Foundations of Natural Language Processing

Peking University, 2024

Lab 3: Due Sunday, June 16, 2024 by 23:59

1. Directions

PLEASE read these instructions to ensure you receive full credit on your homework.

- Submit your homework as a zip file through **Course**, which should include one report in PDF, source code in python and prediction results in json.
- The report should be generated from the **LaTeX template** provided in Lab 1 and you do not need to submit the data you used.
- The code should be paired with a README file describing dependencies, code structures, etc.
- There is no need to submit the data you used and the model weights. Your grade will be based on the contents of the report, prediction results and the source code.

LATE SUBMISSION POLICY

- Late homework will have 5% deducted from the final grade for each day late.
- No submission will be accepted after June 23, one week after the due date. Your homework submission time will be based on the time of your last submission to Course.

Therefore, do not re-submit after midnight on the due date unless you are confident that the new submission is significantly better to overcompensate for the credits lost. You can resubmit as much as you like, but each time you resubmit be sure to upload all files you want to be graded! Submission time is non-negotiable and will be based on the time you submitted your last file to Course. The number of points deducted will be rounded to the nearest integer.

2. Problem Description

Fact verification is the task of predicting veracity of a claim based on evidence that supports or refutes the claim. The pipeline to solve this task is generally divided into two steps, evidence retrieval and claim verification. In this lab, we will implement the first step, that is retrieving sentence from Wikipedia as evidence to verify the claim. We provide each claim with some candidates from Wikipedia and your task is to retrieve five sentences as evidence for the verification step.

We will cover BM25[1] during class, a retrieval method that matches keywords efficiently with an inverted index and can be seen as representing the query and context in high dimensional, **sparse** vectors. Conversely, the **dense** retrievers use latent semantic encoding for retrieval. Our assignment is to implement a dense retriever. For details, please refer to 2.1 and 2.2.

Note: Students with limited compute resources can choose to run experiment on a subset of the dataset. If you choose to do so, you need to report the size of data you use for experiment. You can use any packages you want to implement your retriever.

Requirements of Report： Your report should include comprehensive introduction of sparse and dense retrieval methods, implementation details,

experiment results and result analysis. And you need to submit your prediction results on the dev and test set following the format described in readme.md.

2.1 Dense Retriever

In this section, we will evaluate the performance of dense retrieval **without finetuning** on the training set.

2.1.1 Implementation (40%)

Read the paper introducing Dense Passage Retriever[2] and then implement the dense retriever algorithm with a pretrained language model. Run your algorithm to retrieve sentences as evidence for claims in the development (dev.json) and test sets (test.json). You should submit the source codes of your implementation and introduce the implementation in the report.

2.1.2 Evaluation (20%)

Please retrieve five sentences as an evidence set and evaluate this set using recall as metric. The performance of the dev set should be written in the report and please also submit the predictions on both the development (dev.json) and test sets (test.json), following the format specified in the readme.md file.

Your recall on the development set should be higher than 40%, and the performance difference in recall between the dev and test sets is expected to be within 10%. For example, if the recall in dev set is reported to 50%, we suppose the recall in test set should not be lower than 40%.

2.2 Finetuning Dense Retriever (40%)

If you do not have enough compute resource to finetune dense retriever, you

can choose to do 2.3 instead.

Finetune your dense retriever on the training data (train.json) and compare the evaluation result on the development set before and after training. Include implementations details in your report. Describe how you construct your training samples from the candidates to train your retriever, especially how you choose the negative samples for training. Please also perform some error analysis.

2.3 Sparse Retriever (40%)

If you have already done 2.2, you can skip this section. If you choose to implement both 2.2 and 2.3 in this lab, your score will still be graded ***only*** *based on your results in 2.1 and 2.2.*

Implement sparse retriever such as BM25 to retrieve sentence evidence for the development set and compare the evaluation result with dense retriever. Include implementation details in your report. Please also perform qualitative analysis on your retrievers. For example, demonstrate some examples to compare the performance of sparse and dense retrievers.

3. References and Useful resources

[1] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., & Gatford, M. (1995). Okapi at TREC-3. Nist Special Publication Sp, 109, 109.

[2] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6769—6781, Online. Association for Computational Linguistics.

[3] https://colab.research.google.com/ You can use colab for free GPUs.

[4] https://www.sbert.net/ SentenceTransformers is a Python framework for state-of-the-art sentence, text and image embeddings. You can use pre-trained sentence transformer models to implement your dense retriever.

[5] https://github.com/facebookresearch/faiss Faiss is a library for efficient similarity search and clustering of dense vectors.

[6] https://github.com/dorianbrown/rank_bm25 A collection of algorithms for querying a set of documents and returning the ones most relevant to the query.

[7] Rami Aly, Zhijiang Guo, Michael Sejr Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. 2021. The Fact Extraction and VERification Over Unstructured and Structured information (FEVEROUS) Shared Task. In Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER), pages 1—13, Dominican Republic. Association for Computational Linguistics.