Entity Aware Machine Translation

Shannon Rumsey
sarumsey@ucsc.edu

Darian Lee
daeilee@ucsc.edu

Abstract

To mitigate the effects that a named entity can have during translation, we implement a named entity translation lookup, paired with code-switching to reintegrate it back into the original source sentence. With the already translated named entity, this sentence is fed into a model to be translated as a whole. Using a Seq2Seq and Transformer model, we were able to achieve excellent results with COMET scores of 63 and 68 respectively, making our models comparable to state-of-the-art models despite a difference in size.

1 Background and Related Work

Our model architecture and methodology are in formed by previous research. Our code-switching approach is inspired by Song et al. (2019) and Whitehouse et al. (2023), which demonstrate the benefits of replacing entities with their translations during preprocessing to achieve more natural translations and capture nuanced meanings. For our baseline model, we relied on Wu et al. (2016) for insights into how a state-of-the-art Seq2Seq model can be built. Additionally, our final decoder-only model was built based on the architecture invented in Vaswani et al. (2023).

2 Task

We set out to create a machine translation system that can accurately translate sentences containing named entities (e.g., movie titles, people, corporations, etc.). Translating sentences with these entities poses a challenge, as entities are often culturally localized and not always translated literally. Some have direct one-to-one translations (e.g., *The Godfather* translates directly to *Il Padrino* in Italian). In contrast, others may have different translations (e.g., The movie *Eternal Sunshine of the Spotless Mind* translates to *Se mi lasci ti cancello*, or "If you leave me I will erase you"). These inconsistencies can cause models to make erroneous mistakes when they would otherwise perform well.

The original SemEval dataset includes "Source" and "Target" columns, where the "Source" column contains English text, and the "Target" column provides corresponding translations. Separate datasets were available for each language, but for our analysis, we combined the Spanish, Italian, and French datasets. These languages were chosen to fine-tune our model specifically for Romance languages.

Source Text	Target Text
Was The Notebook released in 2004?	Le pagine della nostra vita è uscito nel 2004?

Table 1: Example from the French dataset. Notice "The Notebook" translates to "Le pagine della nostra vita" in French, which means "The page of our life."

Our models differ slightly in implementation, but both take in an English source sentence and outputs a translation in one of three possible target languages. Each input sentence includes one or more named entities which have been pre-translated into the target language. The objective is to produce a natural translation of the English sentence while ensuring the entity is accurately conveyed in the target language.

3 Approach

3.1 Entity tagging and Mapping

Entities were tagged and mapped using the pretrained model `Babelscape/wikineural-multilingual-ner` `tokenizer and entity detector`. First, we checked if the number of entities detected in the source and target sentences were the same. If the counts did not match, the entity was marked as a potential problem for further review. If the frequency matched, we then checked whether the

entities in corresponding positions had the same tags (e.g., "person," "location", etc.). If the tags aligned, we verified whether the source entity consistently matched the same target entity across all sentences. When this condition was met, the translation was saved in a lookup table.

If the tags aligned but the source-to-target entity alignment was inconsistent, we analyzed the frequency of possible translations. If one translation occurred more frequently than the others, it was saved in the lookup table. Otherwise, the entity was added to the list that contains the problematic entities. Finally, for all entities in the aforementioned list, we used the Wikipedia API to fetch data about the topic from the relevant Wiki page. This page was then translated into the target language, and the title of the translated page was used as the correct translation for the entity.

3.2 Code switching

After constructing the dictionary lookup table, we replaced entities in the English sentences with their translations in the target language—a process commonly referred to as code-switching. This approach, inspired by the linguistic phenomenon of switching between languages within a sentence, has proven particularly effective for entity detection in translation tasks. Research has demonstrated that code-switching yields more natural and accurate translations compared to approaches where entities are translated separately or during model training (Song et al., 2019) (Whitehouse et al., 2023).

