TLDR for Scientific Papers

Dmitrii Iakushechkin
yakushechkin.dmitry@gmail.com
University of Passau

1 EXPERIMENT IMPLEMENTATION

We will discuss all attempts, techniques which were utilized for reproducing the SciTLDR results [1], in order from extractive (PacSum) to abstractive (BART) methods. Since the goal is to reproduce the results, we tried to follow steps from the original article wherever possible.

The main problem for our experiments is a lack of codes and guides for that particular summarization problem. This circumstance created the conditions in which we often used intuition and were creative. All above-mentioned models are fine-tuned on SciTLDR, except for PACSUM.

1.1 PacSum on CNN|DM

The first part of the experiment was to get to know the PacSum implementation and run the model on the original dataset which was used during training with the correct schema, structure.

The authors of the paper *Sentence Centrality Revisited for Unsupervised Summarization* [2], Hao Zheng and Mirella Lapata, uploaded a fine-tuned version of PacSum model, but it was unclear what data had been used for that process. So, we sent a request to the authors and got a response: "We used the datasets CNN|Daily Mail and NYT". Therefore, we decided to exploit the CNN|DM test dataset to run the primary experiment.

1.1.1 ***Experimental Setup****.* The PacSum code does not have lots of dependencies, just the following: Python 3.6, PyTorch >= 1.0, NumPy, Gensim, PyRouge. But the real challenge was to configure PyRouge package because we got lots errors during running PacSum. The PyRouge is based on ROUGE-1.5.5 package. The following issues were solved:

- *The file 'settings.ini' is not found*: The solution is to clone both of the PyRouge repositories in different folders and then explicitly point to the ROUGE-1.5.5 in the tools folder.
- *Cannot locate XML/Parser.pm*: The solution is to install libxml-parser-perl package by Advanced Package Tool.
- *Cannot open exception WordNet-2.0.exc.db file for reading*: The problem was solved by building a new link to the WordNet-2.0.exc.db exception.

The original PacSum code[1] is designed for utilizing NVIDIA CUDA devices. Thus, we adapted the code to run the pre-trained model on a Central processing units (CPUs), but due to a huge number of training parameters and high CPU load, the process was long, and we found a solution in the form of switching to Google Colaboratory with free access to the Graphics processing units (GPUs) accelerator.

[1] https://github.com/mswellhao/PacSum

The PacSum repository has links to the fine-tuned version of the model and data used in the paper. The data is already in HDF5 format (a data type for storing data frames). A running process comprises several possible sets of arguments which must be explicitly specified:

- *Mode: 'tune' or 'test'.* Since the SciTLDR authors didn't fine-tune the model, we use 'test' mode.
- *Representation: 'bert' or 'tfidf'.* It's possible to choose a symbolic sentence representation TFIDF or a pre-trained BERT as a sentence encoder. We tried them both. The model with the TFIDF representation works much faster, moreover, as shown in the paper [2], there is a small difference between the results, so we utilize 'tfidf'.
- *Paths to data files.* Paths to the model, training and test data.

Finally, we tested the fine-tuned PacSum version on the CNN|Daily Mail, just to make sure it works correctly. Our ROUGE metrics and original are quite similar, but the results and analysis will be given in the section *Results and Analysis*.

1.2 PacSum on SciTLDR

The PacSum model had been successfully tested on CNN|DM, so we started preparing the SciTLDR data for the same test experiment.

1.2.1 ***Data conversion****.* The raw SciTLDR data is in JSONL format, while the PacSum can only work with H5DF data. The data schema is also different and we should take this into account.

The PacSum data has keywords: "article", "abstract", "rouge", "oracle_sens". The last two are pre-computed and added for generality, they may only be necessary for use by other researchers for various purposes and are not used in our study. An oracle instance consists of sentence's ids which maximize the ROUGE score against the gold summary. The PacSum data schema is shown below:

```
{
  "article":[
    "sent0",
    "sent1",
    "sent2",
    ...
  ],
  "abstract":[
    "sent0",
    "sent1",
    "sent2",
    ...
  ],
  "rouge": 0.55,
  "oracle_sens": [1,2,12]
}

where 0.55, 1, 2, 12 - just random numbers for
    demonstration.
```

