Text Summarization with Deep Learning

Evangelia Kyriakopoulou

Department of Informatics

Aristotle University of Thessaloniki

June 27th, 2023

Introduction

Text Summarization is the technique of reducing a lengthy text document into a concise and well-written summary that captures the vital information and primary ideas of the original text, which is accomplished by highlighting the document's significant elements.

There are generally two different approaches used to summarize text: Extractive & Abstractive Summarization.

Introduction

Because of the massive volume of textual content on the Internet, Text Summarization using Deep Learning approaches is becoming increasingly relevant. In this report we will describe two different uses of Deep Learning algorithms to solve the Text Summarization task, one of which is State of the Art.

Deep Reinforcement Learning

Reinforcement Learning (**RL**) is a sort of machine learning that focuses on developing agents that can make data-driven decisions in order to maximize some form of reward.

Recent breakthroughs in deep learning (**DL**) have broadened the applications of reinforcement learning to real-world situations, spawning the subject of Deep Reinforcement Learning(**DRL**).

Deep Reinforcement Learning

DL and DRL involve training deep neural networks, DL focuses on learning from labeled or unlabeled data to make predictions or classifications, while DRL combines deep learning with reinforcement learning to enable an agent to learn optimal strategies through interaction with an environment.

Each time t, the agent in state s_t achieves goals through trial-and-error settings by investigating possible actions and transitions and selecting those that result in the largest receiving reward $r_t + 1$.

Deep Reinforcement Learning

So, the goal of the agent is to learn the best set of actions, termed a **policy**, in order to generate the highest overall cumulative reward. Any one set of actions that an agent takes from start to finish is termed as an episode.

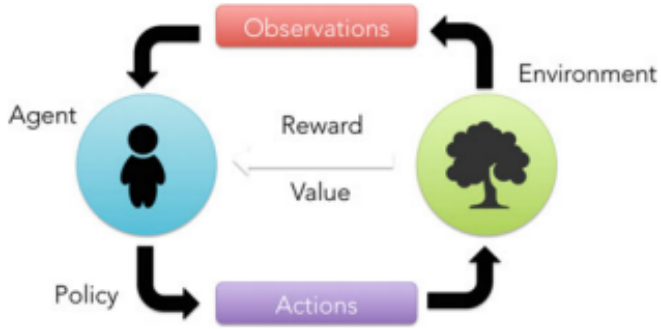

Figure 1: Agent–environment interaction in reinforcement learning.

Seq2seq

Recent advances in Deep Learning methods for sequence-to-sequence(**seq2seq**) models have led to interesting Deep Reinforcement Learning applications for NLP. Neural Machine Translation, Speech Recognition, Sentiment Classification, Text Summarization are sequence-oriented tasks. Seq2seq is one of the most common architectural techniques for these kinds of tasks.

There are two major components of a Seq2Seq model:Encoder and Decoder.

Seq2seq

The encoder and decoder work together to transform input sequences into output sequences. The encoder processes the input sequence and produces a context vector that summarizes the input information. The decoder then takes the context vector and generates the output sequence based on it, using an autoregressive approach.

The encoder-decoder architecture enables various sequence-related tasks such as machine translation. In this case of machine translation, the input is a text in one language and the output is also a text in another language: I love food $\rightarrow$ me encanta la comida

Seq2seq

One disadvantage of this strategy is that the hidden state tends to reflect the most recent information, thereby losing memory of previous content.

This forces a limitation for long sequences, where all information about that sequence must be summarized into a single encoding. Seq2seq models suffer from two main limitations:

Seq2seq

- **Train-test inconsistency:** The most common method for training a seq2seq model uses a supervised learning algorithm (**teacher forcing**), where ground truth sequences are used to minimize the maximum-likelihood (ML) loss at each decoding step. However, at test time, discrete metrics like Word Error Rate (WER) are often used to evaluate a model. Discrete metrics are non-differentiable and cannon be used in an ML framework for training. It is easy to optimize for ML loss at train time only to yield sub-optimal metrics at test time.

Seq2seq

- **Exposure bias:** While teacher forcing uses a ground truth label at each step to decode the next element of the sequence, this label is not available at test time. As a result, seq2seq models can only use its predictions to decode a sequence. This means that errors will accumulate during output sequence generation.

Seq2seq

Reinforcement Learning offers a way to overcome these limitations:

- By incorporating the discrete metric like WER as a reward function, reinforcement learning methods can avoid the train-test inconsistency.
- Since the state of a RL model is given at each time step by the output state of the seq2seq decoder, exposure bias can be avoided.

First Case Study

Following the Case Study from Chapter 13 of [1], we tried to implement the following 2 algorithms for DRL on Text Summarization, applied on a pre-trained Seq2Seq model for summarization:

- Policy Gradient
- DDQN

First Case Study

The software tools and libraries that were used in [1], were:

- **TensorFlow:** open-source software library from Google, used for machine-learning applications

- **RLSeq2Seq:** open-source library which implements various RL techniques for text summarization using sequence-to-sequence models.

- **pyrouge:**python interface to the perl-based ROUGE1.5.5 package that computes ROUGE (Recall-Oriented Understudy for Gisting Evaluation) scores of text summaries.

First Case Study

The dataset that was used in the aforementioned study was a small choice of the Cornell Newsroom Dataset [2].

