Neural Machine Translation

Michael Bartoli

December 10, 2014

Abstract

Neural machine translation is an emerging approach to machine translation that seeks to construct and train a deep recurrent neural network that outputs a correct translation for a given sentence. We successfully implement the model presented in (Bahdanau et al. [2014]) using the GroundHog package developed by members of the LISA lab at the Université de Montréal. After our model finishes training on the English-French parallel EuroParl corpus, we expect a BLEU score (approx. 36.5) comparable to state-of-the-art machine translation systems as achieved by (Bahdanau et al. [2014]).

1 Introduction

In the last five years, the machine learning community has seen a resurgence of interest in neural networks, specifically deep neural networks, due in part to the reduced cost of computing power and the emergence of general-purpose computing on graphics processing units as an essential piece of the high-performance computing toolchain. As a result, the implementation of, once only theoretical, deep neural nets has resulted in record breaking results in computer vision tasks (Bengio [2009]). The next reasonable extensions would be domains with foundations built on machine learning, like natural language processing.

Traditional statistical machine translation models encode and decode sentences associated parallel corpus to create a system that outputs a correct translation for a given sentence (Jurafsky and Martin [2008]). The first proposed machine translation models relied on the training separate neural networks for the encoding and decoding process. In the summer of 2014, this framework was extending to using two deep recurrent neural networks to individually handle the encoding and decoding processes (Cho et al. [2014]). This Fall, a new model that combines these two processes to create a single, deep recurrent neural net to handle the entire encoding-decoding translation model (Bahdanau et al. [2014]).

These neural networks work by extracting linear combinations of derived features as hidden layers and model the target as a nonlinear function of these features (Hastie et al. [2001]), the most simple of which is a single hidden layer, feed-forward neural network (see Figure 1). Deep neural networks extend this model to thousands of hidden layers (Bengio [2009]).

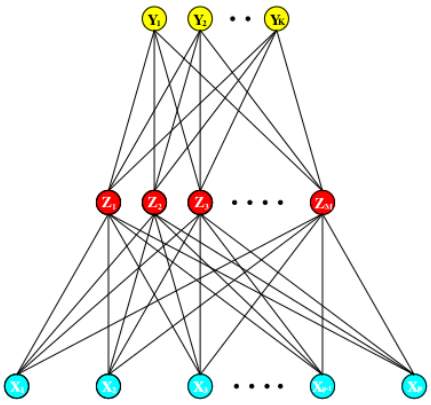

Figure 1: Schematic of a single hidden layer, feed-forward neural network (Hastie et al. [2001])

2 Methods

Our implementation of the deep recurrent neural network machine translation model developed by (Bahdanau et al. [2014]) was developed using the GroundHog package developed by members of the LISA lab at the Université de Montréal based on Theano, software that efficiently defines, optimizes, and evaluates mathematical expressions involving multi-dimensional arrays (Bergstra et al. [2010]). The initial parameters for tuning the deep recurrent neural network were additionally provided by (Bahdanau et al. [2014] and configured to work with GroundHog and Theano. We further tuned the initial parameters by decreasing the word embedding dimensionality from 620 to 420 and increasing the size of a hidden layer n from 1000 to 1500 (Pascanu et al. [2013a]). The aligned training and testing data was from the EuroParl English-French corpus. It was preprocessed using tokenization scripts from Moses (Koehn et al. [2007]), and subsequently optimized using Pickle and H5FS. We trained our model on a workstation with 132 gigs of memory and a 12-core AMD Opteron 6168 processor. Configuring this model to specified workstation took a nontrivial amount of time. However, this setup is a joke for training deep neural networks; it doesn't even have a GPU!

3 Results

The deep recurrent neural networks discussed in this paper relies, mathematically, on stochastic gradient descent for parameter optimization. This, however, means that the training process takes a long time, especially when running on a single multi-core CPU (Bengio et al. [1994]). Training deep recurrent neural networks is therefore computationally nontrivial, due to SGD and other NP problems (Pascanu et al. [2013b]). As a result, as of December 11th, our English-French translation model is still being trained. However, we project to achieve similar BLEU scores as (Bahdanau et al. [2014]) (see Figure 2).

