Ukrainian Catholic University

Applied Sciences Faculty

Data Science Master Programme

Abstractive text summarization with Recurrent Neural Networks

Machine Learning project report

Authors:
Hanna Pylieva
Yuriy Mykhalchuk
Irynei Baran

June 4, 2018

Abstract

Invent a meaningful abstract.

In this work, we cast text summarization as a sequence-to-sequence problem and apply the attentional encoder-decoder RNN that has been shown to be successful for Neural Machine Translation [1].

1 Introduction

Invent a meaningful introduction. Per Shelpuk: Good project will be dedicated to an important problem and have a clear vision of what value it can bring to the potential users. All stages will be explored and analyzed, the approach for each of them is selected thoughtfully, compared to the alternatives and clearly explained. The results are evaluated and explained (explanation should provide additional information, not just restating the results or the code in English).

In the modern Internet age, textual data is ever increasing. According to this each Internet user will highly benefit from condensing data while preserving the information and meaning. This idea is a driver of growing interest among the research community for developing new approaches to automatically summarize the text. Automatic text summarization system generates a short summary that captures the main ideas of an input text. Since the advent of text summarization in 1950s, researchers have been trying to improve techniques for generating machine summaries which are not worse than human made summaries [2].

There are two prominent types of summarization algorithms.

- Extractive summarization copies parts of the source text through some measure of importance and then combines those part/sentences together to render a summary. Importance of sentence is based on linguistic and statistical features.

- Abstractive summarization generates new phrases, possibly rephrasing or using words that were not in the original text. Naturally abstractive approaches are harder as it involves robust natural language processing. For perfect abstractive summary, the model has to first understand the document and then express that understanding in succinct form possibly using new words and phrases. Abstractive summarization has complex capabilities like generalization, paraphrasing and incorporating real-world knowledge [3].

Majority of the work has traditionally focused on extractive approaches due to the easy of defining hard-coded rules to select important sentences than generate new ones. Also, it promises grammatically correct and coherent summary. But they often don't summarize long and complex texts well as they are very restrictive.

Abstractive methods, on the other hand, provide highly powerful and promising results. That is why in this project we implemented an algorithm for abstractive text summarization to build solid understanding if this approach and discover how it can be improved.

2 Model

2.1 Background

Models for abstractive text summarization fall under a larger deep learning category called sequence-to-sequence models, which map from an input sequence to a target sequence. This approach is illustrated on Figure 1. It was initially used for Neural Machine Translation as described in [4].

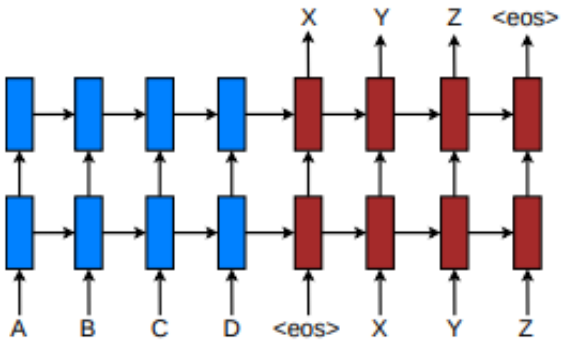

Image source: [4]

Figure 1: The model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the <eos> token. It then starts emiting one target word at a time.

An effective and standard approach to build sequence-to-sequence models is using Encoder-Decoder architecture that act as an encoder and a decoder pair. The encoder reads the entire input sequence and encodes it into an internal representation, often a fixed-length vector called the context vector. The decoder reads the encoded input sequence from the encoder and generates the output sequence. Both the encoder and the decoder submodels are trained jointly, meaning at the same time.

Naturally we need the input and output to be of variable-length. This can not be reached with ordinary Deep Neural Networks (DNNs) which require that the dimensionality of the inputs and outputs is known and fixed.

The Recurrent Neural Networks (RNNs) are generalization of feedforward neural networks to sequences. Given a sequence of inputs $(x_1, ..., x_T)$, a standard RNN computes a sequence of outputs $(y_1, ..., y_T)$ by iterating the following equation [5] :

$$\begin{aligned} h_t &= f(W^{hx}x_t + W^{hh}h_{t-1}) \\ y_t &= W^{yh}h_t \end{aligned} \tag{1}$$

where f - activation function, x_t - input vector, h_t - hidden layer vector, y_t - output vector, W - parameter matrix

An RNN can map sequences to sequences provided that the alignment between the inputs the outputs is known ahead. In other words RNN needs a defined one-to-one correspondence between a sequence of inputs and sequence of outputs which can be understood better from RNN unfold representation on Figure 2.

