A Small but Deep Model for Translating English to Spanish

Thomas Cholak

06 May 2024

Abstract

Machine learning is a popular solution to the task of machine learning as it allows us, when it comes to algorithmic natural language translation, to be able to translate between a source and target language without the need of creating an extensive list of complex rules. My project followed the process of utilizing a transformer-based architecture to encode, train the machine learning model, and then decode the results into the target language. To create this dataset, I used a popular dataset from Kaggle with over 100,000 sentences, and pre-processed the data with the standard "[start]" and "[end]" tokens, as covered by the textbook [Qin21].

The model was broken into the standard 70% training data, 15% testing data and 15% validation data for my dataset. 83,276 sentence pairs were used for training, with 17844 sentence pairs used for both testing and validation, respectively. The data was then encoded with positional embeddings before being fed into a transformer-based architecture and was then decoded (also using a Transformer).

After running my model for over an hour of training time on GPU, and utilizing a full 30 epochs, my model reached convergence at around 70 percent accuracy on the validation data set. My model was accurate for the most part, but sometimes will miss a word or will not recognize a token. Future plans for my model will include finding a way to host the model on my website so that visitors to my web page will be able to interact with the model and possibly add to the dataset as well if they choose to opt-in.

1 History of Machine Learning Translation

Before vectorization and deep learning models were popularized, the process for translation was extremely complex and often was inaccurate.This is because they relied on a host of rules and utilized vast dictionaries of words regarding both the source and target language. This made language translation extremely slow due to the enormous amount of rules required and it was difficult to adequately translate words since the model would often be unable to determine context and would instead have to analyze all words in a sentence in order to determine what rule to utilize. This often meant words were translated one-for-one and then effectively rearranged into the 'correct' order that was determined by the algorithm.

Eventually, around the 1980s and 1990s, it was discovered to be much more efficient to just feed a plethora of various examples of the same language in two different languages in order to train the model much better than previous models. This is because the model is then able to learn the rules on its own instead of having to manually write out a complex series of multiple rules. This meant that researchers no longer had to continually modify long lists of rules and could instead just retrain the model over and over with different datasets and different rules of data pre-processing. This was when phrase-based translation really took off [Pes18].

1.1 The Modern Age and Future of Translation

In the modern age, translation is done most often using large neural networks. Convolutional neural networks are often unable to work in regards to a non-image dataset so instead a residual neural network is utilized. Residual neural networks are very effective for constructing deep models and work to highlight the work of multiple layers. Additionally, bi-direcitonal residual networks then replaced those previous models, as bi-directional layers are able to analyze text in both directions. This served to better highlight the relationship of a word utilizing the words on both sides of it within a sentence.

The future of this technology would be the development of instant speech translation, in order to effectively act as a middle-man between people who may not share a common language. This will allow people of different cultures and backgrounds to communicate as much as possible in order to better facilitate and exchange ideas without having to wait too long for a translation or having to utilize typing. This technology will also make it easier for people to learn foreign languages since they will be better able to practice pronunciations much easier.

2 Project Summary

Notice: the link to the deep learning model I constructed can be found here.

2.1 A Brief Introduction to the Model

For my project I decided to make a natural language translator which is able to interpret sentences in English and translate the sentences accurately into Spanish. Language translation is a very challenging task for traditional computer algorithms to accomplish due to how vague some meanings of words can be. For example in the sentences: "Look *right* over there," and "Take a *right* at the next road," the word 'right' is spelled the same in both circumstances but the meaning is entirely different due to the context. Deep learning is useful for being able to solve complex problems which traditional algorithms are completely unable to solve on their own.

2.2 What are the Differences Between English and Spanish?

For this problem, we need a large amount of data in order to adequately train the model. In Spanish especially, a large part of the language are 'verbs' which are then conjugated in order to create different meanings. For example the verb "bailar" can be conjugated as "bailo" to mean 'I dance' or "baila" to mean 'he/she dances." This effectively means that there are a plethora of words in Spanish which mean the exact same word in English. For this it's important for the model to understand that some words can have the same meaning as one another and that it's not a one to one translation of every word from English to Spanish.

In addition, the sentence structure of English is also different compared to that of Spanish. In Spanish, verbs will often come before nouns while in English this is often not the case. This is why our model will have to be capable of following word order when performing translation tasks since it means the words will effectively need to be rearranged when we build the sentence in the target language. So this means the model will need to have some sort of method for keeping track of chronology rather than the classic *bag-of-words* approach to finding a solution to the dataset which is unable to track word order.

