Nahuatl-Spanish Neural Translation

Luis Garcia

Abstract

This paper will summarize my findings on the implementation of a Neural Machine Translator for the Spanish and Nahuatl languages. Nahuatl is a low resource indigenous language which posed challenges in gathering sufficient data for training a Machine Learning model to proficiency. This project tested whether word embeddings could help improve an NMT model for such this low resource application. The project attempted to match or exceed a baseline BLEU score of 3 points fund in literature about the topic. However, due to several factors, this baseline score could not be replicated.

1 Credits

This document has been written by Luis Garcia, 2020. The models used for this project were based on Pytorch's seq2seq tutorial, as well as an adapted version of it created by Dr. Mandy Korpusik. The New Testament texts utilized as this projects corpus is the work of Christos Christodoulopoulos.

2 Introduction

Nahuatl is a native languge Indigenous to Mexico and the cultural successor to the language of the Aztecs. It is spoken by millions of people in Mexico despite being endangered, but its use is slowly declining. The advent of the internet is contributing to the decline of endangered languages because the lack of content in these languages makes languges with higher use more advantageous.

This project is born of a wish to aid in the conservation of the Nahuatl language and its indigenous culture by facilitating access of online content to speakers of Nahuatl, and in doing so opening the doors for other indigenous languages.

For this purpose this project focuses on Neural Machine Translation, a form of Natural Language Processing that would generate translation without need for extensive research on languages that may not have online resources explaining grammar and sytanx, or could be lacking a formal definition of such things altogether.

This project has used a seq2seq model in an attempt to match and potentially improve the BLEU scores for this task compared to the scores listed by (Manger, 2018). My approach is based on utilization of word embeddings, following the findings of (Qi, 2018) and originally also the findings of (Zhu, 2020) for contextual embeddings with BERT.

3 Corpus

Consistent with the findings of (Manger, 2018), several difficulties arose within the course of the project; the first and perhaps greatest was persent from the start and it was finding an adequately sized corpus for the machine learning process.

I ultimately parsed the contents of the bible corpus(https://github.com/christos-c/bible-corpus) created by Christos Christodoulopoulos into a usable csv file containing corresponding Nahuatl and Spanish verses in rows. Since the Nahuatl bible only contains the New Testament I made sure to parse only the verses that were contained in both Bible files.

4 Model

The models used in this project have been built upon Dr. Korpusiks Neural Machine Translation model (https://colab.research.google.com/drive/1-LBNA8o-U1oOlXGadtuRzIeFQO1j4oa6?usp=sharing#scrollTo=VOV7PnQwKI8z) following this pytorch tutorial for Seq2Seq translation (https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html).

The neural networks used for this project are part of a Sequence to Sequence RNN model that encodes tensors representing sentences in an input language and decodes these tensors into sentences in the output language. Following the Pytorch tutorial, this project contains a decoder with an attention mechanism and a decoder without one. The structures of the encoder, decoder, and attention decoders are shown in figures 1,2 and 3

respectively, and come from the Pytorch Seq2Seq tutorial site.

For this project I have consistently used Spanish as the input language (because I am a fluent speaker of Spanish). Moreover, I have adapted several parts of the base file to fit my choice of language and corpus. The changes made to the base program have been listed in the following section.

4.1 Changes to code base

The following items consolidate the list of changes I have applied to the code base that was provided by Dr. Korpusik for this project.

4.1.1 split_corpus: I made this method to split entries on the csv file I created. It uses 60% of it for training, 20% for validation and 20% for testing. Following a stackexchange post that I reference in my code, this method splits a pandas dataframe at the 60% mark and then again at the 80% mark ending up with a total of three pieces of the original.

4.1.2 readLangs: I changed this function which originally processed the data downloaded from the corpus provided in the Pytorch Documentation into pairs of french and english sentence translations, and input and outpul language objects. My most important change is that now I feed it a corpus (since I split one into training, validation, and test). I then make a dataframe with the corpus and use the dataframe to generate translation pairs as in the original function, however, I normalized strings after pairing them unlike the original function.

4.1.3 filterPair: I removed the usage of the eng_prefixes tuple that was part of the original translator since I will not be using english words.

4.1.4 prepareData: I added an additional parameter so that a corpus can be passed to readLangs.

4.1.5 train: I added a boolean parameter called uses_attn to specify the use of the provided attention decoder over the non attention decoder. This allowed me to train both decoders without having to change the code every time.

4.1.6 trainIters:I added a boolean parameter called attn to specify the use of the provided attention decoder over the non attention decoder. Like with the train method, this allowed me to train both decoders without having to change the code every time.

4.1.7 evaluateRandomly: I added a parameter to specify a list of sentence pairs to evaluate from to account for training, validation, and test sets.

4.1.8 evaluate_bleu: I added a parameter to specify a list of sentence pairs to evaluate from to account for training, validation, and test sets. I also applied smoothing method number 3 from the nltk SmoothingFunction methods to my evaluation. I did this because running the program as it was produced various warnings and depending on the sentences evaluated, errors.

