Applying Deep Learning Neural Network Machine Translation to Language Services

Special Course

Yannis Flet-Berliac
MSc in Digital Media Engineering
DTU

[Supervisor] Michael Kai Petersen
Associate Professor
DTU

Abstract

Recurrent neural networks (RNNs) have been performing well for learning tasks for several decades now. The most useful benefit they present for this paper is their ability to use contextual information when mapping between input and output sequences.

Nevertheless, standard recurrent neural network architectures challenged with translation tasks can only provide a limited range of context. Context which is of massive importance for complex translations.

A deep neural network for machine translation implies the use of a sequence-to-sequence model, consisting of two RNNs: an encoder that processes the input and a decoder that generates the output. To address this, we will use a recurrent neural network with a Long Short-Term Memory architecture as depicted in the next section. This recent architecture - whose first attempts were made in the 1990s - will be evaluated with highly specialized texts currently challenging language services.

To meaningfully assess the model's performances, texts from a translation company and thoughts from skilled experts about specialized topics will be tested.

1. INTRODUCTION

To understand why a Long Short-Term Memory architecture gives better results for machine translation, we need to evoke the so-called vanishing gradient problem.

This is a problem arising in gradient-based learning methods: those methods learn a parameter's value in the neural network by understanding how a small change in the parameter's value will affect the network's output.

If a change in the parameter's value causes very small change in the network's output, the network can't learn the parameter effectively, which is a problem.

This is exactly what is happening in the vanishing gradient problem - the gradients of the network's output with respect to the parameters in the early layers (the previous words in the input sequence, see later for a graphic illustration) become extremely small ie. even a large change in the value of parameters for the early layers doesn't have a big effect on the output.

This is reasonably avoided by basic RNNs for short sentences or when the distance between the relevant information and the place where the model needs to predict the next word is small, as depicted in figure 1. In that architecture, A is a chunk of the neural network (a hidden layer) looking at some input x_i and outputing the value h_i. The highlighted elements x_0 and x_1 can easily be understood as the words that gather the relevant information in the sentence and the h_3 the word-to-predict in that same sentence. The information flow is depicted by the horizontal arrows that go from left to right.

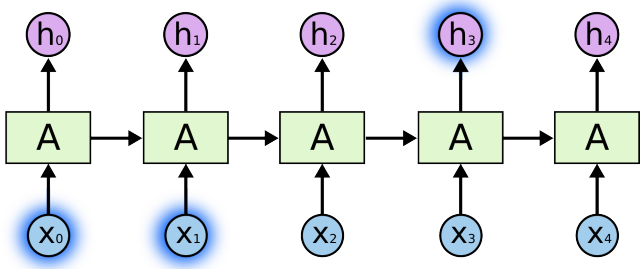

Figure 1: The RNN succeeds in learning to use the past information for small gaps in the sentence

But when this gap enlarges like in figure 2, the accuracy of the basic RNN model can't be proven anymore. The information flow suffers from the vanishing gradient descent and the more it goes through the chunks A, the more reduced its intensity is. Long term dependencies are harder to learn and this is even more reported for texts with complex context that require language services' expertise for a proper translation.

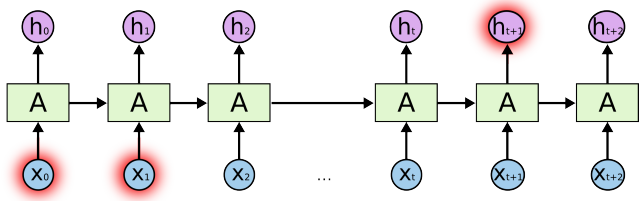

Figure 2: The RNN fails in learning to use the past information for large gaps in the sentence

As shown in figure 3 the sensitivity decays over time as new inputs overwrites the activations of the hidden layer. The RNN "forgets" the first inputs (ie. the impact of previous words is reduced and the context is forgotten).