The processed version that was used, consisted of 10,000/1000/1000 articles and summaries for training, validation and testing. The samples were tokenized and mapped using 100-dim embeddings generated with word2vec. For memory considerations, the vocabulary was limited to 50,000 words.

Training Issues

The first step to apply the DRL algorithms, was to pre-train the seq2seq model. The code from the original RLSeq2Seq [3] paper is no longer maintained ($\geq$ 4 years ago) because of deprecated functions and 5-year-old TensorFlow implementations.

Training Issues

After thorough bibliographical search for different DRL integration methods on Text Summarization, with newer libraries or more recent research, we decided to change the approach for text summarization.

Transformers

The Transformer architecture was introduced in June 2017. The focus of the original research was on translation tasks, and afterwards, several influential models were introduced. The ones that we tried to use where **BART/T5**.

These models are **sequence-to-sequence models**, so we used them to get our method as similar as possible to the RLSeq2Seq method that was used in the first case study.

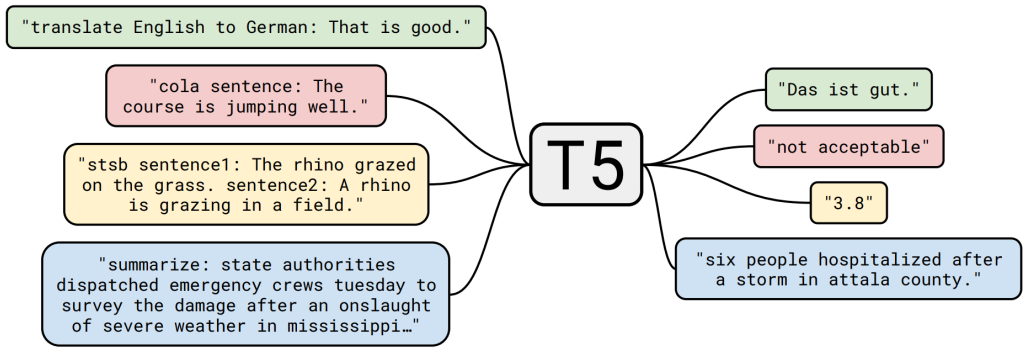

Figure 2: Model Card for t5-small [3].

Transformers

The Transformer model that we used, is trained as a language model. This means that for its training, large amounts of raw text have been used in a self-supervised fashion.

Self-Supervised learning is a type of learning in which the objective is automatically computed from the inputs.

So, the model learns just a statistical understanding of the language that is have been trained on. Since this is not very useful for a practical task, the best practice is to fine-tune the model in a supervised way, using annotated labels.

Second Case Study

Google's T5 Model [3] which was pre-trained on the C4 Dataset:

• C4: A colossal, cleaned version of Common Crawl's web crawl corpus. Based on Common Crawl dataset: "https://commoncrawl.org".

We fine-tuned our model on an easily loaded News Dataset, practically identical with the Cornell Newsroom Dataset, xsum

• Xsum consists of 226,711 news articles accompanied with a one-sentence summary. The articles are collected from BBC articles (2010 to 2017) and cover a wide variety of domains. The official random split, is 90 % -5 % -5 %.

Second Case Study

We fine-tuned the model for 1 epoch, on the xsum dataset, and then evaluated it on our test set. Below we present the comparative results of F-scores between the models. Since we were not able to reproduce the results using either the code from [1] or [2], we compared to the numbers from the printed copy of [1].

Second Case Study

It should be noted that the results bellow for T5 are for the smallest model possible and it only needed 5.5 hour fine-tuning on Google Colab and 2.5 hours with Google Colab Pro. While the seq2seq + DDQN algorithm require at least 7-8 hours to produce results like that, in a very limited Vocabulary space.

Second Case Study

Table 1: RESULTS COMPARISON

Model/ F-Score	***Rouge-1***	***Rouge-2***	***Rouge-L***
T5-Small (3 epoch fine-tuning)	31.07	10.15	24.26
T5-Small (1 epoch fine-tuning)	28.64	7.6	22.16
Seq2Seq	15.6	1.3	17.6
Policy Gradient	22.4	6.0	17.6
DDQN	34.6	21.4	30.4

Used values from [1]

[1] U. Kamath, J. Liu, and J. Whitaker, Deep Learning for NLP and Speech Recognition. Springer International Publishing, 2019. doi: 10.1007/978- 3-030-14596-5.

[2] M. Grusky, M. Naaman, and Y. Artzi, "Newsroom: A Dataset of 1.3 Million Summaries with Diverse Extractive Strategies," Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). Association for Computational Linguistics, 2018. doi: 10.18653/v1/n18-1065

[3] Y. Keneshloo, T. Shi, N. Ramakrishnan, and C. K. Reddy, "Deep Reinforcement Learning For Sequence to Sequence Models." arXiv, 2018. doi: 10.48550/ARXIV.1805.09461.

[4] [1]S. Narayan, S. B. Cohen, and M. Lapata, "Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Ex- treme Summarization." arXiv, 2018. doi: 10.48550/ARXIV.1808.08745.

[5] C. Raffel et al., "Exploring the Limits of Transfer Learn- ing with a Unified Text-to-Text Transformer." arXiv, 2019. doi: 10.48550/ARXIV.1910.10683.