The key advantage of code-switching lies in its ability to integrate translations seamlessly into the sentence structure, providing context for the surrounding text. This encourages the model to inherently recognize and handle entities within the sentence because it is trained to identify which parts should remain unchanged in the output, which will be the pre-translated entities. The result is a more fluid and contextually appropriate translation that aligns more closely with natural language use.

4 Models

Both models use dynamic embeddings that are learned through the model and stored using PyTorch's `nn.Embedding()` module. The transformer model utilizes an additional set of embeddings that represent token position in a sequence.

4.1 Seq2Seq with Attention

Based on the nature of machine translation, a model must be able to accurately map from one language, in this case English, to the target language. Many times, an English sentence's equivalent in another language will have a different length (e.g., "I will pick you up" in Spanish is "Te recojo"), so any model created must be able to handle varying sequence lengths.

One model that tackles this problem is the Seq2Seq model, which has an Encoder-Decoder architecture (Sutskever et al., 2014). The Encoder model encodes an input sentence with variable length into fixed-length embeddings that convey the meaning of the input. The decoder model uses these embeddings and predicts the appropriate outputs, in this case, a translated sentence.

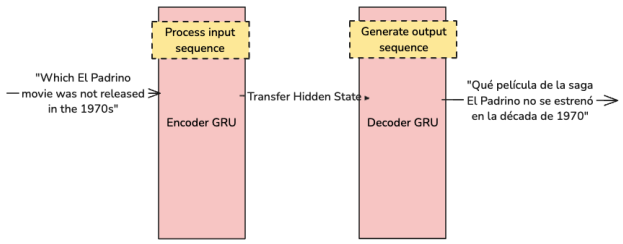

Figure 1: Overview of the encoder-decoder structure in the Seq2Seq model.

The Seq2Seq model presented in this work consists of two unidirectional GRU models with an added attention mechanism in the decoder model Team (2023). The input embeddings are fed into the encoder, which produces a final hidden state, also called a context vector, that summarizes the model's findings regarding the inputs (Voita, 2023). The encoder learns representations of the input sentence that better reflect the output sentence, creating updated embeddings. The final hidden state and embeddings are inputted into the decoder model which further learns from them and creates the final outputs. In our decoder model, attention is added to allow the decoder to focus on different parts of the input vector at different time steps (Bahdanau et al., 2016). This makes the model more flexible in that it can look at any stage of the inputs (e.g., any word) that best helps transform the outputs, regardless of order.

PyTorch's `nn.GRU` model is used to create the encoder and decoder model. Bahdanau attention is

used in the decoder, which calculates the alignment scores by adding the hidden state from the decoder to the hidden state from the encoder. A softmax function is applied to this sum in order to calculate the attention weights, which then have the weighted sum taken to determine the amount of attention to give to the particular sequence.

The baseline model to be experimented with has 0.1 dropout, and a hidden size of 256. Student forcing was used in order to ensure consistency with training and inference time results.

4.2 Transformer

The transformer followed a decoder only architecture (Guo et al., 2024) with an embedding size and hidden dimension of 400, 5 attention heads with 5 layers, and a vocabulary size of 25689 after subword tokenizing using BPE. This all totaled to a parameter size of over 30 million. The model employed the AdamW optimizer with a learning rate of 0.0001 and a dropout rate of 0.3 to mitigate overfitting. The final translation model took in an input in the form of "<source tag> source sequence –> <target tag> target sequence".