2.3 The Positional Embedding Approach

Instead of a bag-of-words approach, in my project I utilize positional embeddings in order to maintain word order. For this, each word in a sentence of both languages is tokenized and then encoded using a positional embedding function. The design of this function is so that each word maintains its original index when it's encoded. Since this index is maintained, the deep learning model is always able to keep track of these indices when performing its training. Using these numbers, the model is able to capture relationships between words in a structured space; this is what makes it possible for maintaining the overall word order.

Another issue is decoding the model. When first constructing my model, I was able to train the model in order to capture these relationships with an overall high accuracy but finding a way to convert these words back into the original Spanish and English text wasn't possible with my original model. Using transformer-based architecture we're able to create respective encoding and decoding architecture using custom methods. Alongside this model architecture, we want to utilize masking in order to increase the overall efficiency of the overall model.

```
({'encoder_inputs': <tf.Tensor: shape=(64, 20), dtype=int64, numpy=
array([[  19,  333,   23, ...,    0,    0,    0],
       [  24,   96,    3, ...,    0,    0,    0],
       [   9,    8,    7, ...,    0,    0,    0],
       ...,
       [ 291,  672,    0, ...,    0,    0,    0],
       [  10, 5452,   20, ..., 1980,    0,    0],
       [   9,    8,  853, ...,    0,    0,    0]])>, 'decoder_inputs': <tf.Tensor: shape=(64, 20)
array([[   2,   62,   11, ...,    0,    0,    0],
       [   2,   54, 1175, ...,    0,    0,    0],
       [   2,   20,   12, ...,    0,    0,    0],
       ...,
       [   2,   76,   61, ...,    0,    0,    0],
       [   2,   52, 8015, ...,   10, 1342,    4],
       [   2,   23,  667, ...,    0,    0,    0]])>}, <tf.Tensor: shape=(64, 20)
array([[  62,   11,   22, ...,    0,    0,    0],
       [  54, 1175,   34, ...,    0,    0,    0],
       [  20,   12,   13, ...,    0,    0,    0],
       ...,
       [  76,   61,  859, ...,    0,    0,    0],
       [  52, 8015,   28, ..., 1342,    4,  265],
       [  23,  667,   18, ...,    0,    0,    0]])>)
```

Above: An example of how encoder and decoder inputs appear

2.4 Masking

Masking is an essential piece of architecture in transformer-based natural language translation. When we create our respective embeddings, a large issue is being able to effectively traverse through these embeddings in an overall efficient manner. When we create word position-based embeddings, this can inadvertently create large arrays of zeros which clog up the model since the deep learning model has to train large arrays of zeros and waste a lot of processing power. In order to ameliorate this issue, we introduce the concept of masking in order to avoid this issue altogether.

Masking works essentially as a boolean in data whereas if an array is greater than 0 it's marked as "true" and "false" otherwise. By implementing this feature, we're able to process arrays much faster by essentially which parts of the encoded arrays actually require any processing. By enabling masking, the time and processing requirement can be significantly reduced in order to run the model on less powerful equipment in less time. Therefore, utilization of a masking layer when constructing a model that uses extremely large vectors is always going to serve as a recommendation of which to be cognizant of.

2.5 How Feature Stripping Works

Another feature which is present in Spanish is the use of accented characters such as 'é' and upside down exclamation points ('¡') and question marks ('¿'). These features are not present in English and due to this, for the sake of simplicity, I stripped these features from the model in order to reduce training time and increase the accuracy of the model. However, the embeddings still keep track of the accents so they are maintained when the final translation is output to the final output. However, in an actual model it would be better to keep these features but create specialized tokens specifically to cater primarily to them.

English	Spanish
We're investigating the murder of Tom Jackson.	[start] Estamos investigando el asesinato de Tom Jackson. [end]
Who knows what we'll find up in the attic?	[start] ¿Quién sabe qué encontraremos en el ático? [end]
I'm staying with Tom.	[start] Me quedo con Tom. [end]
I want to eat a mango.	[start] Quiero comer un mango. [end]
I'll show you around the town.	[start] Te enseñaré la ciudad. [end]

Table 1: Showing how the Spanish sentences look after adding '[start]' and '[end]' tokens.