The main idea behind the SciTLDR data is quite similar. The schema has the following keywords: "source", "source_labels", "rouge_scores", "paper_id", "target", "title". An source_labels instance is the same oracle but in one-hot representation. The SciTLDR data schema is shown below:

```
{
  "source":[
    "sent0",
    "sent1",
    "sent2",
    ...
  ],
  "source_labels":[binary list in which 1 is the
      oracle sentence],
  "rouge_scores":[list, precomputed rouge-1
     scores],
  "paper_id":"PAPER-ID",
  "target":[
   "author-tldr",
    "pr-tldr0",
    "pr-tldr1",
    ...
  ],
  "title":"TITLE"
}
```

We only need "source" and "target" for testing the model. Firstly, we simply changed the keyword's names, e.g., a keyword "source" turned into "article". After this manipulation, there is still a problem with targets. The internal PacSum evaluation metric only works with one possible summary output. We used the following strategy to solve this problem: create as many samples as we have TLDR's, i.e., each entry has a unique 'abstract', but not unique "article". This strategy only related to the test dataset, because train and validation datasets have only one possible output. The test dataset has 618 articles and 1324 TLDRs, consequently we artificially have increased its size by 2 times and got for each entry only one possible output.

$\boxed{\text{The preprocessed test dataset has 1324 entries.}}$

Thus, we have written functions to convert JSONL to H5DF format with the required schema and format for both tasks: $SciTLDR_A$ (the input is the abstract) and $SciTLDR_{AIC}$ (the input is abstract+introduction+conclusion).

1.2.2 ***Experimental Setup.***

Finally, we tested PacSum on $SciTLDR_A$ and $SciTLDR_{AIC}$ using TFIDF's data representation. The experiment on $SciTLDR_A$ data shows very close results, but there is a gap between the original and our metrics for $SciTLDR_{AIC}$. A possible reason is the way we modified the dataset for using an unique output. We sent a request to Isabel Cachola, Kyle Lo and Arman Cohan (authors of the paper *TLDR: Extreme Summarization of Scientific Documents* [1]) with questions about what evaluation methods were used, whether they used the internal PacSum module for evaluation and whether they rewrote this module, if so, how. In the section *Results and Analysis*, we will discuss how we have brought our results closer to the benchmarks.

1.3 BART on SciTLDR

The implementation is based on Fairseq toolkit[2] developed by Facebook AI. The workflow for that part of our research is as follows:

(1) Setup the environment
(2) Load the model
(3) Preprocess the data
(4) Set the hyperparameters and fine-tune the model
(5) Evaluate the model

Let's discuss them sequentially and the problems we have encountered.

1.3.1 ***Environment Setup.***

Since we had a negative experience launching models on CPU, we immediately started working on Google Colab with GPU accelerator, but it wasn't enough, and here's why. Google Colab provides the following features:

- *Disk storage:* about 38 GB in combination with GPU and 78 GB with CPU.
- RAM: 12 GB.

The Fairseq running script saves checkpoints (training parameters) after each training epoch, furthermore, it saves the best and the last versions of weights as separate files. Each checkpoint takes about 4 GB, so if we train the model for 6 epochs, we already have more than 30 GB. As we said, Google Colab provides with 38 GB, but this storage is already occupied by the pre-trained model and dependencies. onsequently, the maximum number of epochs is 5, which is not enough for our needs.

[2]https://github.com/pytorch/fairseq

The solution was to utilize a remote machine from the University with about 370 GB of memory and a powerful GPU. We used the SSH Protocol to establish the connection. Additionally, we created python virtual environment for convenient use of the machine with different sets of packages. For continuous training of the model, each of which took several hours, we created sessions using *tmux*. This allowed us to disconnect from the connection or conveniently use the machine for multiple tasks. In addition to all this, when using a machine, you need to transfer files from the local machine to the remote one, and for this purpose, we used the *scp* command.

Most of the time we used the terminal command line, but sometimes it was convenient to use the Jupyter Notebook. The problem with the Notebook is that it uses the default python environment and to execute the code in the Notebook correctly, we added another profile (kernel) to it, which allowed us to work in a virtual environment created for the experiment.

1.3.2 ***Load the model.*** For this step, we downloaded the archive with the pre-trained BART model, a configuration file, a vocabulary from the official Fairseq repository.

1.3.3 ***Data preprocessing.*** The authors of SciTLDR provided a data preprocessing script for models using the Fairseq toolkit. We used it, but to run the script, we had to install all the necessary libraries and packages. Unfortunately, one of the libraries was causing problems and we solved this by forking the original repository and changing the requirements file. Otherwise, this procedure did not cause problems.

