COL772 A3: Lay Summarization of Biomedical Research Articles

https://biolaysumm.org/

1 Motivation

Biomedical science has made many important discoveries, especially in healthcare. Most publications in this area use technical jargon understood only by area experts, doctors, and researchers. As a result, they are difficult for the general public to understand.

In this assignment, You will participate in the ongoing competition BioLaysumm. Your goal is to use a **pre-trained language model** (PLM) and train a system that produces a layman's summary given a research publication from the biomedical domain.

2 Task Definition

Given the abstract and main text of an article, your goal is to train a PLM-based model that generates a layman's summary for the article. The task has two datasets - eLife and PLOS. An example from the dataset is given below.

```
{
    # article lay summary
    "lay_summary": "Messenger RNAs carry the...",

    # article main text (abstract included), sections separated by "/n"
    "article": "Gene expression varies widely ...",

    # the article headings, corresponding to the sections in the text
    "headings": ["Abstract", "Introduction", "Results", ..],

    # keywords describing the topic of the article
    "keywords": ["genetics", "biology", "genomics", ...],

    # article id
    "id": "journal.pgen.1002882",
}
```

Your approach can make use of the meta-information headings and keywords in addition to the main article and text.

3 Datasets

The task has two datasets - eLife and PLOS. You can download the datasets from https://www.codabench.org/competitions/1920/. Both datasets have train, validation, and test splits where the test split does not have the gold layman's summary.

4 Evaluation Criteria

Your model will be evaluated based on the following metrics

1. **Relevance**: ROUGE (1, 2, and L) and BERTScore [Zhang et al., 2019, Lin, 2004].
2. **Readability**: Flesch-Kincaid Grade Level (FKGL)
3. **Factuality** - AlignScore [Zha et al., 2023],

The evaluation script is available here.

5 Possible Solutions

You can model the task as *seq2seq* and fine-tune a PLM on the given datasets. You can start with **small** or **base** versions of the Flan-T5 and BioGPT. No other pre-trained language models are allowed for fairness, and we will test you in a limited resource setting. You may also want to automatically decide which sections in the input article to consider when producing the summary.

You may explore domain ideas from the text summarization domain. For instance, you can use coverage loss to avoid focusing on the same section of the article [See et al., 2017]. You can augment the available dataset using techniques like back-translation, self-training, and para-phrasing. You may further explore a multi-stage summarization similar to Zhang et al. [2021] to handle large articles.

Finally, you can consider using external knowledge sources that provide layman's explanations for biomedical terms. **Make sure you consult the TAs before committing to this approach.** You can refer to the competition page for additional ideas.

6 Implementation Details

6.1 Allowed Libraries:

- PyTorch
- Huggingface transformers, accelerate and datasets
- Gensim for accessing Glove, word2vec, or FastText embeddings
- Numpy, Pandas, Sklearn, Scipy
- Any other standard Python library included in the default installation

If you need to use additional packages, you can request them on Piazza. They will be allowed subject to verification.

6.2 Output Format

We will follow the output format, which is the same as the competition's. Your code must produce a text file containing one lay summary per line.

6.3 Parameter Efficient Fine-tuning (PEFT)

Feel free to use PEFT techniques such as adapters, LoRA, and mixed-precision training discussed in the class. You can request the required packages for PEFT on Piazza. They will be made available during training.

7 Submission Instructions

7.1 Deadline

The assignment must be submitted by **April 25, 2024, 11:59 PM**.

7.2 Submission Format

You must submit a zip file on Moodle with the name kerberos_id.zip (E.g. csz208845.zip). Unzipping this should generate a directory named kerberos_id (E.g. csz208845). This directory must contain a run_model.sh and writeup.txt. The command to train the model is

```
bash run_model.sh train <path_to_data> <path_to_save>
```

path_to_data is a directory containing all the train and validation JSONL files for both eLife and PLOS datasets. *path_to_save* is also a directory where the trained model must be saved.

The command to run inference is

```
bash run_model.sh test <path_to_data> <path_to_model> <path_to_result>
```

path_to_data is a directory containing the two test files, *path_to_model* is a directory containing the model stored by the training command. Finally, *path_to_result* is a directory where the predicted summary files pols.txt and elife.txt must be stored.

We will release detailed instructions soon for submitting the trained model and generated summaries.

8 Submission Guidelines

Your submission must adhere to the following instructions:

- The assignment can be done individually or in a group of two. No quality difference is expected between groups of size one or two, and they will be evaluated with the same criteria.
- You must register in BioLaySumm Codabench competition here
- **You must have a single model for both datasets.**
- **You must fine-tune a PLM for this assignment.**
- **The competition only allows a maximum of 15 test submissions. Use them wisely.** Also, evaluation time on the competition page is 2 hours.
- Only the packages above are allowed to be used. Any additional packages can be requested over Piazza and are only allowed after verification.
- Feel free to use parameter-efficient training techniques like - adapters, prefix tuning, LORA, etc for faster training. You can raise a request if a certain package you need is not available.
- Your code will be checked on a single V100 GPUs on HPC.

- You must not discuss this assignment with anyone outside the class. Make sure you mention the names in your write-up in case you discuss with anyone from within the class. Please read academic integrity guidelines on the course home page and follow them carefully.
- We will run plagiarism detection software. Any person found guilty will be awarded a suitable penalty as per IIT rules.
- Your code will be automatically evaluated. You will get a 20% flat penalty if it does not conform to output guidelines.
- Your code must be your own. You are not allowed to use ChatGPT or any coding assistant for this assignment.
- We will run plagiarism detection on the model generated code, and any similarity will be penalized as per IIT rules.

9 Clarifications

For any doubts or clarifications, please use the Piazza forum. Ensure your code is well-commented, follows the submission guidelines, and is tested thoroughly to avoid any penalties related to non-conformance to output formats or runtime errors. It is your responsibility to seek clarification on any aspect of the assignment you find unclear before the deadline.

References

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *ArXiv*, abs/1904.09675, 2019. URL `https://api.semanticscholar.org/CorpusID:127986044`.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Annual Meeting of the Association for Computational Linguistics*, 2004. URL `https://api.semanticscholar.org/CorpusID:964287`.

Yuheng Zha, Yichi Yang, Ruichen Li, and Zhiting Hu. Alignscore: Evaluating factual consistency with a unified alignment function. In *Annual Meeting of the Association for Computational Linguistics*, 2023. URL `https://api.semanticscholar.org/CorpusID:258947273`.

A. See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. *ArXiv*, abs/1704.04368, 2017. URL `https://api.semanticscholar.org/CorpusID:8314118`.

Yusen Zhang, Ansong Ni, Ziming Mao, Chen Henry Wu, Chenguang Zhu, Budhaditya Deb, Ahmed Hassan Awadallah, Dragomir R. Radev, and Rui Zhang. Summ^n: A multi-stage summarization framework for long input dialogues and documents. *ArXiv*, abs/2110.10150, 2021. URL `https://api.semanticscholar.org/CorpusID:239049877`.