11-411/11-611 Homework Assignment 4: Entity Recognition

NLP Teaching Staff

Due: December 5, 2024

1 Introduction

Your friend has just started pursuing a joint degree in paleontology and sociolinguistics at the prestigious Cranberry Melon University, and has embarked on a project analyzing how dinosaurs are named. They have noticed that many dinosaurs seem to be named after deities, seemingly in contrast with other types of animals. Unfortunately, there is no centralized database describing where dinosaur and other animal names come from, so they have begun the long and tedious process of *manually coding* or *annotating* a corpus of articles mentioning dinosaurs and other animals to identify where dinosaurs got their name, and in particular whether it was inspired by a deity. After this coding process is done, then they will be able to calculate corpus statistics such as the frequency at which dinosaurs are named after deities compared to other things or other animals, as well as more detailed analysis such as the belief systems and/or geographic origins of the dinosaurs and deities, and ultimately formulate a new understanding of paleontologist culture through the lens of the words and concepts they choose to associate with their subjects of study.

Inspired by what you've learned in NLP class, you tell them that you can use NLP and machine learning to save them tons of time on the annotation process by performing much of this task ✨automatically✨ using information extraction, so that they can spend more time on the fun and interesting task of actually analyzing the data, and the society it reflects.

In this homework, you will experiment with a variety of methods for the task of *entity recognition*, or assigning labels to spans of text. Specifically, you will be (1) building off HW3 to finetune a BERT-style model for recognizing relevant entities (mainly, dinosaurs and deities) using sequence labeling, and (2) comparing this approach to in-context learning using an LLM API. You'll have the opportunity to stretch your creative muscles to see how well you can get an LLM to perform on this task, motivated by the friendly competition of a Gradescope leaderboard.

2 Learning Objectives

- Using PyTorch and the Hugging Face Transformers library to finetune and evaluate pretrained decoder-only models for token labeling.
- Using an LLM API for zero-shot and few-shot in-context learning.
- Prompt engineering.
- Hands-on experience with some of the challenges of evaluating LLMs, particularly on information extraction tasks such as entity recognition.
- Designing a controlled ML experiment and clearly reporting experimental methodology and results.

3 Preliminaries

We've provided some starter code in the form of two Python notebooks: `HW4_BERT.ipynb` and `HW4_LLM.ipynb`.

`HW4_BERT.ipynb`: The BERT notebook is essentially the same as the notebook from HW3, and you will be tasked with modifying it to finetune BERT to perform *sequence labeling* rather than *sequence classification*.

`HW4_LLM.ipynb`: The LLM notebook provides a starter example for querying the OpenAI API to complete the same sequence labeling task. You will be responsible for implementing the functionality required to convert the data to and from LLM prompt formats.

As in the previous homework, finetuning BERT will require some compute cycles. You can run this on Google Colab, or on your machine (just using the CPU and no GPU), though the latter will likely be significantly/painfully slower. If you have access to an Apple Silicon MacBook Pro, you can also use its GPU, which runs about 3-4 times faster than the CPU but also slower than a Colab Nvidia GPU. Likewise, if you have a GPU on your PC laptop, you can likely use it for this assignment. However, you will be on your own with respect to support.

If Google Colab claims you have used too much GPU you can run on AWS SageMaker. You can also connect to a Jupyter notebook from an AWS EC2 instance. You can find support for doing this online. The tutorial here is a good place to get started, but is outdated in certain details.

4 Data: Dinos and Deities

Your friend has provided a small dataset of documents already annotated with labeled spans corresponding to dinosaurs and deities. More specifically, they have defined the following set of 6 entity types: `Aquatic_mammal`, `Deity`, `Cretaceous_dinosaur`,

From her ideological conception, Taweret$_{\text{Goddess}}$ was closely grouped with (and is often indistinguishable from) several other protective hippopotamus$_{\text{Aquatic_mammal}}$ goddesses.