Model	All	No UNK°
RNNenc-30	13.93	24.19
RNNsearch-30	21.50	31.44
RNNenc-50	17.82	26.71
RNNsearch-50	26.75	34.16
RNNsearch-50⋆	28.45	36.15
Moses	33.30	35.63

Figure 2: Projected BLEU scores of trained models computed on the test set (Bahdanau et al. [2014])

Additionally, we setup and ran an SMT model on Moses to get a BLEU score of about 35.5. At no point were we able to find a baseline SMT model that reached the level that Bahdanau et al. reached with RNN-search, and we anticipate that our RNN model will perform comparably well when trained.

4 Discussion

Training deep recurrent neural networks for machine translation is still in its infancy. Although we successfully implemented a recently proposed model, we ran into difficulty with the computational complexity on our albeit workstation. We would expect faster results if we ran Theano on a CUDA environment, given the speed ups of the use of GPUs in high-performance computing. Additionally, since a large portion of the computational complexity in training the deep recurrent neural network is a result of stochastic gradient descent (Bengio et al. [1994]), which boils down to floating-point arithmetic, a tier 1 GPU or a cluster of GPUs would be an ideal choice as GPUs are developed and optimized for similar mathematical calculations.

Despite the drawbacks of the computational complexity, neural machine translation is already pushing the boundaries of state-of-the-art machine translation. We expect this trend to continue due to macro trends in machine learning towards similar deep architectures. A simple extension of the work done for this paper would be to explore translation models for the other 22 languages in the EuroParl corpus. Building neural machine translation models for other languages would simply be a waiting game.

Recently, in December 2014, the same architecture was applied to speech recognition (Chorowski et al. [2014]). It's hard not to see deep learning as a framework on the eve of transforming natural language processing even further. For example, the next step might be to combine this implementation with (Chorowski et al. [2014]) to create a near real-time translation system for voice and video calls. With enough time (and money) we'd be interested in exploring this potential connection further.
Further, a few days ago at NIPS, we saw even greater improvements expanding

on this model (Sutskever et al. [2014]).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL `http://arxiv.org/abs/1409.0473`.

Yoshua Bengio. Learning deep architectures for AI. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009. ISSN 1935-8237. doi: 10.1561/2200000006. URL `http://dx.doi.org/10.1561/2200000006`.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. URL `http://www.iro.umontreal.ca/~lisa/pointeurs/ieeetrnn94.pdf`.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL `http://arxiv.org/abs/1406.1078`.

Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. End-to-end Continuous Speech Recognition using Attention-based Recurrent NN: First Results. *ArXiv e-prints*, December 2014.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition) (Prentice Hall Series in Artificial Intelligence)*. Prentice Hall, 2 edition, 2008. ISBN 0131873210.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. Moses: Open source toolkit for statistical machine translation. In *Annual Meeting of the Assoc. for Computational Linguistics*, pages 177–180, Prague, Czech Republic, June 2007.

Razvan Pascanu, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. *CoRR*, abs/1312.6026, 2013a. URL `http://arxiv.org/abs/1312.6026`.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *ICML (3)*, volume 28 of *JMLR Proceedings*, pages 1310–1318. JMLR.org, 2013b. URL `http://dblp.uni-trier.de/db/conf/icml/icml2013.html#PascanuMB13`.

Ilya Sutskever, Oriol Vinyals, and Quoc V. V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014. URL `http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf`.

5 Appendix

5.1 Installation

Our current code can be found at `https://github.com/mbartoli/nmt`. To run the software you'll need to satisfy all dependencies for Theano, Scipy, tables, h5fs, and pickle. Additionally, you will need to install a BLAS compatible with your OS. Additional information on setting up Theano can be found at `http://deeplearning.net/software/theano/install.html`. To download our code to the current directory run

```
git clone https://github.com/mbartoli/nmt.git
```

5.2 Configure

After installation, navigate to the top-level of the GroundHog folder in the cloned repository and, where 'path/to/GroundHog' is the path to your GroundHog folder, execute

```
sudo python setup.py develop
export PYTHONPATH=/path/to/GroundHog
export GHOG=/path/to/GroundHog
cd ./data
unzip source-en-en-binarized_text_h5.zip
unzip source-en-en-binarized_text_pkl.zip
unzip target-en-fr-binarized_text_h5.zip
unzip target-en-fr-binarized_text_pkl.zip
cd ./GroundHog/experiments/nmt/
cat deepAttention_model.npz.zip.gz* | zcat > deepAttention_model.npz.zip
unzip deepAttention_model.npz.zip
```

5.3 Training

To continue training the deep neural network navigate to GroundHog/experiments/nmt/ and execute

```
THEANO_FLAGS=device=cpu,floatX=float32 python train.py
```