Machine Translation in Argos Translate (2020)

Introduction

In this video we will be discussing machine translation in Argos Translate, an open source Python library and Desktop application for doing neural machine translation.

Sequence to sequence model

The heart of a modern machine translation system is a neural network that does sequence to sequence modeling, that is converting a sequence of tokens in the source language into tokens in the target language.

Human language is very nuanced and it would be extremely difficult for a programmer to write a set of rules to fully capture its complexity. Neural networks trained on a large amount of data are able to capture this nuance much better using a process that is roughly analogous to what neurons in our brains do.

Argos Translate uses Transformer models, a neural network architecture with an attention mechanism that can be trained in parallel more efficiently than past approaches. Models are trained using OpenNMT, an open source neural machine translation system.

Inference

To run the models Argos Translate uses CTranslate2 an optimized inference engine that supports both CPU and GPU execution. Fast CPU execution without the need for specialized hardware is important because it allows for simplified server deployment and local translations for enhanced privacy.

Argos Translate also supports pivoting through an intermediate language to perform translations that aren't directly installed. For example, if Portuguese to English and English to Korean models were installed you could translate from Portuguese to Korean by passing through English.

Argos Translate has a desktop application based on PyQt and the LibreTranslate project has a web application.

Data

The primary source of data for Argos Translate is the Opus parallel corpus which has compiled a number of open data sources in a standard format for easy access. Wiktionary definition data is also used.

Data processing

The training scripts for Argos Translate load and filter the data for training. Emojis are removed from the training data to preserve emoji integrity in translations by preventing the sequence to sequence model from modifying them. There is also support for adding “special tokens” to the data to add functionality to the model. For example, the <define> token can be used with Wiktionary data to improve translation quality in languages without a large quantity of data available and to improve single word translations.

Finally, the training scripts package trained models in a standard format for use.

Tokenization

The sequence to sequence model operates on a series of discreet tokens. In theory you could make each character its own token. However, Argos Translate uses the SentencePiece tokenizer to split input text into sub-word tokens to take some of the burden of spelling words off of the sequence to sequence model. By splitting into sub-words instead of full words the sequence to sequence model is still able to understand uncommon compound words.

Sentence boundary detection

Currently the sequence to sequence model is only able to translate individual sentences meaning input text needs to be split into sentences. There are techniques for doing this effectively using periods, however, these techniques wouldn’t work for languages without periods. Instead Argos Translate uses Stanza which uses neural networks to do sentence boundary detection in a large number of languages.

Package management

Argos Translate has a package manager and package index with pretrained models. Trained models are packaged with metadata and the files needed by Stanza and SentencePiece for easy distribution and installation. Packages can either be installed manually from a file or automatically downloaded from the package index using the Python library or desktop GUI.

Peer to peer

Packages have peer to peer links for downloading models using either IPFS or BitTorrent. Peer to peer distribution of models reduces the cost of distribution and is more resilient without a single point of failure.

Web

LibreTranslate is an API and web app built on top of Argos Translate using Python and the Flask framework. LibreTranslate can be run from a Docker container and has language bindings for Rust, C#, and NodeJS. It also supports detecting languages, and text in scripts not present in the dataset such as Russian written using the Latin alphabet.

Conclusion

Together these projects provide a powerful, extensible, open source translation stack with many potential applications. What are you going to build with it?