CS4740/5740 Introduction to NLP
Fall 2019
Project 4. Story Cloze Test

Due via Gradescope, CMS, and Kaggle by Friday, December 6, 11:59 pm

1 Task

In this project, you will work with ROCStories [1], a dataset for evaluating the ability of computational approaches to story comprehension. The task of ROCStories is to choose the correct ending to a four-sentence story given two options for the ending. No human intervention is allowed in deriving answers. Table 1 shows samples of the dataset.

Four-Sentence Story	**Ending 1**	**Ending 2**
Karen was assigned a roommate her first year of college. Her roommate asked her to go to a nearby city for a concert. Karen agreed happily. The show was absolutely exhilarating.	**Karen became good friends with her roommate.**	Karen hated her roommate.
Jim got his first credit card in college. He didn't have a job so he bought everything on his card. After he graduated he amounted a $10,000 debt. Jim realized that he was foolish to spend so much money.	Jim decided to open another credit card.	**Jim decided to devise a plan for repayment.**
Gina misplaced her phone at her grandparents. It wasn't anywhere in the living room. She realized she was in the car before. She grabbed her dad's keys and ran outside.	**She found her phone in the car.**	She didn't want her phone anymore.

Table 1: Three instances of the ROCStories dataset. The input to the system is a four-sentence story (left column) and two ending options (middle and right columns), exactly one of which is correct. The system is expected to output the correct ending. Correct endings are in **bold**.

The goal of this project is two-fold: to gain experience in the implementation, evaluation, and analysis of a reading comprehension system based on what you have learned from the class (Part A), and to gain a bit of experience in working with an on-the-shelf, cutting edge natural language understanding framework (Part B).

1.1 Part A: Apply Your NLP Knowledge

In this part, you are to apply your NLP knowledge learned from the class to **b**uild a feature-based classifier for ROCStories. The features that you use are entirely up to you. As always, a good strategy is to work through lots of examples from the training data to determine what types of features would be most useful.

You are **NOT** allowed to use (i) deep contextualized pre-trained language models (e.g, ELMo, GPT, BERT) since they will be covered in Part B, or (ii) any existing code written for the ROCStories task. Otherwise, feel free to re-use any code you have written for previous projects in the class as well as **general** ML/NLP tools (e.g., PyTorch, spaCy).

1.2 Part B: Fine-tune a Pre-trained Language Model

In this part, the goal is to build a strong system by fine-tuning a pre-trained language model [2]. You can use **any** public code, **any** pre-trained language models, and **any** hardware resources — anything other than code specifically written for ROCStories.

If you have no experience in fine-tuning a pre-trained language model, we recommend that you build your system based on HuggingFace's Transformers[1] [3]. This site makes available the Transformer architecture described in class. You can find detailed guides on GitHub[1]. In particular, you can learn from its official examples for multiple-choice tasks[2]. Come to office hours if you encounter any difficulties.

Fine-tuning a pre-trained language model can be resource-demanding and time-consuming. However, since the dataset for our project is small, you should be able to finish running your experiment within several hours on your laptop CPU[3]. To further help speed up your experiment, we release along with the project BERT-mini[3], following the format of HuggingFace's Transformers[1]. Feel free to use it or not.

1.3 Data

We provide you with the following files.

- **train.csv** The training set.
- **dev.csv** The development set.
- **test.csv** The test set.
- **bert-mini.zip** A 3-layer pre-trained language model that can be used in HuggingFace's Transformers. To use it, unzip the folder `mini` from `bert-mini.zip`. Then, set `model_type` to `bert` and `model_name_or_path` to the path of `mini`.

2 What to Turn in

You should work in groups of 2–4 people. You must create your Group on both CMS and Gradescope. Be sure to re-create your Group with every resubmission to Gradescope.

- Code and Readme (CMS). Only include code that you write yourselves. DO NOT include pre-trained models (e.g., word embeddings, BERT-mini). The Readme should detail how to run your code.
- Report (Gradescope and CMS). Each part should be clearly labelled, and each page should be correctly assigned to the corresponding Gradescope rubric item.

[1]`https://github.com/huggingface/transformers`

[2]`https://github.com/huggingface/transformers/tree/master/examples#multiple-choice`

[3]We benchmarked fine-tuning BERT-base [4] and **BERT-mini** on this project for three epochs using a 2-core laptop CPU. BERT-base takes about 2 hours, and BERT-mini takes about 30 minutes.

- Prediction (Kaggle and CMS). Note that to be graded as a group, you must form groups with your teammate(s) on Kaggle. If you do not submit your prediction as a group, your Kaggle submission will be invalid. Kaggle links will be posted on Piazza.

3 Rubric

Since this is a new project, we may need to make changes. We will announce all changes in pinned posts on Piazza, and you are responsible for reading all of them.

- (60 pts) Part A
  - (10 pts) A description of your system. Enough detail should be provided so that, in theory, at least, we could re-implement it solely based on your description.
  - (5 pts) A justification of why you think the particular feature set you use would be effective.
  - (5 pts) A detailed illustration of extracted features for one training example (any one).
  - (10 pts) Implementation of three or more valid linguistic features.
  - (16 pts) Quantitative and qualitative analysis on the development set. Consider answering the following questions: How well does the system work? What features work/do not work? Any thoughts on further improvement?
  - (4 pts) Kaggle submission and screenshot of your best system's score.
  - (4 pts) The test set accuracy of your best system is higher than 54%.
  - (4 pts) The test set accuracy of your best system is higher than 62%.
  - (2 pts) The test set accuracy of your best system is higher than 70%.
- (34 pts) Part B
  - (5 pts) A detailed description of your implementation.
  - (5 pts) Plot the learning curve of your best system. The curve should include the training loss and development set accuracy by epoch.
  - (16 pts) Quantitative and qualitative analysis on the development set. In particular, include a comparison between this system and your system in Part A.
  - (4 pts) Kaggle submission and screenshot of your best system's score.
  - (4 pts) The test set accuracy of your best system is higher than 70%.
- (5 pts) Your report can have a maximum of 10 pages, with main body font size $\geq$ 10 pt. Write names and netIDs of every member in your team, and your Kaggle team name on the first page of your report. Include a section describing how each member contributes to the project. **You will lose all 5 pts if you do not follow any of the requirements.**
- (1 pt) Feedback for the project. Do you find the project too difficult/easy? How much time do you spend on it? Feel free to leave any suggestions if you would like to.

References

[1] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California, June 2016. Association for Computational Linguistics.

[2] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. In *https://openai.com/blog/language-unsupervised/*, 2018.

[3] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proeedings of the NAACL-HLT*, Minneapolis, MN, 2019.