STAP: Simultaneous Translation And Paraphrase

Progress Report for D1.2-1.3

Dmitry Grigorev, DS2
Vakhitova Alsu, DS1
Ilalova Alina, DS2
Kozlov Pavel, RO

Introduction

In this report the progress on implementation of STAP will be reviewed. STAP - Simultaneous Translation And Paraphrase neural network model aimed at generating multiple semantically similar translations of an input text.

LSTM

Long Short Term Memory (LSTM) is a special kind of Recurrent Neural Network, and its name comes from the fact that LSTM is capable of learning long-term dependencies. To accomplish that, each network in RNN has been redesigned in the way it is shown in the Figure 1.

This architecture is able to solve the vanishing gradient problem of RNN by introducing a memory unit into the network. The Figure 1 represents that the current output is based not only on the previous output (as in RNN), but also takes into account memory from previous block. Therefore, the information from the earlier time steps can make its way to later time steps, reducing the effects of short-term memory.

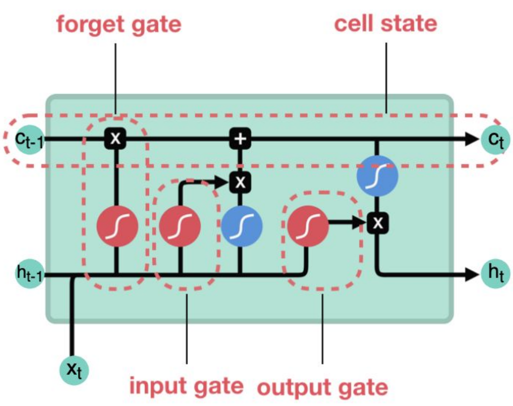

Figure 1: Building Block of LSTM

Seq-to-seq models

Deep neural networks are extremely powerful tools that can be applied to various recognition problems and achieve outstanding performance. However, these networks are limited by the representational capacity of fixed-length vectors. [Sutskever et al., 2014] have addressed this problem in their research and have introduced a method of training models to convert arbitrary-length sequences from one domain to arbitrary-length sequences in another domain.

The paper has proposed a 2-LSTM architecture, where the first deep network acts as an encoder (maps input into a fixed dimension vector) of the input sequence, and the second - decoder (maps fixed vector to an output sequence). The model showed outstanding results for English to French translation, despite being very straightforward and relatively unoptimized. Moreover, the proposed architecture outperforms the standard statistical machine translation model [Schwenk, 2013], even on long sentences. The Figure 2 shows basic scheme of sequence to sequence model.

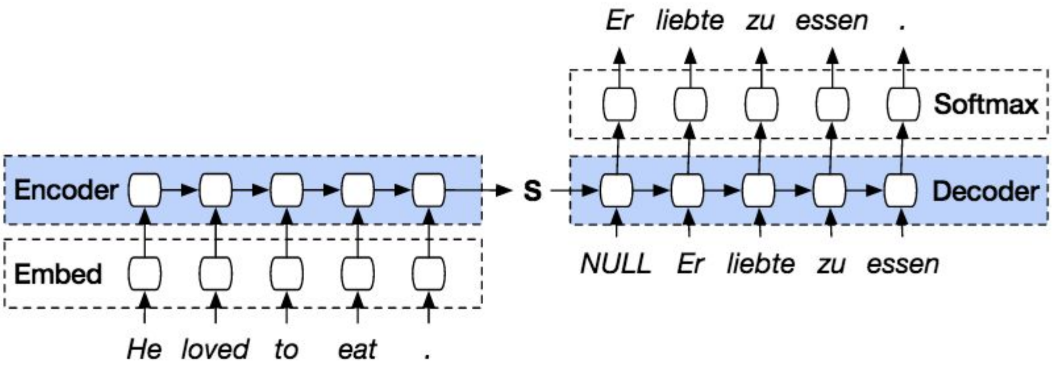

Figure 2: Sequence to Sequence model illustration

This approach is widely used in a field of speech recognition, video captioning and machine translation. Moreover, the sequence-to-sequence learning (seq2seq) method and all its corresponding techniques and modifications can be applied to a variety of other problems.

Architecture Idea

The main goal of our model is to modify prediction and loss phases of an ordinary seq2seq model. An input to that model is most often one-hot vector or index of a word (then they are combined into a sequence) and the output is a probability vector. A procedure to get the word from the target is often just argmax operation over that probabilities. This works when there is only one target option, but it can be used to if the target requires several equivalent translations to one phrase (that is our case).

To address this problem we want the model to predict not just a single word at once, but several, that are located in the same position in translation sentences. Here is an example:

Let's assume that the phrase "все ок" can be translated from Russian to English into these:

- i am ok
- feels good
- that is ok

Than we want the output for the first token of the input sequence ("все") to look like this (that is an artificial example, don't look at the details):

0	a
0	am
...	...
1	feels
0	feet
...	...
1	i
0	ia
...	...
1	that
...	...
0	zebra

As you can see, ones are set on indexes of first elements of output target sequences. However, there are several challenges with this approach:

- It is possible to have several translations starting with the same words. To address this situation, we are going to replace 1 with k/n where n is the number of provided translations for the input sequences and k is the number of translations starting with the particular word.
- Since we cannot use argmax on predicted probabilities anymore because of multiple values, during inference stage we are going to compare all the predicted probabilities with a threshold value (it will be provided as a hyperparameter) to distinguish whether a word with this index may be consider as a translation option.
- After we got all the output vectors for each positions in the translation sentences, we have to combine them into the final translation sentences. However, with the current setup the model would not know word correspondence in the adjacent vector (for example vector 0 has words "feels,

I, that” and vector 1 has “am, good, is” and we have to pair them). This is a serious problem and we are still in concern of how to solve it. Our current idea is to split output vectors 0 into one-hot and process them separately to get possible continuations for each word, but that will increase the processing time. So, if you have any idea of how to deal with this problem, we would be thankful for sharing it with us.

We will use cross-entropy as a loss function on output vectors since it is highly appropriate in our case. This is only an idea of an architecture and we haven’t discussed such things as internal layers, attention mechanisms, preprocessing and so on.

Technical Details

We will try to use 2 GPU Nvidia GeForce 2080ti for training developed model. It will help to increase the speed of training process, hence, we will be able to try out more configurations and find better parameters set. Finally, we plan to achieve substantial accuracy.

Conclusion

This report sums up results of our research and presents our idea of the project implementation. In the following week we plan to start implementation, which will follow Scrum methodology.

References

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. NIPS 2014, pages 3104–3112.

[Schwenk, 2013] Schwenk, H. (2013). Cslm - a modular open-source continuous space language modeling toolkit. INTERSPEECH.