2.6 How Tokens work inside Transformer Architecture

It's necessary to have a place to tell the model to stop encoding. To do this, we create two very distinct tokens when reading the Spanish text in order to prevent the model from further decoding. These tokens are the distinct "[start]" and "[end]" tokens that are added to the beginning and end strings of the Spanish sentences in the dataset to tell the model when to begin and stop the translation process. Otherwise, the model will not be able to decode the process and will enter an infinite loop and will spam "end" repeatedly over and over.

Vectorization is an extremely important factor to consider when creating a model. This is because each word has to have a large array in order to store different categories related to each word. For example, gender plays a massive role in the Spanish language due to many of the nouns being gendered with the letters "o" and "a." As an example, in Spanish "novio" refers to 'boyfriend' and "novia" refers to 'girlfriend' and they are differentiated only using the letter at the end. Therefore, when we conduct the overall task of vectorization, all kinds of related information such as that needs to be stored within an array for each individual word [Cho22].

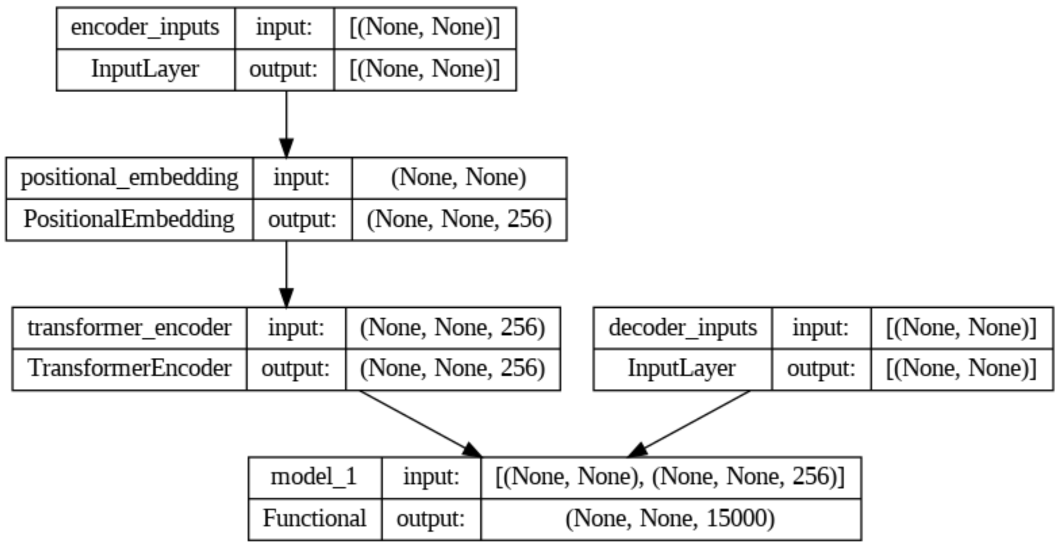

Figure 1: Showing how the Spanish sentences look after adding '[start]' and '[end]' tokens.

2.7 Overall Training Time

When training the model, I noticed that the model was exceptionally difficult to train. My first model had around a million parameters but was unable to properly capture the relationship between the two languages. My seconds model had around 30 million total parameters but after some adjusting with parameters the parameters were a little under 15 million. Originally I attempted to train this model with the CPU on Google Colab but this took around an hour per epoch and crashed after epoch five. Even after switching to GPU to more effectively run the model, this still took around 2 minutes per epoch for 30 epochs to reach convergence for a total of around an hour of overall training time. I purposefully chose 30 epochs since this was how long it took to reach convergence as the accuracy of the validation set settled a little above 70% [Cho23].

