CS577 Homework 2

Due April 20, 2025 (11:59pm EDT)

---

1. Introduction

Large Language Models (LLMs) have demonstrated remarkable improvements in mathematical and symbolical reasoning through *Chain-of-Thought (CoT)* (Kojima et al., 2023; Wei et al., 2023) prompting. Recent approaches use inference time scaling by prompting LLMs to produce long context CoT for guiding the models. For example, models like OpenAI-o1 OpenAI (2024) and DeepSeek-R1 DeepSeek-AI et al. (2025) show that reasoning quality increases significantly when intermediate thoughts are generated before making predictions.

In this homework, we investigate whether **a small encoder-only classification model (BERT)** Devlin et al. (2019) can benefit from such CoT supervision, even though BERT is not auto-regressive. The goal is to determine whether intermediate reasoning steps generated by another model (Flan-T5) Chung et al. (2022) can improve BERT's final predictions in multiple-choice QA tasks.
Flan-T5 is an encoder decoder model which is pre-trained in an instruction prompt fashion Ouyang et al. (2022) to support the CoT style reasoning.

Dataset: We use the MMLU Hendrycks et al. (2021) dataset, which includes multiple-choice questions across various disciplines. Each question is associated with four options (A, B, C, D), out of which only one option is the correct choice.

- **Train Split:**
  - high_school_mathematics
  - machine_learning
  - high_school_computer_science
- **Test Split:**
  - college_computer_science

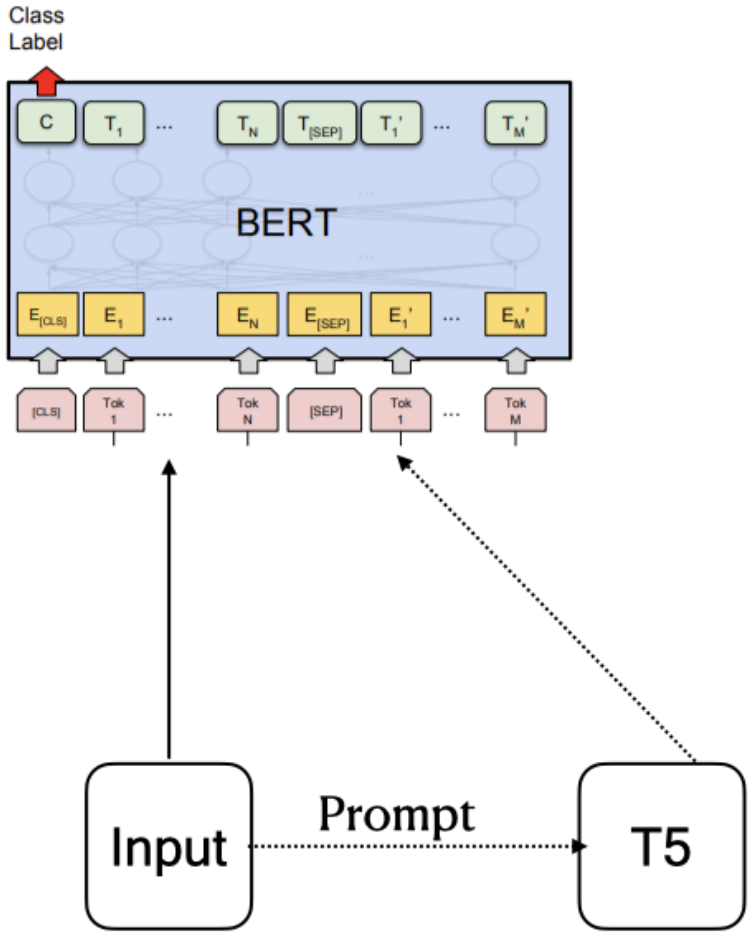

Figure 1: Pipeline overview: Input contains the QA example as input to BERT and optionally (dashed arrow) to T5 for augmenting CoT for the answer label classification

1.1. Design.

1.1 We evaluate the usefulness of reasoning in the form of Chain-of-Thought (CoT) to BERT using the input format shown in Fig. 1. BERT receives both the original question with options, and optionally the CoT reasoning as part of its input.

1.2 We use the Flan-T5 model, which is trained on CoT-style tasks, to generate the CoT reasoning for BERT input context. This model is frozen unless otherwise fine-tuned by you.

1.3 The next step is to fine-tune the Flan-T5 model. This allows learning from task-specific CoT style and encourages better generalization. We provide gold CoT for the train split for fine-tuning. The gold CoTs were generated using another LLM, we control for the quality but given the fact that they are generated by LLM would add some noise. However, it is a good starting point for fine-tuning.

1.2. (Extra Credits) Handling Longer Context BERT. BERT has a maximum context length for the input, in our design the max length is 512. However, in some cases

the input with the question + options + CoT could be longer than 512, which would be truncated, therefore throwing away some information. One technique to represent longer context is using sliding window to segment the long input into multiple sequences. You start with the first 512 tokens of the input, then slide the start to be the 256 token (512/2) and take the next 512 tokens, and continue to iterate. Basically you always take a context length of T tokens (T = 512) to form a sequence and then the next sequence starts after T/2 tokens, so that and every sequence pair overlaps T/2 tokens. Then pass all the list of sequences individually to BERT, to obtain list of [CLS] representations. You then need to pool all these [CLS] representations to obtain a final representation. This can be implemented using a BiLSTM or you may use a fixed size neural-network to compute the final embedding. The sliding window technique is discussed in this paper Sec 2.2 BERT Overlap (Joshi et al., 2019).

