CS-497: Deep Learning for Natural Language Processing
Final Projects

(50.0 points)

Winter 2022

Description: For the Final Project you will select between a Conversational Agent and a Commonsense Question Answering project. Each of these projects use established datasets, as described in the accompanying papers, and have leader boards that track current state-of-the-art results. Please feel free to review the papers used to generate leader board results for inspiration, but note that you are NOT expected to be competitive with these results, although it would be really great if you are!

For either project you will construct two baselines based upon naïve and simple models, respectively. Your task is to design and implement two approaches that you believe will beat each of these baselines. These approaches can be focused improved neural architectures or better ways to encode information relevant to training off-the-shelf models on these tasks Please note that code is provided to generate the baselines which can be easily modified to execute your own approaches.

Projects will be graded based upon clarity of presentation materials, the amount of thought and effort put into your approaches, and your analysis of results. The quality of your results (*e.g.,* did you beat the baselines?) will not be considered when grading, although results often correlate with the thought and effort put into your approaches.

Project A: Conversational Agent

The Conversational Agent project is based upon the MultiWOZ 2.2 dataset (for detailed descriptions see https://arxiv.org/pdf/1810.00278.pdf and https://aclanthology.org/2020.nlp4convai-1.13.pdf). This dataset contains multi-turn conversations between an agent and a user trying to accomplish a task in multiple domains (e.g., find a restaurant, book a hotel, etc.). Most turns in the conversation are annotated with specific values related to actions, user-intent, descriptive slot/value pairs and state-tracking, among others. Your task is to generate one or both sides of the dialogue to improve upon baselines -- how you do this is entirely up to you.

Code is provided to perform three basic versions of this task (see `woz.py`). Specifically, this code:

1. Processes the "dialogue" JSON files into _a, _b and _c datasets which are used to generate both sides of the dialogue using a (i) zero-shot model, (ii) a model fine-tuned on the dialogue language only, and (iii) a model fine-tuned on the dialogue and accompany structured data, respectively.
2. Uses GPT-2 as the generation model in all cases, and can be fine-tuned on the _b and _c datasets using the script included with the assignment (see `fine_tune_gpt2.sh`).
3. Performs text generation using (i) greedy decoding, (ii) top-p, or nucleus, decoding and (iii) beam search. An example using *logits* generated by the model is also presented in the event that you want to experiment with your own decoding algorithm. Please note that these *logits* are unexponentiated and were not put through a *softmax* to generate probabilities.
4. Evaluates the generated dialogue using BLEU score against the observed dialogue as the reference text.

Please note that this is NOT "starter code" and there is no requirement to use it. The code is provided as an example to get you up-and-running with the Huggingface codebase as simply as possible. You are welcome to use this code for the entirety of your assignment and focus you efforts on how to encode the data for model fine-tuning.

To get started, you will need to perform the following steps:

1. Install `transformers` from Huggingface (see https://huggingface.co/transformers/v2.0.0/installation.html).
2. Download the MultiWOZ 2.2 dataset from GitHub (see https://github.com/budzianowski/multiwoz/tree/master/data/MultiWOZ_2.2)
3. Create a "zero-shot" baseline by generating dialogue with a pre-trained language model.
4. Create a "language model" baseline by generating dialogue with a model fine-tuned on the dialogue only.
5. Download metrics from Huggingface (see https://huggingface.co/docs/datasets/using_metrics.html) and calculate the BLEU scores for each baseline.

To complete the assignment, you must implement two approaches that you believe will beat the "zero-shot" and "language model" baselines. Some approaches that you may want to consider include:

- Using a language model other than GPT-2 for generation (you will also need to create your baselines with this model).
- Using different decoding techniques (see https://huggingface.co/blog/how-to-generate).
- Implementing your own decoder to perform controlled/constrained generation.
- Using different metrics to evaluate performance (see https://huggingface.co/metrics).
- Conditioning generation on different parts of the dialogue.
- Encoding structured data in different formats and/or with different degrees of detail (*e.g.* encodings different from the _c dataset).
- Experimenting with different combinations of domains, or focusing on a single domain.
- Using the schema (see https://github.com/budzianowski/multiwoz/tree/master/data/MultiWOZ_2.2) provided with the dataset for further constrain generation or to implement a "hierarchical" decoding algorithm.
- Performing evaluation on classes of tokens instead of exact text (*e.g.* instead of evaluation on a specific time like `20:00`, use a `<time>` tag instead).
- Using dialogue from other datasets (*e.g.,* external training data) or structured external knowledge (*e.g.* ConceptNet, Freebase, etc. to identify concepts related to a domain).

Please note that this is not an exhaustive list, and that you are encouraged to come up with your own approaches.