Below is an example (simplified for readability; actual inputs used numerically encoded sub word tokens):

$x = [\langle\text{en}\rangle, \text{“was”}, \text{“Boris”}, \text{“Eltsine”}, \text{“the”}, \text{“first”}, \text{“president”}, \text{“Russie”}, \text{“?”}, \rightarrow, \langle\text{fr}\rangle, \text{“Boris”}, \text{“Eltsine”}, \text{“était-il”}, \text{“le”}, \text{“premier”}, \text{“président”}, \text{“de”}, \text{“la”}, \text{“Russie”}, \text{“?”}]$

$y = [\text{“was”}, \text{“Boris”}, \text{“Eltsine”}, \text{“the”}, \text{“first”}, \text{“president”}, \text{“of”}, \text{“Russie”}, \text{“?”}, \rightarrow, \langle\text{fr}\rangle, \text{“Boris”}, \text{“Eltsine”}, \text{“était-il”}, \text{“le”} \text{“premier”}, \text{“président”}, \text{“de”}, \text{“la”}, \text{“Russie”}, \text{“?”}, \langle\text{END}\rangle]$

The output was masked to only include the tokens between the target tag and the end tag, and our performance metrics were calculated only on this.

5 Experiments

5.1 Seq2Seq with Attention

The metrics used to evaluate hyperparameters are training and validation loss, and the average COMET and METEOR score over training and validation. All of the Seq2Seq experiments were performed on a subset of the training data. The final model was trained and tested using all of the data. Each experiment used the baseline model with 1 layer, 0.1 dropout, and a hidden size of 512. Incorporating an additional layer showed promise of slightly better performance, its extended runtime diminished any minor advantages from using it, thus only one layer is used in the model. The COMET and METEOR scores calculated during the experiments were on the validation set.

An experiment was done to determine if code-switching would improve the model's ability to translate sentences. To do this, we looked at model performance on entities when they were separated by spaces and underscores. The way that the entities are separated influences their tokenization, determining whether they are stored as a single token referring only to that entity (similar to how they would be represented using a tagging approach for entity translation), or as a sequence of ambiguous tokens (how they are stored under the code switching method). Using underscores ensures that the model learns the entire named-entity as one object, whereas with spaces, each word is learned individually. While the results were close, we found that the model was better when using code-switching. This experiment influenced our decision to incorporate code switching into the transformer model.

	Train Loss	Val Loss	COMET	METEOR
No Underscores	1.91	3.78	0.63	0.59
Underscores	1.97	4.22	0.60	0.54

Table 2: Model performance on entities that contain underscores vs none.

Another experiment was done to observe the differences in performance when the data was tokenized as whole words versus sub-words. The Seq2Seq model showed preference for whole word tokens, however, sub-words were still used in the transformer model. This was done because the transformer has a larger vocabulary size due to the addition of the SQuAD data, greatly impacting computational resources and runtime. As a way to mitigate these effects, sub-words were used to effectively shrink the vocabulary.

	Train Loss	Val Loss	COMET	METEOR
Whole Words	1.91	3.78	0.63	0.59
Sub-Words	2.77	4.52	0.32	0.29

Table 3: Model performance when trained on sentencepiece sub words vs whole words.

5.1.1 Results without Pretraining

The model without pretraining preformed very poorly, with nonsensical outputs composed of mostly <PAD> tokens and tokens belonging to dif-

ferent languages than the target. Increasing model size or number of epochs in training caused minimal improvements. After consulting with prior research we determined that the solution was to pretrain the model on similar data in order to teach it the structure of the languages involved.

5.1.2 Pretraining

The model was pretrained on the questions from the Stanford Question and Answer (SQuAD) dataset. This dataset was chosen as it was available in each of our target languages (English, Spanish, Italian, and French) and the style of sentences closely matched that of our translation data, composing mainly of Entity-related questions like "When did Beyonce release Dangerously in Love?". The SemEval and pretrain data was sub word tokenized and put into a dataset together in order to ensure that they would have the same vocabulary size and be encoded in the same way. For pretraining, the model took in multilingual inputs containing a language symbol followed by a random chuck of 30 tokens from that language's corpus, which was composed from concatenating all the questions in the SQuAD dataset for that language. The model was then evaluated on its perplexity for predicting the next word, and was stopped at a final best validation perplexity of 58.45.