Later, Paul Sereno proposed a new definition, the most inclusive clade containing Rhabdodon priscus$_{\text{Cretaceous_dinosaur}}$ but not Parasaurolophus walkeri$_{\text{Cretaceous_dinosaur}}$.

Figure 1: Some example sentences annotated with entities.

`Mythological_king`, `Goddess`, and `Aquatic_animal`. Figure 1 shows a couple example sentences annotated with those entities.

We have split the data into train, dev and test sets and already performed preprocessing to map the labels into BIO-encoded labels. The code for loading the data is included in the BERT notebook. The train and dev (validation) data are shared with you, the test data will be used to evaluate your code on Gradescope.

5 Task 1: Implementation (35%)

5.1 BERT (15%)

The starter code loads a pretrained BERT model, the training and development data, and fine-tunes the BERT model for sequence classification. **You are tasked with modifying the notebook to perform sequence labeling, so that it can predict a label for every token**, rather than just one label for the entire sequence. To clarify the BIO tagging scheme used for labeling entities: "B" (Begin) denotes the start of an entity, while "I" (Inside) indicates continuation within the same entity. For instance, "Tyrannosaurus" might be tagged as "B-Cretaceous_dinosaur" and "rex" as "I-Cretaceous_dinosaur."

Feel free to play around with the model a bit to optimize accuracy, but you should be able to get an F1 score of around 0.4 using the provided `seqeval` evaluation code using just a simple linear classifier head applied to the last layer of the BERT model, and the hyperparameters provided in the notebook. These are the criteria for getting full credit on this part. You should not replace the PyTorch training loop or model code with Hugging Face trainers.

5.2 LLM (20%)

For this homework, we'll be using the OpenAI API. Each student has been allocated $50 in OpenAI API credits, which should be more than enough to complete all tasks. Individual API keys have been shared through email.

The starter code includes functions to help you make requests to the OpenAI API for performing the sequence labeling task. Note that the OpenAI model may prepend "Labels:" to its output, so be prepared to handle this accordingly. Using the "vanilla" 5-shot setup provided in the assignment, you should be able to achieve an **F1 score greater than 0.25** on the dev set.

`HW4_LLM.ipynb` provides starter code that shows how a basic prompt can be constructed and sent to the model in order to obtain labels compatible with the entity recognition task. We refer to this as the "baseline" model, which uses a simple HTML-tag-style format to indicate entity labels on spans of text. **You are tasked with implementing the mapping from examples in the dataset to a zero- or few-shot prompt for the model, and for mapping back from the output of the model to a representation that can be used to evaluate** precision, recall and F1 using the same `seqeval` library we are using to evaluate the BERT model. To get the full 25% credit here, you will just need to implement the basic HTML-style tag format described in the starter notebook. When generating predictions, ensure that the number of predicted labels matches the reference labels in the test set. Mismatched counts may arise if extra tokens are predicted, so be mindful of token handling to avoid discrepancies. You will need to implement four functions in order to get this baseline running:

`get_chat_history`: This function is where you construct the prompt for the model, including any number of demonstration examples. It takes the number of shots, dataset, and a list of the relevant entity types, and returns the list of parameter maps to be passed through the `chat_history` parameter of the LLM API.

`get_message`: This function is where you construct the final message/prompt to pass to the model, i.e. where you provide the specific example for which you want the model to generate labels. It takes an example as input and returns the string to be passed through the `message` field to the LLM API.

`convert_response_to_bio`: This function enables evaluation by converting the response from the LLM into the same BIO format used to encode labels in the training and evaluation data. It takes an LLM response as input, and returns a list of labels corresponding to the entity label predictions indicated by the response.

`convert_bio_to_prompt`: This function performs conversion in the other direction: from an example which includes a list of tokens and their corresponding BIO labels to a plain-text format encoding the labeled spans. It takes an example as input and returns a string which represents the example using whatever format is being used to prompt the model.

See the comments and code in the notebook for more details.