1.3.4 ***The hyperparameters for fine-tuning.*** This step was the most problematic. We used all the available hyperparameters that were specified in the original article, but the authors specified only a part of them, such as the number of steps or the sequence length. Other parameters that are also necessary for fine-tuning were selected manually. The only clue we had was an example of fine-tuning a BART on CNN|DM[3].

1.3.5 ***Evaluate the model.*** To evaluate the fine-tuning model on SciTLDR, we used a checkpoint with the best results. Besides, we used a special module for evaluating models located in the original SciTLDR repository. To use it, we had to go back to solving the problems with PyRouge, which are already described in the PacSum implementation section.

2 RESULTS AND ANALYSIS

In this section, we will show the results obtained and compare them with the control results obtained by the researchers. Moreover, the comparison is made not only with the results from the SciTLDR article but also, for example, with PacSum to check the correctness of our implementation. We will follow the same sequence that was used to describe the experiments. The tables use the ROUGE F1 metric: ROUGE-1 and ROUGE-2 are shorthands for unigram and bigram overlap, ROUGE-L is the longest common subsequence.

2.1 PacSum

Let's look at the results achieved. All questions regarding the implementation of the experiments have already been discussed above. For a visual comparison of the results, we have provided Tables 1, 2 that differ in the type of representation: a symbolic sentence representation TFIDF or a pre-trained BERT as a sentence encoder.

2.1.1 ***TFIDF representation.*** As can be seen from the results presented in the Table 1, the first experiment on CNN|DM showed very similar results. The same thing is observed for $SciTLDR_A$, despite the approach used to artificially obtain a single output in the data set. As for $SciTLDR_{AIC}$, the situation looks worse here, the results obtained, because there is a gap between the original and our metrics. This gave us the idea trying a different representation. So we changed the representation from TFIDF to BERT.

Table 1: The PacSum experiment results on the $CNN|DailyMail$, $SciTLDR_A$ and $SciTLDR_{AIC}$ datasets using ROUGE F1. The TFIDF sentence representation was used.

Dataset + Approach		ROUGE-1	ROUGE-2	ROUGE-L
CNN+DM	original	39.2	16.3	35.3
	ours	**37.8**	**15.5**	**34.1**
$SciTLDR_A$	original	22.5	7.5	17.4
	ours	**23.8**	**7.8**	**19.5**
$SciTLDR_{AIC}$	original	23.0	7.4	17.8
	ours	**17.2**	**4.9**	**13.8**

2.1.2 ***BERT representation.*** Looking at the results obtained (Table 2) using the BERT representation, we can conclude that the authors of the SciTLDR paper used the same approach. The discrepancy between our results and the control results is much smaller now. The remaining discrepancy may be caused by the way we achieved a single possible output already mentioned above. At this stage, we consider that a positive and very good result has been achieved with reproducing the PacSum implementation.

2.2 BART

In the course of experimenting with hyperparameters, we obtained various results. But in table 3, we give the best achieved. These poor results can be caused by:

[3]https://github.com/pytorch/fairseq/blob/master/examples/bart/README.summarization.md

Table 2: The PacSum experiment results on the $SciTLDR_A$ and $SciTLDR_{AIC}$ datasets using ROUGE F1. The BERT sentence representation was used.

Dataset + Approach		ROUGE-1	ROUGE-2	ROUGE-L
$SciTLDR_A$	original	22.5	7.5	17.4
	ours	**23.2**	**7.2**	**19.0**
$SciTLDR_{AIC}$	original	23.0	7.4	17.8
	ours	**23.5**	**7.9**	**19.4**

1. As mentioned earlier, we used known hyperparameters, but we guessed a significant part of them since they were absent in the article. Given that we had already sent a request regarding PacSum and did not receive a response, we did not send another one.

2. The authors of SciTLDR have their module for evaluating models, judging by the written code, this is exactly the module that the authors used not only for TLDRGEN, but also for BART model, since these models use the same Fairseq toolkit.

Table 3: The BART experiment results on the $SciTLDR_A$ dataset using ROUGE F1.

Dataset + Approach		ROUGE-1	ROUGE-2	ROUGE-L
$SciTLDR_A$	original	41.6	19.8	33.8
	ours	**19.3**	**2.6**	**14.7**

REFERENCES

[1] Isabel Cachola, Kyle Lo, Arman Cohan, and Daniel S. Weld. 2020. TLDR: Extreme Summarization of Scientific Documents. *ArXiv* abs/2004.15011 (2020).

[2] H. Zheng and Mirella Lapata. 2019. Sentence Centrality Revisited for Unsupervised Summarization. In *ACL*.