Deep Learning for Text Summarization

Evangelia Kyriakopoulou
Aristotle University of Thessaloniki
Thessaloniki, Greece
evangelikk@csd.auth.gr

***Abstract*—A report on the problem of Text Summarization with Deep Learning using Deep Reinforcement Learning techniques and a Transformers-based approach.**

I. Introduction

Text summarizing 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 Summarization: identify the important sentences or phrases from the original text and extract only those from the text.

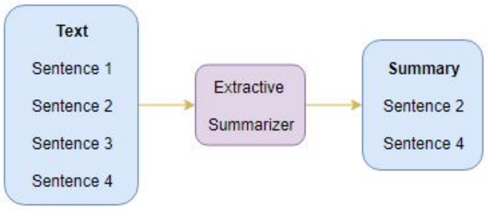

Fig. 1. Extractive Summarization.

Abstractive Summarization : create new sentences from the original text. The created sentences through the abstract may not be present in the original text.

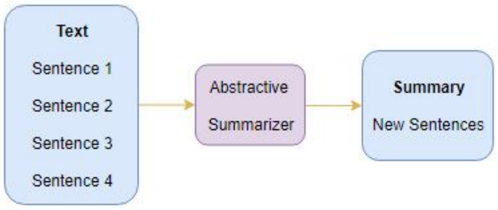

Fig. 2. Abstractive Summarization.

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 tackle the Text Summarization task, one of which is State of the Art.

II. 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**). 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 $r_t + 1$.

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 an episode. This sort of learning employs algorithms that use reinforcement strategies such as rewards and punishments to affect an agent's actions through interactions with its environment. Also, RL and DRL enables machines to learn from their mistakes and make better decisions over time as more data about their motions inside a specific environment is acquired.

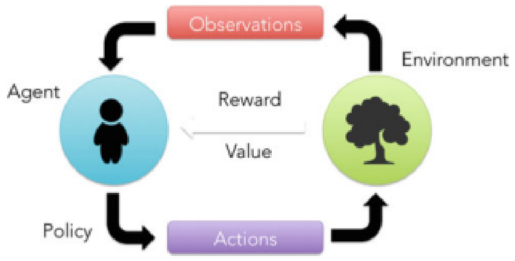

Fig. 3. Agent–environment interaction in reinforcement learning.

III. SEQ2SEQ

In Reinforcement Learning, Deep Learning methods have multiple significant functions. 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.

Encoder: takes as an input a sequence, such as a sentence, and encodes it into a fixed-length vector representation called a context vector. It captures the basic information and meaning of the input sequence by processing it step by step. The context vector serves as a compressed representation of the input sequence and preserves the encoded information.

Decoder: receives the context vector produced by the encoder. It generates the output sequence based on the context vector and its own internal state. The decoder generates the output sequence autoregressively, step by step, using the context vector as a guide.

Overall, 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

One disadvantage of this strategy is that the hidden state tends to reflect the most recent information, thereby losing memory of previous content. For extended sequences, this requires a constraint that all information about the sequence must be summed into a single encoding. Seq2seq models suffer from two main limitations:

• **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.

• **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.

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.

IV. DDQN

Double DQN: Double DQN (**DDQN**) is an extension of the Deep Q-Network (DQN) algorithm that aims to address a known issue with overestimation of Q-values in the original DQN. In DQN, the Q-values used for action selection and evaluation are estimated using a single neural network. However, this can lead to an overestimation bias, where certain actions are consistently overestimated, resulting in sub-optimal decision-making.

Double DQN was proposed as a solution to mitigate this bias. The key idea in Double DQN is to decouple the action selection and evaluation steps by introducing a second network, called the target network. The target network is a copy of the main network that is used to estimate the Q-values for action evaluation. The main network is still used for action selection, determining the best action to take in a given state. By using the target network to estimate Q-values for action evaluation. Double DQN reduces the overestimation bias. The target network's parameters are periodically updated to match those of the main network, typically after a fixed number of training steps.

This decoupling of action selection and evaluation helps stabilize and improve the learning process in reinforcement learning tasks.

A. 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 Seg2Seq model for summarization:

• Policy Gradient

• DDQN

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.

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.

B. *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) and no more DRL seq2seq methods have been actively researched since, for reasons we will mention in later chapters. The main issue with reproducing [1] results then was to manage the deprecated code and libraries.

After thorough work and debugging, we managed to make the code deployable. More than 900 changes in code and re-installations of dependencies were needed. While training the seq2seq model though on the aforementioned pre-processed training set and vocabulary, we realized that the loss stayed constant after almost 100 epochs (started from 11, decreased to 6 and stayed stable), while in [1] case study, the epoch number hyper-parameter ≥ 5000.

It was apparent that our trainings did not have good results, because in all inferences the result of our seq2seq network was the same for all test samples. For example, the results could look like this:

• Summary: " the creative team behind the musical afa from heaven a in previews at playwrights horizons grappled with translating the source filmas cinematic style for the stage "

• Prediction: " the the of ofof the of the thethe to the the a a the a the the and the the in the a of the a to the of a the to a "

We tried to fine-tune the seq2seq models with the Policy Gradient loss, but the results were obviously equally bad. We tried to play with hyper-parameters for multiple trainings no improvement was made.

We realized that the issue was with the batching and the models gradients. For unknown reasons the models kept behaving as if their weights were exactly the same as when initialized. That led us believe that even though the gradients were calculated during training, the changes were not applied to the networks weights. We could not resolve the error, since it was because of deprecated functions and 5-year-old TensorFlow implementations.

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.

V. USING TRANSFORMERS FOR TEXT SUMMARIZATION

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.

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. We fine-tuned our model on an easily loaded News Dataset, practically identical with the Cornell Newsroom Dataset, xsum [4]

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 %

Transformers, in general are huge models that need several GPU / TPU weeks or even months to be trained. Hugging Face (and several other companies) offer pre-trained big Transformer models to be able to use them without having to train them from scratch, resulting in serious decrease in time needed, and environmental damage produced.

A. *Second Case Study*

We decided to use a Google's T5 Model [5] which was pre-trained on the wiki-dpr Dataset and on the C4 Dataset:

• wiki-dpr: This is the Wikipedia split used to evaluate the Dense Passage Retrieval (DPR) model. It contains 21M passages from Wikipedia along with their DPR embeddings. The Wikipedia articles were split into multiple, disjoint text blocks of 100 words as passages. The Wikipedia dump is the one from Dec. 20, 2018.

• C4: A colossal, cleaned version of Common Crawl's web crawl corpus.

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].

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.

It is completely reasonable to assume that a slightly bigger Transformer model, would easily surpass DDQN.

TABLE I
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

[a] Used values from [1]

References

[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 Extreme 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.