2. Coding

The goal is to implement the training functions for BERT and T5 within the provided codebase. You will find a starter code repository with:

- 'main.py': the main design file (with evaluation)
- 'README.md': instructions and dataset field descriptions
- 'data/': contains 'train.csv' and 'test.csv' files

You will need to implement the training methods and complete the `generate_CoT(...)` method. You have the implementation flexibility, by making design choices, helper functions, adding optional arguments to any method, but you **must not change** "the function signatures and return types of all the provided methods". **Follow** the instructions in README. Also, by default we use the small size of the LMs, please do not change them as we are constrained with the compute budget.

You are required to save and evaluate predictions for all the possible designs, i.e. the six combinations of flag settings described in the README:

1. No training, no CoT
2. No training, with CoT
3. Train only BERT (no CoT)
4. Train only BERT with CoT
5. Train only T5 with CoT
6. Train both BERT and T5 with CoT

Each run should produce a CSV file in the format:

`test_pred_Bert_<bert_train>_CoT_<cot>_T5_<t5_train>.csv`

Most of the logic is handled in `def predict(...)` and `main()`.

GPU Access: You may use GPUs for faster training. The CS department clusters provide access to GPU nodes. Follow these instructions:

- SSH login: `ssh username@data.cs.purdue.edu` (works without VPN).
- From there, use `ssh queue` to access GPU nodes.
- Use `squeue` to list available GPU nodes. Jobs can be scheduled using `slurm`.

Coding Evaluation: We will run your code for each combination of train, no-train, and CoT, which should produce a `csv` and the `accuracy` score.
Points Breakdown:

- Coding: **75 points** + Extra Credits (10 points)
  - **50** for working code (partial credit for partial implementation)
  - **25** for max Acc $> 30\%$ (we compute the max accuracy across all the 6 settings)
  - **10** for optional extra credits
- Discussion: **25 points**

Discussion Questions

(A: Abhishek will grade the question, N: Nathaniel will grade the question)
Answer the following questions:

1. **Compare Results Across Settings.** Create a table reporting the accuracy from each of the six experiment settings. Use the *no-training, no-CoT* configuration as your baseline (i.e., pre-trained BERT predictions on the test set).

   Then, answer the following:

   - How do the other five settings compare to the baseline?
   - Does adding Chain-of-Thought (CoT) help or hurt performance?
   - What impact does training BERT or T5 have on accuracy?

   Support your analysis with two examples from the test set:

   (a) One example where BERT alone is correct but BERT + CoT is wrong

   (b) One example where BERT + CoT is correct but BERT alone is wrong

   Briefly describe what the CoT helps capture in these examples, or why it may confuse the model. (8 points; A)

2. **Generalization and Transfer.** The training data includes high school-level mathematics and computer science questions. The test set contains college-level computer science. Reflect on the following:

   - Do you believe your BERT model is learning general concepts that transfer to more difficult questions?
   - Use both your accuracy table and qualitative examples from above to support your reasoning.
   - Do you see evidence that the model is combining concepts from different subjects (e.g., math + CS)?

   Now, suppose the task was much simpler — like basic grammar correction. Would your answer about the usefulness of CoT reasoning or model (any Language Model) training change? In what types of tasks does reasoning seem most beneficial, and where does it add little or no value? (6 points; A)

3. **Applying CoT Elsewhere.** Consider the task of training an autoregressive LM to find the shortest path between two vertices, given a set of labeled vertices and a set of undirected edges between them (i.e., $G = (V, E)$) in text form.

   - What might a CoT response look like in terms of this problem?
   - Is there a specific type of graph that would allow the model to make good use of CoT?
   - Is there a specific type of graph that the model would likely be unable to solve even with CoT?

(6 points; N)

4. **Attention and CFGs** Consider the following CFG from the lecture slides:

```
S   → NP VP
VP  → V NP
PP  → P NP
NP  → NP PP
NP  → N
V   → 'saw'
P   → 'with'
N   → 'binoculars' | 'Sally' | 'Alex'
```

What might one of a perfectly trained auto-regressive transformer's attention pattern (i.e., attention weights) look like if the model has perfectly learned this CFG? In other words, if given the sentence, "Sally saw Alex with binoculars", what word(s) would attend to the word "Alex" the most in an ideal attention head? How does that relate back to the original CFG rules? The transformer is decoder-only (there is a causal mask). (5 points; N)

5. **Optional Extra Credit**: Briefly describe your implementation for handling longer context for BERT and create another table by expanding the table from Q1 and reporting the performance with longer context, side by side for each design. (10 points; A)

Submission Instructions

- Code: Place your code in a directory called `hw2_dir` and submit your code using the `turnin` command on the CS machine (`antor` or `data`):

  ```
  turnin -c cs577 -p hw2 hw2_dir
  ```

  Verify your submission using: `turnin -c cs577 -p hw2`.

- Discussion Questions: Submit a typed PDF of your answers on **Gradescope**.

- If you are unable to access your account on the cs machine (only) then please email your coding submission to one of the TAs.

References

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models, 2022. URL https://arxiv.org/abs/2210.11416.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song,

and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Mandar Joshi, Omer Levy, Luke Zettlemoyer, and Daniel Weld. BERT for coreference resolution: Baselines and analysis. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5803–5808, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1588. URL https://aclanthology.org/D19-1588/.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023. URL https://arxiv.org/abs/2205.11916.

OpenAI. Learning to reason with language models. https://openai.com/index/learning-to-reason-with-llms/, 2024. Accessed April 10, 2025.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.