Hopefully, a Long Short-Term Memory (LSTM) architec-

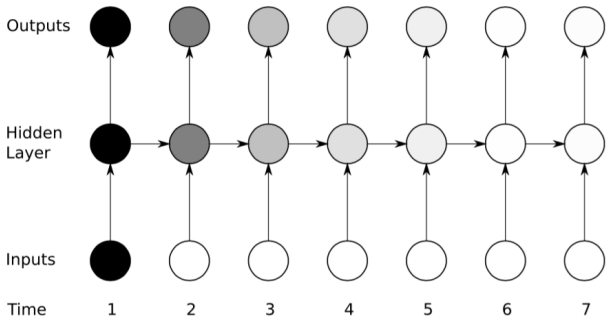

Figure 3: The sensitivity of hidden layers decays over time

ture can avoid that and gives interesting results for sequence-to-sequence learning.

2. THE MODEL

2.1 LSTM

The LSTM model introduces a new structure called a memory cell (see figure 4 below). Compared to a simple RNN cell structure (see figure 5 below), LSTM memory cells are composed of four main elements: an input gate, a neuron with a self-recurrent connection (also called the cell state which is represented as the long horizontal arrow at the top of the cell), a forget gate (which looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each value in the cell state - a 1 will keep all of the input information while a 0 will ignore it) and an output gate. As shown, the gates serve to modulate the interactions between the memory cell itself and its environment.

The input gate can either allow incoming signal to alter the state of the memory cell or completely block it. On the other hand, the output gate can either allow the state of the memory cell to have an effect on other neurons or prevent it. Finally, the forget gate can modulate the memory cell's self-recurrent connection, allowing the cell to remember or forget its previous state, as needed according to cost function during the training of the model.

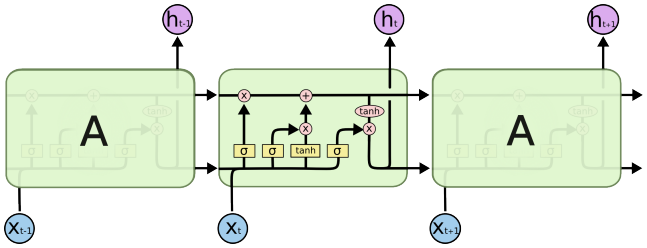

Figure 4: Repeating module of an LSTM memory cell

Thinking in terms of the forward pass, the input gate learns to decide when to let activation pass into the memory cell and the output gate learns when to let activation pass out of the memory cell.

The core of our model consists of an LSTM cell that processes one word at a time. Using all those flows of information it computes probabilities of the possible continuations

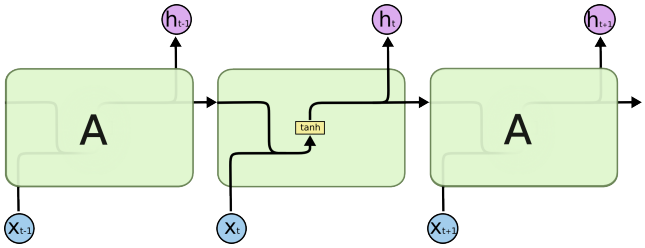

Figure 5: Repeating module of a standard RNN cell

of the sentence given an input of sequence of words (we will address the sequence-to-sequence formulation in the next section). The memory state of the network is at the very beginning initialized with a vector of zeros and gets updated after reading each word.

In our case, for computational reasons we will process data in mini-batches of size 64 (64 sentences per time step).

2.2 Sequence-to-Sequence

For our model we will use a neural network architecture that learns to encode a varying-length sequence into a fixed-length vector representation and to decode a given fixed-length vector* representation back into a variable-length sequence.

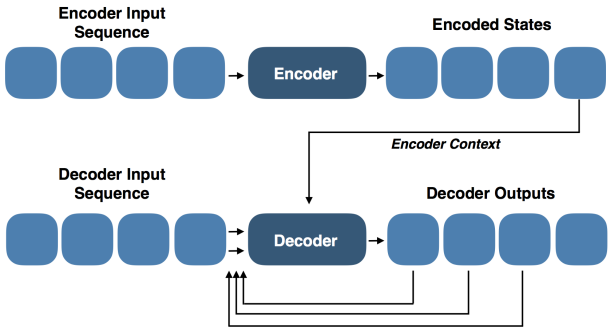

Figure 6: Sequence-to-Sequence RNN Encoder-Decoder model

The encoder is a RNN that reads each word of an input sequence x sequentially. As it reads each word, the hidden state of the RNN changes (the LSTM memory cell state changes, ie. the information flow gets updated recursively). When the RNN reaches the end of the sequence the hidden state of the RNN is a summary of the whole input sequence: the context information has been preserved.