6 Task 2: Experimentation & Prompt Engineering (65%)

Unlike previous homeworks, this homework will be a bit more open-ended to allow for you to exercise your creativity in the task of prompt engineering. **Your task is to experiment with different methods of constructing demonstrations and prompts the LLM to try to improve its performance on the task. While the goal is to improve F1, you will be graded not on your ability to get a high F1 score, but rather for the clarity with which you justify and describe the approach you chose, and present your experimental results.** As a starting point, you may find that a straightforward 5-shot prompt setup

with OpenAI yields an **F1 score of around 0.2–0.25** on the dev set. We recommend initially experimenting with simple prompt structures. You will describe your methodology, experiments and analysis in the report (see the next section for more details.) You are responsible for developing any code you need for instrumentation, data collection, analysis, and plotting.

All of your methods should work by modifying the same three functions as you implemented in Task 1 in order to get the baseline running: `get_chat_history`, `get_message`, `convert_response_to_bio`, and `convert_bio_to_prompt`. You are only allowed to use examples from the training data in order to prompt the model, but other than that you are free to modify anything about the prompt and response format, selection of demonstrations, etc. Please be reasonable and don't cheat!

You should experiment with a range of shots, both for your best/favorite model, as well as the baseline model from Task 1 (see the report write-up description for more details.) Your experimentation should not be random and haphazard, but should be well thought out and ideally justified with citations to previous work (which you will write up in your report. Towards this end, **you must propose and run your own experiment** exploring some aspect of LLM performance. While you may experiment with a wide variety of methods, your reported models and results should be focused on just one controlled experiment. For example, you might design a controlled experiment that elucidates the impact of different approaches for selecting demonstrations, or you might instead design an experiment that varies output formats in a controlled way.

7 Report write-up

You will write up a report consisting of three parts (and an appendix for code). You should think about this report as a mini research paper describing the LLM prompting experiments you ran. *Please make sure each part starts on a new page.*

Part 1: Methodology will succinctly describe your prompting experiments, and justify your decisions and experimental design. Supporting your choices with convincing reasoning is required. The most convincing reasoning will cite external evidence / previous results (citations to papers or tech reports are best; posts on Reddit, tweet threads, and similar are also acceptable). There should be sufficient detail provided that we could sit down and re-implement your approach, and we should be able to ascertain the necessary information without significant effort or reading a novel.

Part 2: Experimental Results will present the results of your approach (on the dev/validation set). This should include:

- A table of results that looks like Table 1.
- Two plots that show how performance changes as a function of number of few-shot examples (demonstrations), with shots on the x axis and precision/recall/F1 on the y axis, for the baseline model as well as your best-performing model.

Approach	Shots	Accuracy	Precision	Recall	F1
Finetune BERT	–				
Zero-shot LLM Baseline	0				
Few-shot LLM Baseline	1				
Few-shot LLM Baseline	5				
Few-shot LLM Baseline	10				
Few-shot LLM Baseline	20				
Few-shot LLM Baseline	100				
Your Prompting Approach 1					
Your Prompting Approach 2					
...					

Table 1: Example results table.

Part 3: Analysis and Discussion will present further analysis, observations and conclusions. We expect that this section will contain graphics, plots or data appropriate to support your analysis. There is no one correct answer to this part, and we have been intentionally vague so you can pursue your own questions and analysis. Your submission will be evaluated on the depth, breadth and **clarity** of your analysis and discussion. This includes both the written descriptions, as well as appropriate use of tables, figures and captions. All axes in figures must be clearly labeled, legends included, with legible fonts. Captions should include sufficient information to understand any plots/figures. If you are unsure about what this should look like, refer to published papers that have been linked in the slides throughout the course.

Appendix: Code will include any code you developed in order to produce the results and analysis in parts 2 and 3. *In order to receive full credit, you must comment this code clearly and concisely so that we can easily understand what you have done.*

Again, *please make sure that each part starts on a new page* to aid in grading.

8 Submission

Reports and code will be submitted and graded through Gradescope. More details on submission are available on Piazza.