Language	**Sample Sentence**
English	What was the first Greek building in England?
Spanish	¿Cómo se llama el país en las sesiones de las Naciones Unidas?
Italian	Dove ha fatto Mongke Khan attaccare la dinastia Song?
French	Quand le professeur Stefan Grimm a-t-il été retrouvé mort?

Table 4: Sample sentences from each language's version of the SQuAD Dataset. Notice each is a question containing entity names (ex. United Nations, Song Dynasty, Professor Stefan Grimm) similar to the semEval data

Pretraining Details	**Results**
Dataset	Stanford Question and Answer (SQuAD)
Languages	English, Spanish, Italian, French
Input Format	Language symbol + rand. seq. of 30 tokens
Evaluation Metric	Perplexity (next word prediction)
Best Perplexity	**58.4468**

Table 5: Pretraining results and details.

5.1.3 Finetuning on Translation Task

After pretraining, the best model was loaded and used as a starting point for finetuning our model to our specific translation task with the inputs and outputs mentioned earlier. Additional positional embeddings were created and appended to the pretrained model to account for differences in max sequence length, which was 30 in the pretrain dataset and over 100 in the training set. The training sequence lengths were expected to be longer as they included sentences concatenated with their translations, where as in the pretrain data, each input was in one language only. The model was trained for 50 epochs, with validation METEOR score and loss measured at each epoch and used for model selection. The best model was found after 38 epochs. After loading the best model, we tested it with the test set by evaluating cross-entropy loss, perplexity, entity-level precision, recall, and F1, METEOR, and COMET. The results of this analysis are summarized in the results section.

6 Evaluation Metrics

Our models were evaluated using broad translation metrics such as METEOR, COMET, and Perplexity (for pretraining the transformer model), along with entity-specific metrics like entity-specific F1, precision, and recall. METEOR and COMET were chosen for overall translation quality because they incorporate semantic similarity, unlike BLEU-based methods that focus only on n-grams. COMET measures semantic similarity at both word and sentence levels, while METEOR considers word alignment, stemming, synonymy, and word order. These metrics allow for a more nuanced evaluation, taking into account both the meaning and structure of the translation, which is particularly important for our task, where entity preservation is crucial. The entity-specific metrics were calculated by comparing entities detected in the predicted and target translations to ensure correct entity translation, with F1, precision, and recall chosen to ensure even rare entities were properly translated and evaluated.

7 Results

7.1 Seq2Seq

After fine tuning, we found our best performing model to have a dropout rate of 0.1 and a hidden size of 512.

The final Seq2Seq model achieved a COMET score of 0.55 and METEOR score of 0.42 on the

Dropout	Hid. Size	Train Loss	Val Loss	COMET	METEOR
0.1	256	1.91	3.78	0.63	0.59
0.1	512	1.55	3.60	0.63	0.61
0.1	1024	2.72	3.73	0.56	0.58
0.2	512	1.93	3.65	0.62	0.61

Table 6: Hyperparameter values experimented with and their effects on loss, COMET, and METEOR.

Train Loss	Val Loss	COMET	METEOR	F1
1.78	3.62	0.52	0.43	0.21

Table 7: Final Model results on the validation and training set.

COMET	METEOR	F1
0.51	0.42	0.22

Table 8: Final Model results on the test set.

test data. The model had roughly the same performance on both the validation and test data, but did slightly better with named entity recognition on the test data, as seen by the F1 score in Table 8.

Below is an example of a prediction created by the model and how it compares to the actual translation. The model does fairly well and only misses a few words, which seems to be the case for many of the predicted translations.

pred = [“Quando", “ebbe", “inizio", “la", “costruzione", “dellArco"]
actual = [“Quando", “ebbe", “inizio", “la", “costruzione" “<UNK>", “della", “porta"]

7.2 Transformer

The final transformer received a METEOR score of .6 with a COMET score of .68. This is even higher than Multilingual BART (mBart) which obtained a COMET score of .67. Considering that most state of the art models receive COMET scores between .7 and .8, we are very impressed with our models score. Unfortunately, the entity specific scores were less good than we expected, with entity-specific F1 precision, and recall scores between .3 and .32. Due to the time limits on this project, we were unable to experiment with more methods of boosting these scores, although we plan to explore this in future research. There are also limitations in our evaluation of entity-specific scores which may have caused the reported scores to be artificially low or high compared to the actual results. This will be discussed further in Limitations.

