, we want to see how well it does at tokenizing our Hittite words before and after text normalization.

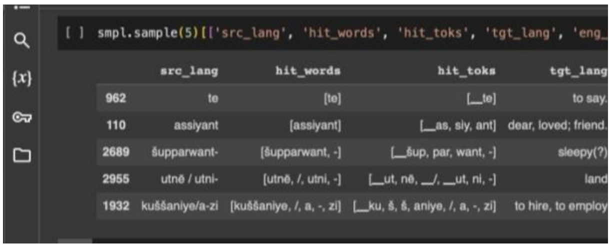


Figure 3: Hittite words to tokens with NLLB tokenizer

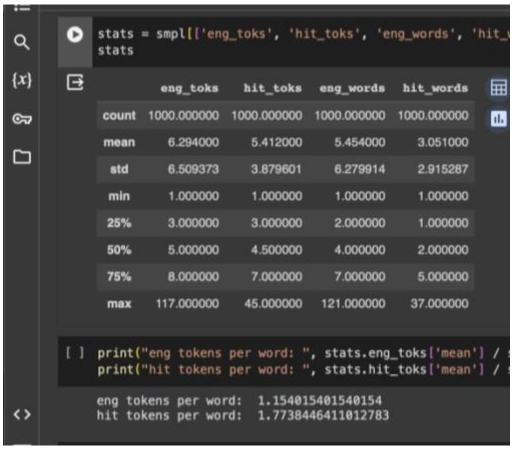


Figure 4: Average tokens per word

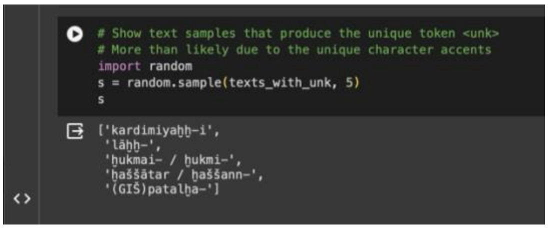


Figure 5: Examples of Hittite words that produce <unk> tokens

Given these samples we can see that the probable cause of these <unk> tokens is the unique character encodings. We perform text normalization and test the tokenizer again.

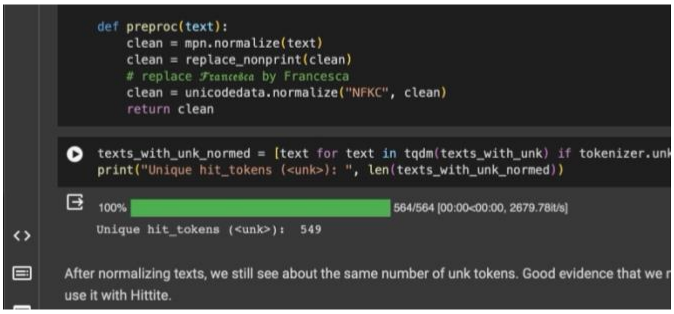


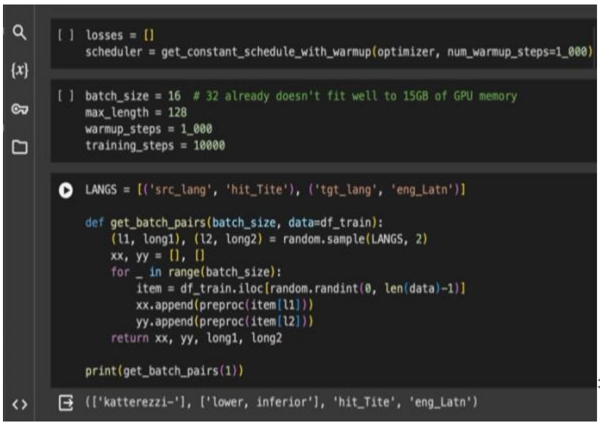
Figure 6: Number of <unk> tokens still present after text normalization.

Since this is like the result we got before normalization, this shows that simply normalizing the text will not be enough to remove the <unk> token problem. In fact, it's likely evidence that we would need to train a new tokenizer to properly process all our data. (Something we were not able to achieve in our implementation due to technical constraints)

4.3 Adding a New Language

To fine-tune the NLLB model on a new language, we must create what's called a new 'language tag' for the model. This language tag is essentially the class token that the model will use to encode its output. We place the new language tag near the Turkish language tag, as it's the language that we agreed would be most like Hittite (at least out of the available languages). This starts the weights for our new language tag at the weights of the Turkish language tag.

Training loop was done with these parameters and hyperparameters:



```
[ ] losses = []
    scheduler = get_constant_schedule_with_warmup(optimizer, num_warmup_steps=1_000)

[ ] batch_size = 16  # 32 already doesn't fit well to 15GB of GPU memory
    max_length = 128
    warmup_steps = 1_000
    training_steps = 10000

LANGS = [('src_lang', 'hit_Tite'), ('tgt_lang', 'eng_Latn')]

def get_batch_pairs(batch_size, data=df_train):
    (l1, long1), (l2, long2) = random.sample(LANGS, 2)
    xx, yy = [], []
    for _ in range(batch_size):
        item = df_train.iloc[random.randint(0, len(data)-1)]
        xx.append(preproc(item[l1]))
        yy.append(preproc(item[l2]))
    return xx, yy, long1, long2

print(get_batch_pairs(1))

(['katterezzi-'], ['lower, inferior'], 'hit_Tite', 'eng_Latn')
```

Figure 7: Model hyperparameters

Training for 10,000 steps on a V100 took approximately 45 minutes.

