Neural Machine Translation

Pradeep Singh

Computational Science Research Center,
San Diego State University

June 2018

Abstract

In this project, I have worked on an End-to-End Machine Translation pipeline that accepts English text as input and returns the French translation.

Introduction

Machine translation is a popular topic in research with new papers coming out every year. Over the years of research, different methods were created, like rule-based, statistical, and example-based machine translation. With all this effort, it's still an unsolved problem. However, neural networks have made a large leap forward in machine translation.

Dataset

The Europarl is a standard dataset used for statistical machine translation, and more recently, neural machine translation. It is comprised of the proceedings of the European Parliament, hence the name of the dataset as the contraction Europarl. I have worked on a dataset which is consist of French and English translations. It consists of 2,218,201 English sentences and 2,190,579 French sentences. I have used a subset of this dataset, as it will take a long time to train neural network on. But, I do plan to work on full dataset in near future.

Data Pre-processing

For a neural network to predict on text data, it first has to be turned into data it can understand. Text data like "dog" is a sequence of ASCII character encodings. Since a neural network is a series of multiplication and addition operations, the input data needs to be number(s).

I have used following pre-processing techniques:

1. Tokenization: We can turn each character into a number or each word into a number. These are called character and word ids, respectively. Character ids are used for character level models that generate text predictions

for each character. A word level model uses word ids that generate text predictions for each word. Word level models tend to learn better, since they are lower in complexity, so we'll use those.

2. Padding: When batching the sequence of word ids together, each sequence needs to be the same length. Since sentences are dynamic in length, we can add padding to the end of the sequences to make them the same length.

Models

Basically, an NMT model first reads the source sentence using an encoder to build a "context" vector, a sequence of numbers that represents the sentence meaning; a decoder, then, processes the context vector to emit a translation. This is often referred to as the encoder-decoder architecture and state of the art results in machine translation use similar architecture.

NMT models can vary in terms of their exact architectures. A natural choice for sequential data is the recurrent neural network (RNN), used by most NMT models. Usually an RNN is used for both the encoder and decoder. The RNN models, however, differ in terms of: (a) directionality – unidirectional or bidirectional; (b) depth – single- or multi-layer; and (c) type – often either a vanilla RNN, a Long Short-term Memory (LSTM), or a gated recurrent unit (GRU).

Current state of the art models in machine translation systems is based on Sequence-to-sequence (seq2seq) models with attention mechanism. But for better understanding (and for fun) I have experimented with various neural network architectures (without any attention mechanism) like:

1. Model 1: Simple RNN
2. Model 2: RNN with Embedding
3. Model 3: Bidirectional RNN
4. Model 4: Encoder-Decoder RNN
5. Model 5: Bidirectional RNN with Embedding

Prediction

While you're training your models (and once you have trained models), you can obtain translations given previously unseen source sentences. This process is called inference/ prediction.

At prediction time, we only have access to the source sentence. There are many ways to perform decoding. Decoding methods include greedy, sampling, and beam-search decoding. I have used the greedy decoding strategy.

The idea is simple:

1. Step 1: We encode the source sentence in the same way as during training to obtain an the last hidden state of the encoder, and this hidden state is used to initialize the decoder.

2. Step 2: The decoding (translation) process is started as soon as the decoder receives a starting symbol.

3. Step 3: For each timestep on the decoder side, we treat the RNN's output as a set of logits. We choose the most likely word, the id associated with the maximum logit value, as the emitted word (this is the "greedy" behavior We then feed this word as input to the next timestep.

4. Step 4: The process continues until the end-of-sentence marker is produced as an output symbol.

Step 3 is what makes prediction different from training. Instead of always feeding the correct target words as an input, inference uses words predicted by the model.

Result

Among all the models that I have trained, model with Bidirectional RNN with Embedding worked best with accuracy of around 85%.

Future Work

1. Train much more deep network.

2. Train on bigger dataset, such as WMT.

3. Work with Attention based mechanism.

Miscellaneous

For deeper reading on Neural Machine Translation and sequence-to-sequence models, we highly recommend the following materials by Luong, Cho, Manning, (2016); Luong, (2016); and Neubig, (2017). You can find code for this project on GitHub. For any feedback you can contact me at pradeepsingh7890@live.com.