**What did you accomplish?**

* Created a Machine Translation for Tibetan to English, a low resource language.
* Performed the full pipeline of Machine Translation from Data collection, cleaning, tokenization, training, and testing.
* Three simultaneous approaches:
  + Used Fairseq, existing tools to train our data with transformer architecture
  + Fine-tuned pre-trained T5 transformer to train our data
  + Build a transformer from scratch

**What did you *not* accomplish that you hoped to accomplish?**

* We did not achieve state-or-art performance as good as high resource languages
  + Why?
    - The size and quality of the data set is very poor
    - The complexity of the Tibetan language made it difficult for proper tokenization - a single alphabet can be formed by using two to three Unicode characters.
    - For a language that is very different from English, the hyperparameter is harder because there hasn’t been any research done on Tibetan prior to this, that we know of.

**What further work could you do on the project—if, for example, you had another six months to work on it? (Those aren’t necessarily things that were in your original plan, but just further ideas for how somebody might make it better.)**

* Data collection
* Working on a baseline model using RNN
* Continue working on improving the hyperparameter for the transformer, and learning the theory behind the transformer.

**What were the challenges?**

* Low resource language therefore data collection was especially difficult
* Accessing good computing power

(Lambus: the usage of HPC was more difficult than expected – how to make sure the code will run correctly on the cluster; how to manage transmission of big file; how to log and save relevant information for later usage. In addition, the python libraries may have been built inappropriately and we constantly bumped into segmentation fault when using Pytorch and huggingface transformer. It was a struggle to figure out the cause of the issue and contact the HPC team at Middlebury)

* Prioritize the best next step was difficult because everything seemed equally important, and they all take a lot of time to experiment with.

(Lambus: There are always too many ideas to experiment with but too little time to run the code and too little certainty to know which approach will work. For example, we could have chosen to experiment with different learning rate schedules, changing the architecture of transformer, build other baseline models such as RNN, writing better decoding algorithms such as beam search and k-nearest neighbor, etc. However, each experiment takes at least double-digit hours to run on GPU, not considering possible breakdown in the middle. Also, we do not know exactly what approach will yield reasonable result)

* The level of understanding of deep learning, and transformer.

 (Lambus: This project inspired us to learn more about deep learning in grad school in a systematic way. For this senior project, we identified valuable real life problem but soon realized that our knowledge is only enough to produce a prototypical result)

**What did you learn?**

* The pipeline for Machine Translation
* How to find the appropriate tool for a specific task
  + How to incorporate already existing tools for a low resource language like Tibetan, with no prior research work.

(Lambus: We have to experiment tools at different levels: transformer architecture provided by FairSeq, transformer provided by huggingface, write a transformer from scratch… then see which one works)

* + How to use tools such as PyTorch, and Hugging face transformer on Tibetan.
  + Deciding between creating a language-based tokenizer or using other pre-existing tokenizers.
* Reading papers to learn about the mechanism, and experiments - T5, and sentencePiece
* Create an efficient workflow for Machine Translation example caching necessary data

(Lambus: caching data, log necessary information, robustness of code as it must run without error on a server, unexpected problems showing up…)

* Patience, persistence, and thinking from different angles.

Result and Conclusion