What to turn in:

1. Proposals (10.0 points, due Friday March 4th): Outline two approaches (about one paragraph each) that you propose to use to accomplish this task. This deliverable is primarily for your benefit to avoid overly-simple or -complex approaches. Please submit as a single PDF file.
2. Preliminary Results (15.0 points, due Friday March 11th): Provide evaluation metrics for the "zero-shot", "language model" and "your approach(es)" using an evaluation metric of your choice. Include a brief description (about one paragraph per approach) describing each approach and an analysis of your results (about one paragraph per approach). A good analysis will include a hypothesis of which factors contributed to poor or strong results, and some form ablation analysis (*e.g.,* analyze performance by sub-classes -- are good results mostly driven by the "Good bye" part of the dialogue -or- a qualitative discussion of you highest/lowest scoring results). Please submit as a single PDF file.
3. Video Presentations (25.0 points, due Wednesday March 16th): Create a 10-to-15 minute video presentation with slides detailing (i) the dataset and task, (ii) your approaches, (iii) any implementation and/or encoding details which might be relevant, (iv) your results in tabular form, (v) a discussion of results, including any ablation analyses, and (vi) a discussion what kinds of other approaches may or may not be suited to this task. Please submit as a video files (e.g., MP4, etc.) and a PDF file of the presentation.

Project B: Commonsense Question Answering

The Commonsense Question Answering project is based upon the OpenBookQA dataset modeled on an "open book" middle-school science exam" format (see https://aclanthology.org/D18-1260.pdf). Multiple-choice questions are posed based upon one of 1326 facts. Your task is to improve upon baselines -- how you do this is entirely up to you.

Code is provided to perform two basic versions of this task (see `openbookqa.py`). Specifically, this code:

1. Processes the question-answer JSON files into _d and _e datasets which are used to answer the questions with a (i) question-answer only model and (ii) a model fine-tuned with the accompanying fact, respectively.
2. Uses BERT-base to perform multiple-choice question answering and is fine-tuned on both the _d and _e dataset using the Huggingface `Trainer()` module.
3. Displays selected question-answer pairs to confirm the data being used to train the models.
4. Evaluates the answers using an accuracy metric.

Please note that this is NOT "starter code" and there is no requirement to use it. The code is provided as an example to get you up-and-running with the Huggingface codebase as simply as possible. You are welcome to use this code for the entirety of your assignment and to focus you efforts on how to encode the data for model fine-tuning.

To get started, you will need to perform the following steps:

1. Install `transformers` and `datasets` from Huggingface (see https://huggingface.co/transformers/v2.0.0/installation.html).
2. Download the OpenBookQA dataset from GitHub using this script https://github.com/allenai/OpenBookQA/blob/main/scripts/download_and_prepare_data.sh
3. Create a "question-answer only" baseline by training the model on the _d dataset.
4. Create a "question-answer with fact" baseline by training the model on the _e dataset.

To complete the assignment, you must implement two approaches that you believe will beat the "question-answer only" and "question-answer with fact" baselines. Some approaches that you may want to consider include:

- Using a language model other than BERT for multiple-choice question answering (you will also need to create your baselines with this model).
- Using some explicit method to eliminate one of the answer choice and retraining your model to perform multiple-choice question-answering with three choices.
- Adding additional knowledge from the `crowdsourced-facts.txt` file included in the dataset.
- Using external knowledge from sources like ConceptNet, Freebase or Wikimedia, among others.
- Learning alignments between the tokens in the questions and multiple-choice answers.
- Identifying and using keywords in the question-answer pairs to decide which item(s) or additional and/or external knowledge to inject during fine-tuning.

- Experimenting with different methods of encoding additional and/or external knowledge (*e.g.,* logical triples, sentence, proto-sentences, etc.).
- Fine-turning a BERT model on a large set of additional and/or external knowledge before incrementally fine-tuning on multiple-choice question answering.
- Fine-tuning on an external set of question-answer pairs (*e.g.,* CommonSenseQA, etc.).

Please note that this is not an exhaustive list and that you are encouraged to come up with your own approaches.

What to turn in:

1. Proposals (10.0 points, due Friday March 4th): Outline two approaches (about one paragraph each) that you propose to use to accomplish this task. This deliverable is primarily for your benefit to avoid overly-simple or -complex approaches. Please submit as a single PDF file.
2. Preliminary Results (15.0 points, due Friday March 11th): Provide evaluation metrics for the "question-answer only", "question-answer with fact" and "your approach(es)". Include a brief description (about one paragraph per approach) describing each approach and an analysis of your results (about one paragraph per approach). A good analysis will include a hypothesis of which factors contributed to poor or strong results, and some form ablation analysis (*e.g.,* analyze performance by sub-classes of questions). Please submit as a single PDF file.
3. Video Presentations (25.0 points, due Wednesday March 16th): Create a 10-to-15 minute video presentation with slides detailing (i) the dataset and task, (ii) your approaches, (iii) any implementation and/or encoding details which might be relevant, (iv) you results in tabular form, (v) a discussion of results, including any ablation analyses, and (vi) a discussion what kinds of other approaches may or may not be suited to this task. Please submit as a video files (e.g., MP4, etc.) and a PDF file of the presentation.