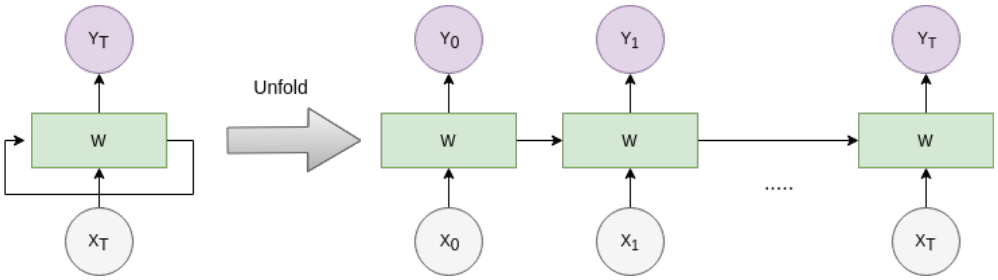

Image source: Created by author
Figure 2: Recurrent Neural Network

However, it is not clear how to apply an RNN to problems whose input and the output sequences have different lengths with complicated and non-monotonic relationships.

The simplest strategy for general sequence learning is to map the input sequence to a fixed-sized vector using one RNN, and then to map the vector to the target sequence with another RNN (the approach is represented in [1]). Since the RNN is provided with all the relevant information this will work, whereas due to the resulting long term dependencies, it will be difficult to train the RNN. Here Long Short-Term Memory (LSTM) layers will come in handy as they are known to learn problems with long range temporal dependencies [6] and will succeed in this setting.

2.2 Overview

The goal of LSTM in general is to estimate the conditional probability of outpuy sequence by input sequence $p(y_1, ..., y_{T'}|x_1, ..., x_T)$ when lengths of those sequences are possibly: $T' \neq T$. An LSTM computes this conditional probability by first obtaining the fixed dimensional representation v of the input sequence $(x_1, ..., x_T)$ given by the last hidden state of the LSTM, and then computing the probability of $y1, ..., yT'$ with a standard LSTM-LM [1] formulation whose initial hidden state is set to the representation v of $x_1, ..., x_T$:

$$p(y_1, ..., y_{T'}|x_1, ..., x_T) = \prod_{t=1}^{T'} p(y_t|v, y_1, ..., y_{t-1})$$

References

[1] Bahdanau,D.; Cho,K.; Bengio Y. *Neural machine translation by jointly learning to align and translate*. CoRR, abs/1409.0473, 2014. `http://arxiv.org/abs/1409.0473`.

[2] Gambhir, M. and Gupta, V. *Recent automatic text summarization techniques: a survey*. Artificial Intelligence Review, 47(1):1-66, 2017.

[3] Singhal,S. and Bhattacharya, A. *Abstractive Text Summarization*. Department of Computer Science IIT Kanpur, 2017.

[1] Language Model (LM) is a probability distribution over sequences of words. LSTM-LM is algorithm of using an LSTM (and softmax function) to predict the next word given your previous words.

[4] Minh Thang Luong; Hieu Pham; Christopher D. Manning. *Effective Approaches to Attention-based Neural Machine Translation.* CoRR, abs/1508.04025, 2015. `http://arxiv.org/abs/1508.04025`

[5] Ilya Sutskever; Oriol Vinyals; Quoc V. Le *Sequence to Sequence Learning with Neural Networks.* CoRR, abs/1409.3215, 2014. `http://arxiv.org/abs/1409.3215`

[6] S. Hochreiter and J. Schmidhuber. *Long short-term memory.* Neural Computation, 1997.

[7] Lopyrev, K.*Generating News Headlines with Recurrent Neural Networks.* CoRR, abs/1512.01712, 2015.

[8] Bishop, Christopher M. *Pattern Recognition and Machine Learning.* Springer, Cambridge, U.K., 2006.