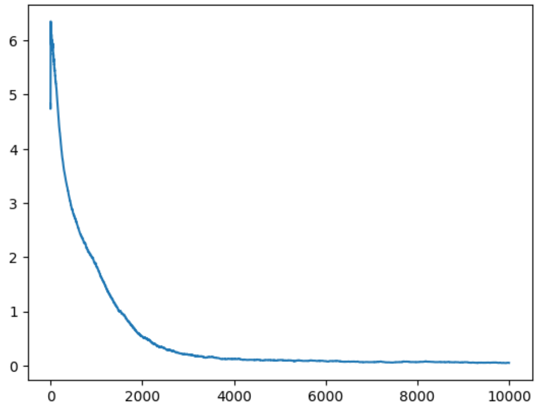


Figure 8: Training steps (x) versus Loss (y)

4.4 Model Metrics

The metric we used to test our model is called the chrF2++ score. The chrF2++, or character-level F-score, is a machine translation metric that evaluates the similarity between the predicted translation (or machine translation), and the test translation (or reference translation) using character n-grams (instead of word n-grams).

We tested on the training set for English to Hittite and Hittite to English respectively. We didn't use a dev set, because no hyperparameters were tweaked on the test set. In other words, the test set is completely unseen in our experiment (until we test for metrics).

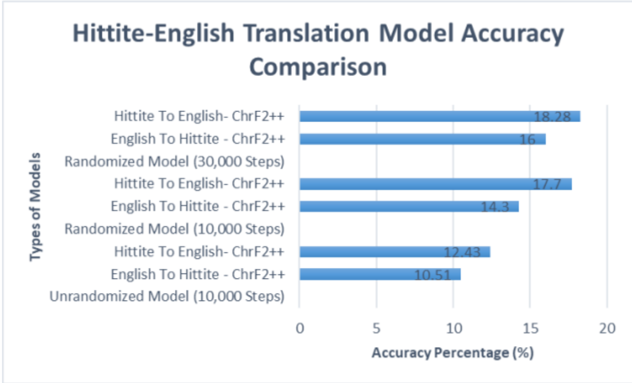


Figure 9: ChrF2++ Scores

Although these scores are relatively low by MT standards, provided the relatively small dataset and lack of a tokenization on certain characters, we believe these metrics are significant enough to show that our model is properly fitting the data.

tgt_lang	hit_translated
witchcraft, magic.	witchcraft, sorcery, spell.
to envy.	to mount, set, frame, plate; to equip.
to eat well, take care of oneself.	to love, to be friendly, to make peace.
saw.	tongue; talk, rumors (id.
crew.	garrisoning.
to settle.	to make ases.
love.	love, friendship.
here! – come! (used as imperative of uwa- "to come"; §164 2b).	sin.
to laugh.	to hang.
guard of a temple.	administrator, deputy.
to dare, risk.	to commit, impose.
to do a miracle.	wisdom; care.
miracle.	inclination, disposition.
postpos.	adv.
town (id.	to sprinkle.
crushed, spoiled, rotten (part.	by word of mouth (id.
born (part.	grand-son and grand-grand-son? (§89b).
handful (quantity).	courtyard.
alive; raw (meat).	noun of unknown meaning.
adv.	truly, really, indeed
belt.	buckle (id.
scout.	aggrieved one by offense.
to close; to block.	to track down, pursue, chase; to report.
evil, nastiness (id.	badness, evil disposition.

Figure 10: Example output of Hittite words translated into English (on the right) and their true reference value (on the left)

Citations

1982. The Hittite dictionary of the Oriental Institute of the University of Chicago

2023. Hittite Grammar – Hittite lexicon. Oliver Lauffenburger.

Harry A. Hoffner Jr, H. Craig Melchert. 2008.A Grammar of the Hittite Language. Eisenbrauns,1st edition.

David Dale. 2023. How to fine-tune a NLLB-200 model for translating a new language

NLLB Team, Marta R. Costa-jussà, James Cross, Others. 2022. No Language Left Behind: Scaling Human-Centered Machine Translation. arXiv:2207.04672

References

Our GitHub:

https://github.com/rfeinberg3/NLPFinalProject

Our Collab:

https://colab.research.google.com/drive/1fmJe9EuumIo-uwfW4Pp3hgyz3SviomaQ?usp=sharing

"No Language Left Behind: Scaling Human-Centered Machine Translation":
https://research.facebook.com/publications/no-language-left-behind/

NLLB model on huggingface:
https://huggingface.co/facebook/nllb-200-1.3B

Fairseq Nllb GitHub:
https://github.com/facebookresearch/fairseq/tree/nllb

How to fine-tune a NLLB-200 language model for translating a new language (by David Dale):
https://cointegrated.medium.com/how-to-fine-tune-a-nllb-200-model-for-translating-a-new-language-a37fc706b865

David Dale's original Collab:
https://colab.research.google.com/drive/1bayEaw2fz_9Mhg9jFFZhrmDlQlBj1YZf?usp=sharing

chrF2++ score:
https://machinetranslate.org/chrF