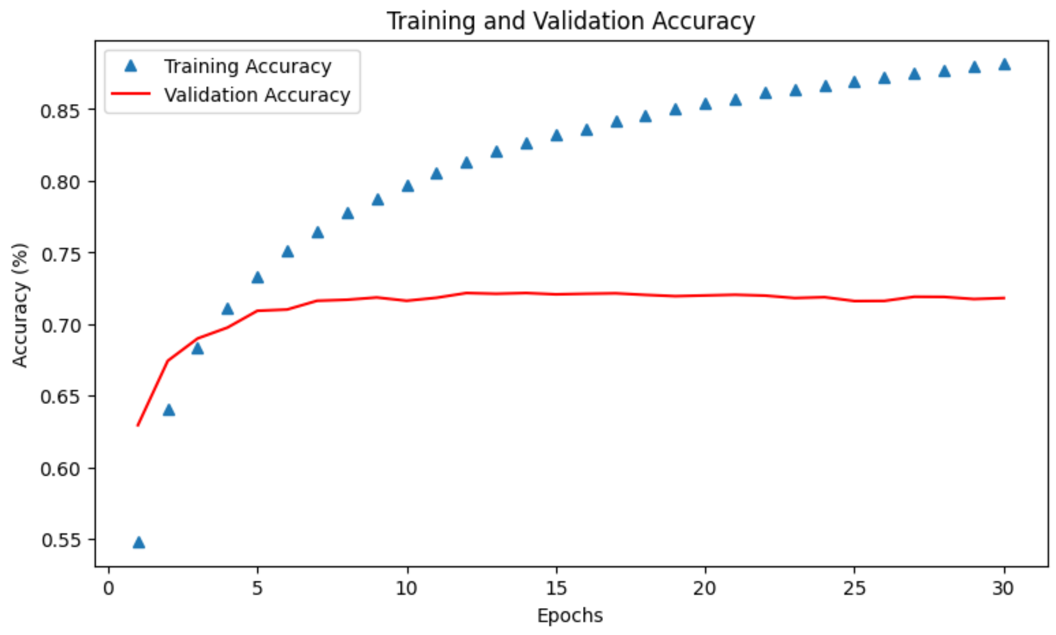

Figure 2: Shows model's accuracy score for both validation and training sets.

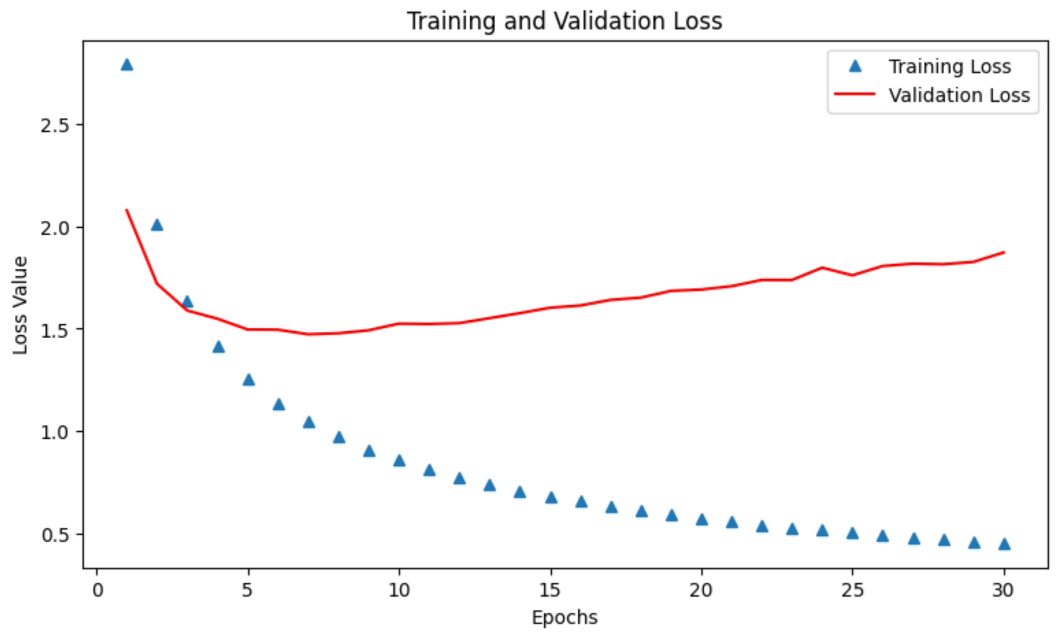

Figure 3: Model's loss score after running 30 epochs; graphs displays for both validation and training sets.

```
Model: "transformer"
__________________________________________________________________________________________________
 Layer (type)                Output Shape                 Param #   Connected to
==================================================================================================
 encoder_inputs (InputLayer  [(None, None)]               0         []
 )

 positional_embedding (Posi  (None, None, 256)            3845120   ['encoder_inputs[0][0]']
 tionalEmbedding)

 decoder_inputs (InputLayer  [(None, None)]               0         []
 )

 transformer_encoder (Trans  (None, None, 256)            1184512   ['positional_embedding[0][0]']
 formerEncoder)

 model_1 (Functional)        (None, None, 15000)          9937048   ['decoder_inputs[0][0]',
                                                                     'transformer_encoder[0][0]']

==================================================================================================
Total params: 14966680 (57.09 MB)
Trainable params: 14966680 (57.09 MB)
Non-trainable params: 0 (0.00 Byte)
__________________________________________________________________________________________________
```