The decoder of the model is yet another RNN which is trained to generate the output sequence by predicting the next word given the hidden state we just talked about.

Until that very moment, the model we described has each input being encoded into a fixed-size state vector*. This may make it difficult for the neural network to cope with long sentences with all the information being compressed into a fixed-size vector. That is the reason why the attention mechanism has been additionally used. Both a recent and interesting architectural innovation in neural networks first introduced by A. Graves and then developed by D. Bahdanau et al., this mechanism encodes the input sentence

into a sequence of vectors and chooses a subset of these vectors adaptively while decoding the translation. The decoder decides which parts of the source sentence it needs to pay attention to. The neural networks can focus on different parts of their input which allows the decoder to appropriately focus on the relevant segments of the information flow spreading along the sequence.

Finally the two components of the RNN Encoder-Decoder are jointly trained to maximize the conditional log-likelihood:

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \mathbf{logp}_{\theta}(\mathbf{y_n} \mid \mathbf{x_n})$$

with θ the model parameters, x_n an input sequence, y_n its associated output sequence and N the total number of pair sentences in the data set (more details raised in the next section). To maximize this log-likelihood, the gradient-based algorithm is eventually called to estimate the model parameters at each step of the training.

3. THE TRAINING

We used TensorFlow's framework and its seq2seq library together with the WMT'14 English to French data set composed of 12 million sentences. In our experiment and for both languages we used a vocabulary size of 40K words with the most frequent ones. This seemed to be a fairly good trade-off between computing time and model's accuracy. Accordingly, every word out of the vocabulary was replaced with a special 'UNK' token. We also used an AWS EC2 instance with GPU Support to have a remote access to a large computing power and additional storage needed for the training. To set up the model, it was needed to install several tools into the instance including TensorFlow 0.6, Essentials, Cuda Toolkit 7.0, cuDNN Toolkit 6.5 and Bazel 0.1.4. An EC2 g2.8xlarge instance using Ubuntu Server 14.04 LTS AMI has been used for the experiment. A public AMI (ami-a5d1adb2) has been created during this project with the resulting machine translation model, as of now.

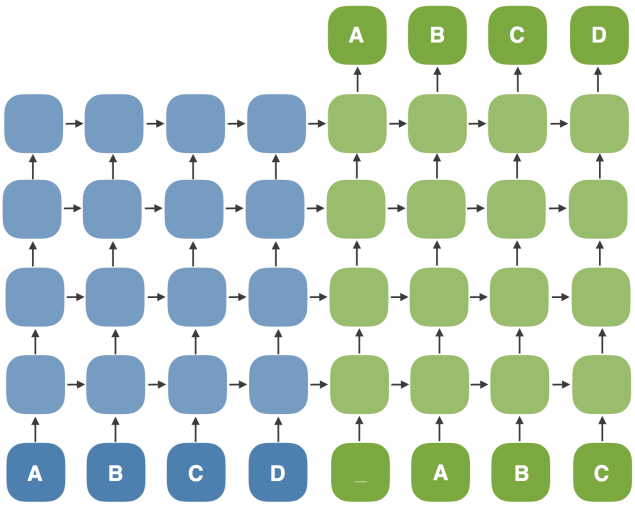

Figure 7: The 4-layers neural network used in the experiment.

40K input words and 40k output words have been used with 4 layers of 512 LSTM cells each with different LSTMs for input and output languages, presenting 1000 dimensional word embeddings. The resulting network looks like the one in figure 7 above. As you can observe, we represented the 512 LSTM cells by only using 4 for the encoder and 4 for the decoder. To save virtual storage space we used batches of size 64 during the training. It took 1.3 seconds for each time-step approximately - given that we trained the model for 160000 steps, the resulting computing time is 58 hours with a g2.8xlarge EC2.

