Extractive Summarization Project

Thomas Reolon (221247)
University of Trento
Via Sommarive, 9, 38123 Povo,Trento TN
thomas.reolon@studenti.unitn.it

***Abstract*— The aim of this project is to build models capable of extracting relevant information from news documents. We use a neural network to identify the important sentences inside an article and build a summary from them. We train and test multiple models and methods on the CNN-dailymail dataset.**

I. Introduction

Automatic text summarization is an important task inside NLP and it is usually addressed with 2 approaches: extractive summarization, where we try to identify the most important sentences, and abstractive summarization, where we try to generate a summary that can contain words that are not present in the original text (it's a more human-like type of summary).

Common extractive techniques use score-based and statistical approaches like TF-IDF, presence of cue phrases, presence of words contained in the title and sentence centrality [1,6]. We can use these statistics to assign a score to each sentence and then build our summary using the sentences with the highest score.

Abstractive summarization is a much harder task and is typically carried out using big language models: for example common architectures for this task are T5 and BART, which are encoder-decoder neural networks [3]. The idea behind these models is the following: we take a document and encode it (we can learn the encoding or use a pre trained one like word2vec), then we pass the document to the encoder, which is typically a sequence of transformers (self-attention between the tokens of the document and a feed forward network); once we have our encoded document, we can pass it to the decoder which will create our summary (to be precise: the input of the decoder is the ground truth summary, while the information from the encoded document will be used inside the cross-attention layers). It is also possible to create these kind of models concatenating two pretrained Language Models: for example we can take a pre-trained BERT model as the encoder and a pretrained GPT model (with cross attention) as the decoder; this is called warm-starting [4].

Going back to extractive methods, state of the art results have been obtained using neural networks capable of understanding the importance of sentences, for example Zhong et al. [5] use a NN to encode a document into a latent space and then train the network by minimizing the distance between a document and its gold summary.

II. Proposed Approach

We propose a neural network that is trained in a multi objective fashion to facilitate the training of the model.
The NN can be divided in three parts:

- embeddings: which is built using a torch module that can learn how to find good embeddings for words
- encoder: it is a transformer based encoder (as in BERT, but much smaller). The first two encoders use self-attention on the whole document, while the last two only concentrate on single sentences (using an attention mask)
- decoder: finally we use a linear layer to get a score for each token, and compute the final score of a sentence as the mean of the score of the tokens in that sentence.

The training is carried on in a multi-objective fashion because the more hints we can give to our NN the easier the task become; for this reason there are four losses:

- sentence importance: we have a ground truth that tells us which sentences should have an high score
- tf-idf: we want our model to learn the tf-idf scores so that it can leverage that information in the later layers of the NN
- latent space: we obtain a vector that represents each document/summary and we want to minimize the distance between a document and its summary and maximize the distance between a document and a random selected summary.
- word importance: if a specific token is used in the summary

The CNN dailymail dataset contains pairs of articles and highlights, these highlights are mainly abstractive summaries, but a version of this dataset is also available here for the extractive task.
I had some problems using that dataset, so I used the one offered by *huggingface* and another one available here.
Preprocessing of the dataset has been carried out in different ways, for example, with the huggingface dataset we used a default tokenizer that was capable of cleaning the data and preprocessing it, while with this dataset, preprocessing had already been carried on.
Anyway, when using NN, we do not need to preprocess the data because the networks are capable of learning patterns even with "dirty" data (anyway preprocessing simplifies the task).
Why is the model using transformers? Transformers helped achieve SoA results in many NLP tasks, some alternatives to these type of model could be: dilated CNN or RNN [7]; unfortunately, a great drawback of the RNN (eg. LSTM/GRU) is that they need to be trained in a sequential fashion (not very parallelizable).

III. How To Run The Models

The project consists in a collection of jupyter notebooks, they are available on colab and can be run just by selecting *"run all"* from the menu.

Each one of them is structured in the following way: load libraries, build a dataset, train a model and test it to obtain a rogue score.
The list of notebooks is available below.

Transformer Encoder

Open in Colab

https://colab.research.google.com/drive/1IxunINb5nfeQJ9mmUs_975F76bdVWdF0

This notebook contains the implementation of the model described in the "proposed approach" section.
This model can reach satisfying results even if they are still far from the state of the art, probably because the CNN dataset was customized, making the task slightly harder. (there is another similar notebook, but it obtains worse performances)

Pretrained Abstract Model

Open in Colab

https://colab.research.google.com/drive/1VsW04wtDccE-Z3LTunyo56YmiTx2xDPz

We use the huggingface library to load a pre-trained BART model from their hub (BART is a neural network specifically thought for summarization and it is often used (eg Cachola et. al [8] finetune a BART model for the extreme summarization task)), and use it to summarize the CNN dataset. The model is pretrained, so I didn't do much, but it is interesting to try out these models too.

BERT Encoder

Open in Colab

https://colab.research.google.com/drive/1S-k5lJcS2pSxYpaVkvS4IimojoB5-gZu

We use BERT to encode the sentence and then extract a score; this kind of approach is very popular in the literature, unfortunately we don't obtain very satisfying results. The problem is probably caused by some misalignment between the hot-encoded sentence and the string sentence, moreover splits between sentences have been customized, thus making the task harder.

IV. Results

The table below reports the comparison between these methods in terms of obtained Rouge score.

	Test Set		
	Rouge-1	Rouge-2	Rouge-L
random baseline	0.24	0.05	0.22
tf-idf score	0.28	0.08	0.25
transformer -encoder	0.31	0.10	0.28
BERT simple	0.22	0.06	0.15
pretrained BART	0.32	0.13	0.23

Even if the results from BART do not seem very good, the quality of the summary is very high (in my opinion), the low Rouge-L score is due to the fact that BART creates abstract summaries, thus it is harder to create long overlaps of tokens.

V. Conclusions

Extractive methods are pretty accurate in finding important sentences inside an article; unfortunately, summaries obtained by concatenating these sentences are still far from summaries written by humans.

Abstractive summarization is trying to fill this gap by generating very accurate and smooth summaries, unfortunately the models capable of doing such a thing are enormous (hundred of millions of parameters) and hard to train, for this reason we can just take pretrained models and finetune them for our tasks.

References

[1] Rajasekaran, Abirami & Varalakshmi, R.. (2018). Review on automatic text summarization. International Journal of Engineering and Technology(UAE).

[2] Sebastian Ruder. (2018). A Review of the Neural History of Natural Language Processing.

[3] Patrick von Platen. (2020). Transformers-based Encoder-Decoder Models

[4] Patrick von Platen. (2020). Leveraging Pre-trained Language Model Checkpoints for Encoder-Decoder Models

[5] Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, Xuanjing Huang. (2020). Extractive Summarization as Text Matching

[6] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assef. (2017) Text Summarization Techniques: A Brief Survey

[7] Rimacyn. (2020). Auto-highlighter: extractive text summarization with sequence-to-sequence model

[8] Isabel Cachola, Kyle Lo, Arman Cohan, Daniel S. Weld. (2020). TLDR: Extreme Summarization of Scientific Documents