Figure 4: A Summary of the Model Parameters

After decoding our outputs, here are the following results we receive for the first ten Spanish-English pairs:

```
English: I try to save 10% of my wages each month.
Spanish: Cada mes intento ahorrar el 10% de mi sueldo.
Translated: intento ahorrar el 10 de mi sueldo
English: We found an anomaly.
Spanish: Encontramos una anomalía.
Translated: encontramos una apruebo
English: This song is catchy.
Spanish: Esta canción es pegadiza.
Translated: esta canción es hecha
English: Tom was watching Mary and John.
Spanish: Tom estaba observando a Mary y a John.
Translated: tom estaba observando a mary y a john
English: She quickly went up the stairs.
Spanish: Ella subió rápido por la escalera.
Translated: ella subió rápido por la escalera
English: I already told Tom everything I know.
Spanish: Ya le dije a Tom todo lo que sé.
Translated: ya le dije a tom todo lo que sé
English: Tom is the kind of guy that argues for fun.
Spanish: Tom es la clase de tipo que discute por diversión.
Translated: tom es el tipo de tipo de buen tipo funcionaban
English: They'll never know we're here.
Spanish: Ellos nunca sabrán que estamos aquí.
Translated: ellos nunca tesis aquí
English: Tom hasn't heard from Mary since she moved to Boston.
Spanish: Tom no ha oído de Mary desde que se mudó a Boston.
Translated: tom no ha oído de mary desde que se mudó a boston  boston
```

```
English: Today, we'll learn three new words.
Spanish: Hoy aprenderemos tres palabras nuevas.
Translated: hoy soplaba sus nuevas palabras
```

As can be seen from the above model output text, the capitalization and punctuation has been removed from the translated versions. In the future I would like to add is extra code to fix the punctuation in the output or by possibly creating unique and separate vectors for punctuation and capitalization instead of stripping them from the model altogether.

Regarding the future plans of my model, I would very much love to integrate the deep learning model onto my website yeetron.io. I think it would be very engaging to have a section which utilizes AI technology in order to create translations. I saved the weights from when I only ran the epochs for a single cycle and some of the translations were very comedic in nature due to being so poor. I think it would be very entertaining to have a website live for people to see purposefully bad natural language translations.

3 Conclusion

If I have more time in the future, I'd like to possibly integrate voice recognition for translating and maybe have different modes for the 'comedic' and 'accurate' versions. I think maybe to accomplish this I would have to transfer my website off of GitHub onto a server which is able to run Flask so I can properly host my server since that was a huge issue I ran into when trying to integrate my Python code and model into my previously existing HTML/Javascript code. I think to make it work properly I may have to rewrite some of my Python code into Javascript and also convert my Python model into a JSON file instead.

Overall, it was very exciting to build a language translation model and it was very interesting seeing what work went into it. I was unaware of how much pre-processing and encoding was required to create a model. Previously I thought it was similar to creating a "Spam or Ham" model where a model is able to determine which messages are spam and which aren't using a Bayesian model. However, for deep learning in Keras it seems it's more efficient to just build the transformer architecture around an existing encoding and decoding architecture so the deep learning model is able to learn word order on its own.

Instead, for this project a large part of it was finding ways to convert words into vectors and then utilize bi-directional layers to properly keep track of word order within a sentence. We use this in combination with masking, and word embeddings in order to efficiently maintain proper word order between the source and target languages for model training. I think this unique structure was what made this project a very exciting project to construct and build.

References

[Cho22] Fraçois Chollet. Deep learning with python, second edition. pages 370–380, 2022.

[Cho23] Fraçois Chollet. English-to-spanish translation with a sequence-to-sequence transformer. 2023.

[Pes18] Ilya Pestov. A history of machine translation from the cold war to deep learning. 2018.

[Qin21] Lonnie Qin. English-spanish translation dataset. 2021.