When in your instance with the provided AMI, type the following requests in your shell to go further with the model's training and see how the perplexity evolves:

```
cd tensorflow/tensorflow/models/rnn/translate
python translate.py --data_dir /data --train_dir /
--size=512 --num_layers=4
```

You should see hundreds of lines of outputs that will demonstrate a working GPU Support (as a side note, you don't necessarily need a g2.8xlarge instance to make it work, g2.2xlarge is also fine but as you can imagine, slower).

You can as well type the following code at any time after interrupting the training to test out the machine translation:

```
cd tensorflow/tensorflow/models/rnn/translate
python translate.py --decode --data_dir /data
--train_dir / --size=512 --num_layers=4
```

This will prompt you with a '>' waiting for an English sentence to be inputted. The model will then translate the latter into French.

4. SOME RESULTS

When running the training we access some information (a few output examples below) concerning the global state of the model and specific states concerning each bucket, every 200 time-steps (one bucket contains sentences within a fixed range of length).

```
global step 119600 learning rate 0.2107 step-time 1.39
perplexity 5.55
  eval: bucket 0 perplexity 5.97
  eval: bucket 1 perplexity 4.07
  eval: bucket 2 perplexity 5.78
  eval: bucket 3 perplexity 7.02
global step 119800 learning rate 0.2107 step-time 1.30
perplexity 5.48
  eval: bucket 0 perplexity 6.98
  eval: bucket 1 perplexity 5.09
  eval: bucket 2 perplexity 5.75
  eval: bucket 3 perplexity 6.95
global step 120000 learning rate 0.2107 step-time 1.41
perplexity 5.58
  eval: bucket 0 perplexity 5.01
  eval: bucket 1 perplexity 4.77
  eval: bucket 2 perplexity 5.73
  eval: bucket 3 perplexity 7.14
```

This gives us useful insight into the perplexity of the model, ie. its accuracy. As a reminder, the perplexity measures the cross entropy between the empirical distribution (the actual one) and the predicted distribution. It then divides the cross entropy by the number of words and takes the exponential after throwing out unseen words. It gives a good understanding of our model's performances.

Now that we ran the model for almost 60 hours, we can test it out by calculating the BLEU score (Bilingual Evaluation Understudy) on some sentences. The BLEU score compares the hypothesized translation from the model with a human translation. We will use the implementation of the BLEU score by Papineni et al. Two identical sentences would have a score of 1.

The SOURCE is the source sentence in English, the REFERENCE is the target sentence in French translated by a human, the HYPOTHESIS_MT is the sentence in French generated by our machine translation, and the HYPOTHESIS_GT is the sentence in French generated by Google Translate.

SOURCE	Although it is based on only one main design idea, it excels on account of the interesting and meaningful selection of architectural features.
REFERENCE	Bien qu'il se fonde sur une seule idée principale sur le plan graphique, il se distingue par une sélection intéressante et cohérente de motifs architecturaux.
HYPOTHESIS_MT	Bien que ce soit soit une seule source de la conception, il est important de faire connaître les opinions et les caractéristiques de la structure des éléments de la structure.
HYPOTHESIS_GT	Bien qu'il soit basé sur une seule idée principale de la conception, il excelle en raison de la sélection intéressante et significative des caractéristiques architecturales.

Figure 8: BLEU score : MT (0.288) and GT (0.217)

SOURCE	This is a good way to keep water cool.
REFERENCE	C'est un très bon moyen de garder l'eau au frais.
HYPOTHESIS_MT	Il est donc bon de la bonne façon de conserver l'eau.
HYPOTHESIS_GT	Ceci est une bonne façon de garder l'eau fraîche.