4.1.9 save_translator: I created this simple function to quickly save the trained models that I produced using the torch.save method. I have some trouble running the program on my laptop (it produces results but crashes my gpu) and it helps to have pretrained encoders and decoders available.

4.1.10 load_translator: This is another short method that will load saved encoder and decoder models.

4.1.11 evaluate_saved: This method will quickly evaluate the BLEU scores of pairs of saved encoders and decoders.

4.2 Word Embeddings

The encoder model as well as the decoder models have been updated to include pretrained fasttext embeddings. This was done by extracting the matrix of fasttext vectors initializing them as the weights of these models' embedding layers.

4.3 Contextual Embeddings

The plan for this project was to include contextual embeddings using BERT following the methodology of Zhu, Jinhua et al. However due to technical difficulties and time constraints, this addition has not yet been implemented and will remain as a future extension of this project. With the addition of these embeddings the model is expected to obtain higher BLEU Scores.

5 Results

Model	**BLEU score**	**source**
Baseline NMT	3.04	**Manger**
Baseline SMT	10.104	**Manger**
Base Model (French to English)	85.4	**Korpusik**
Milestone NMT	0.0348	**Attn, no emb**
Current NMT	0.0334	**Attn & emb**
Milestone NMT	0.0224	**No attn, no emb**
Current NMT	0.0351	**No attn, emb**

Table 1: Bleu Results

My translator has failed to match the proposed baseline BLEU scores by quite a bit, and this applies not only to the basic encoder decoder RNN pairs but also to the attention pairs and their corresponding pairs with pretrained word embeddings. I have failed to obtain a single point on my BLEU score, but I have some hypotheses about why the results have been so underwhelming: corpus size, syntactic differences

between input and output languages, and the nature of the written style of the corpus.

As previously mentioned I have used a bible corpus comprised of around 7,000 New Testament verses for a Nahuatl bible. (Zhu et al., 2020) list their low-resource translation corpora as each having over 100,000 samples. (Manger, 2018) recognizes corpus size as one of the many challenges that plague NLP technologies for Indigenous Laguages, and it seems the bible corpus chosen for this project fits in this mold.

The parallel translations of this corpus also have varying verse lengths across languages. This suggests that the Nahuatl translation of the New Testament may be translating concepts or words that are not native to Nahuatl into phrases. This may explain also why translations created by my model tend to extend the word count of the Spanish input sentence. Additionally, some words such as “para”, Spanish for “for”, are used as is in the Nahuatl bible verses. (Manger, 2018) identifies this issue by the linguistic term context switching. This is a tendency for bilingual speakers to inject words, phrases or sentences of another known language into the one they are speaking. I failed to account for the occurrence of context switching in my model.

In addition, the writing style of the bible corpus matches older patterns of writing that use pompous, longer words. This style is common for Biblical texts, however, considering that the fastText word embeddings I incorporated to the model have been pretrained on text extracted from Facebook, I doubt that the word embeddings had much compatibility with the corpus I used for training, validation, and testing. Perhaps this could be considered a form of overtraining. As I do not know Nahuatl, I have no evidence for this, but I hypothesize this could explain the effect of adding embeddings to the model.

5.1 Conclusion

Ultimately low BLEU scores were not unexpected based on the NLP challenges of Indigenous languages (Manger, 2018). This project charged ahead with the task almost blindly due to unavailable data resources or reference of models for the Nahuatl language.

Moving forward it may be important to increase the size of the corpus, or to experiment further with data augmentation strategies. Moreover, It may be necessary to further preprocess data to account for context switching.

In addition, as contextual embeddings have not yet been incorportated into this project, so these may have a positive impact on the performance of the task.

References

Mager, Manuel & Gutierrez-vasques, Ximena & Sierra, Gerardo & Meza, Ivan. (2018). Challenges of language technologies for the indigenous languages of the Americas.

Mager, Manuel & Meza, Ivan. (2018). Hacia la Traducción Automática de las Lenguas Indígenas de México.

Qi, Ye & Sachan, Devendra & Felix, Matthieu & Padmanabhan, Sarguna & Neubig, Graham. (2018). When and Why Are Pre-Trained Word Embeddings Useful for Neural Machine Translation?. 529-535. 10.18653/v1/N18-2084.

Zhu, Jinhua & Xia, Yingce & Wu, Lijun & He, Di & Qin, Tao & Zhou, Wengang & Li, Houqiang & Liu, Tie-Yan. (2020). Incorporating BERT into Neural Machine Translation.

Supplemental Material

Mager, Manuel & Mager, Elisabeth & Medina-Urrea, Alfonso & Meza, Ivan & Kann, Katharina. (2018). Lost in Translation: Analysis of Information Loss During Machine Translation Between Polysynthetic and Fusional Languages.

Wang, Xinyi & Neubig, Graham. (2019). Target Conditioned Sampling: Optimizing Data Selection for Multilingual Neural Machine Translation.

Zhang, Jinyi & Matsumoto, Tadahiro. (2019). Corpus Augmentation for Neural Machine Translation with Chinese-Japanese Parallel Corpora. Applied Sciences. 9. 2036. 10.3390/app9102036.