Metric	Results
Testing CE Loss	1.1749
Testing Perplexity	5.5528
Testing METEOR score	0.5979
Testing COMET score	0.6819
Entity-level F1	0.3379
Entity-level Precision	0.3545
Entity-level Recall	0.3242

Table 9: Testing results. *Note: Perplexity was measured only on the translated text, and thus was expected to be significantly lower than in a typical language model due to the fact that the context of the source text was provided.*

8 Conclusion

We set out to develop an entity-aware machine translation model. To achieve this, we implemented code-switching to preserve the correct entity translations within our model. We then designed two models: a Seq2Seq model using a GRU and a decoder-only transformer model pretrained on the SQuAD dataset. These models achieved high COMET scores of 0.63 and 0.68, making them comparable to smaller state-of-the-art models like Multilingual BART (mBART). However, our models had lower-than-expected entity-specific evaluation scores, with an F1 score of 0.338. This score may not accurately reflect the entity translation performance, as it was influenced by potential limitations of the pretrained NER model, particularly its tendency to miss certain entities or misclassify noise words as entities. We plan to explore different model architectures and approaches to address these issues and improve entity-specific performance.

Limitations

The primary limitation of our approach lies in how the entity-specific evaluation metrics were calculated. Both the target and attempted translations were processed using the same NER pretrained model that we utilized during our code-switch analysis. However, it is important to note that prior experience with this entity detector revealed that it is not always accurate, and can miss entities or incorrectly label non-entities as entities. As a result, the reported scores could have been positively or negatively affected if a significant number of entities were misidentified.

After detecting the entities, we compared the set of entities in the true and attempted translations to calculate precision, recall, and F1. This method

only considered whether each entity appeared at least once in each sentence, without accounting for duplicates or misplacement of entities. However, we believe this limitation did not significantly impact our results. Our other evaluation metrics, such as COMET and METEOR, which focus more on overall translation quality, are sensitive to these types of errors. If they had occurred in our model, these metrics would likely have reflected them.

Another limitation of our approach was the architecture of the final model, specifically the decision to use a decoder-only transformer instead of an encoder-decoder architecture. We chose a decoder-only transformer due to its wide use and popularity in translation tasks, its scalability, and the fact that we had already explored an encoder-decoder architecture in our Seq2Seq model. However, after obtaining relatively low entity-specific evaluation scores, we conducted further research and realized that, for entity-aware machine translation, having an encoder state into which entity-specific information can be injected is highly beneficial for improving entity-specific metrics. Since a key goal of our task was the correct prediction of entities, we plan to experiment with this approach in future work.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. Neural machine translation by jointly learning to align and translate.

Shoutao Guo, Shaolei Zhang, and Yang Feng. 2024. Decoder-only streaming transformer for simultaneous translation. *arXiv preprint arXiv:2406.03878*. Accepted to ACL 2024. 14 pages, 10 Tables, 5 Figures.

Kai Song et al. 2019. Code-switching for enhancing nmt with pre-specified translation. *ACL Anthology*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks.

PyTorch Team. 2023. Translation with a sequence to sequence network and attention. https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html. Accessed: 2024-12-07.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need.

Lena Voita. 2023. Seq2seq and attention. https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html. Accessed: 2024-12-07.

Chenxi Whitehouse et al. 2023. Entitycs: Improving zero-shot cross-lingual transfer with entity-centric code switching. *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6698–6714.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.