Figure 9: BLEU score : MT (0.653) and GT (0.296)

SOURCE	A possibility is to introduce paid services: the association offers paid services which basically are interesting for all municipalities.
REFERENCE	Une des possibilités est d'introduire des services payants : l'association propose des services payants qui présentent un intérêt pour presque toutes les municipalités.
HYPOTHESIS_MT	Une demande est destinée à offrir des services : « Les services de services de santé sont les services offerts à la disposition des institutions.
HYPOTHESIS_GT	Une possibilité est d'introduire des services payants: l'association propose des services payants qui sont fondamentalement intéressant pour toutes les municipalités.

Figure 10: BLEU score : MT (0.340) and GT (0.375)

We should remain reluctant basing our evaluation exclusively on the BLEU score since the model we have been generating is not strong enough yet. Indeed, although the translations most of the time make sense to us, they don't give a faithful replica of meaning - the model still generates a sentence by reshaping how the words are linked together (more about this later in this section). Below are two interesting examples with a sentence generated by the machine translation (HYPOTHESIS_MY), and a human translation of the translation (RE-TRANSLATION). Thinking of the evaluation of the model, this helps us to convey the meaning of how the machine translation assesses a given sentence.

Examples were extracted from an article by The Atlantic, interviewing the cognitive scientist Donald Hoffman depicting the concept of reality. Obviously a very specific topic.

It is striking how the machine translation adapts the meaning of the sentence without altering the point of it, not by changing it but rather by slightly affecting it. It chooses its own trained perspective. The translation seems not to be a translation of a sentence but rather the translation of the meaning, as perceived by the machine. Last but not least, not any grammatical or orthographic mistakes have been reported.

SOURCE	The central lesson of quantum physics is that there are no public objects sitting out there in some preexisting space.
HYPOTHESIS_MT	Les principaux aspects de la complexité de la technique sont de nature, mais il n'y a pas de frontières dans l'espace de certains espaces.
RE-TRANSLATION	The main aspects of the technique complexity are natural, but there are no boundaries within the space of some spaces.

Figure 11: Human re-translation of a sentence translated with a deep neural network

SOURCE	The true reality might be forever beyond our reach, but surely our senses give us at least an inkling of what it's really like.
HYPOTHESIS_MT	La réalité ne peut jamais être prise en compte, mais nous avons notre chance de nous donner notre sens à notre sens.
RE-TRANSLATION	Reality can never be considered, but we have the opportunity to make the most of our senses to make sense of it.

Figure 12: Another human re-translation of a sentence translated with a deep neural network

5. WHAT IS NEXT

From the outlined translations and many others that have been generated, we can accept the both strong and amazing capabilities of a deep neural network using Long Short-Term Memory cells for specialized texts defying language services. In the coming weeks this model will be improved with a final training session using the remaining train set.

Additionally, another model will be trained to outperform the current one, by improving major elements. First, a more powerful tokenizer can be used when pre-processing the data. We will also focus on tuning the model by adjusting the learning rate, the decay and by initializing the weights using a adaptive pre-factor together with the random number distribution: $\sqrt{\frac{4}{n_i+n_j}}\mathcal{N}(0,1)$ with i and j corresponding to the layers linked by the weights. The optimizer will be changed to AdagradOptimizer (the optimizer that implements the Adagrad algorithm) instead of the common Gradient Descent one. Finally, the vocabulary size will be upgraded to 100K words instead of 40K.

6. REFERENCES

[1] Michael Nielsen. Neural Networks and Deep Learning. 2016.

[2] Alex Graves. Supervised Sequence Labelling with Recurrent Neural Networks. 2012.

[3] Andrej Karpathy. The Unreasonable Effectiveness of Recurrent Neural Networks. 2015.

[4] Sutskever et al. Sequence to Sequence Learning with Neural Networks. 2014.

[5] C. Olah. Understanding LSTM Networks. 2015.

[6] Cho et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. 2014.

[7] Papineni et al. BLEU: a Method for Automatic Evaluation of Machine Translation. 2002.

[8] Alex Graves. Generating sequences with recurrent neural networks. 2013.

[9] S. Hochreiter et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. 2001.

[10] T. Mikolov. Statistical Language Models based on Neural Networks. 2012.

[11] D. Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate. 2015.

[12] J. Pouget-Abadie et al. Overcoming the curse of sentence length for neural machine translation using automatic segmentation. 2014.