SI 630 Homework 4: Prompt-based NLP

Due: Wednesday April 11, 1:00pm

1 Introduction

Training an NLP model can require lots of training data—we saw this in Homework 3 when we had to create our own data, and even that was a challenge! But what if we didn't need a lot of labeled data? One branch of NLP has focused on *few-shot learning* where a model is only given a few examples to learn from and then the model is asked to generalize to classify/label new examples. *If* it works, few-shot learning offers huge advantages since we no longer need to create massive datasets and can instead get by with a potentially much smaller set of examples. The central questions are then how to learn from a few examples and estimating how much can we learn from just a few examples.

In Homework 4, we'll work with one very recent approach few-shot learning: pattern-based learning. Here, we'll rely on large language models (LLMs) like BERT that already know a lot about word order and language to help learn from a few examples. Specifically, we'll take advantage of these models' abilities to fill in the blank. Consider the following sentence with a blank at the end:

> I loved it so much I bought three. I thought it was _____.

If you were to fill in that blank in the sentence above, you might say something like "great" or "amazing." Hopefully, the LLM might look at the earlier part of the sentence to fill in the blank with something similar too! In homework 4, we'll use patterns like this with this type of technique, called *prompt-based learning*, where we put in text that we want to classify and generate a *prompt* that a language model will fill in. The text that the language model fills in will tell us about the class. For example, say we were trying to do a sentiment task over movie reviews. Our prompt might look like

> [review text]. Overall, I thought it was [mask].

where we fill in the "[review text]" part with the text of the instance we're trying to classify and then look at what the model generates for the masked token position at [mask]. For these kinds of prompts, we'll write a *pattern* that lets us plug in the instance text want we want to label (the "[review text]" part above) and then specify a *masked* token that the LLM will fill in that is hopefully related to the label we want to know. These types of system aim to learn how to classify the instance based on what the model fills in. We will specify how to map some of what gets filled in using what's known as a *verbalizer*. In essence, we specify words that correspond to our labels. For example, if we were doing sentiment analysis we specify a verbalizer where the positive class gets mapped to words like "good" or "great" and the negative class to words like "bad" or "terrible." Your job as a practitioner is to figure out a few mappings. Your words don't have to be adjectives

either! You can write prompts that use verbs, nouns, or even adverbs to indicate the class of the text—be creative!

But what if we didn't even know how to create such a prompt—it might not be easy for some tasks! It might be nice if we could just explain the task and have the model do the rest like:

> The task is to classify a post the sentiment of a comment as positive or negative.
>
> Here are examples:
> Post: I loved it so much I bought three.
> Label: Positive
>
> Post: This was terrible.
> Label: Positive
>
> Label the following post:
> Post: some text here
> Label:

and then the language model just finishes the example with the correct label. This kind of idea is actually a reality with zero-shot learning! Here, we don't train the model at all. Instead, the model is provided with *instructions* on what to do and then the model produces the output label. Typically, these models are Causal Language Models (the left-to-right kind for English, such as GPT; in contrast to *Masked* Language Models like BERT that can fill in the middle). Often we call the instructions a *prompt*, much like in the prompt-based model; however, unlike promt-based learning, in zero-shot, we never train the model.

Zero-shot models can take many forms and most of the hard work is designing what kind of instructions to use.[1] ChatGPT and GPT3/4 are good examples of these kinds of models (in general) because they have been trained to produce answers for prompts from many kinds of data with instructions. In fact, this process is often called *instruction fine-tuning*, where instead of training a language model on general text, the later fine-tuning steps train the model to respond to prompts.

Prompt design can incorporate many kinds of design. In our example above, we have included two examples, one for positive and one for negative. What if we included more examples? What if we didn't include *any*? What if we used more structure in the prompt to include when and how to answer, like describing the two labels? What if we phrased it as a yes/no question for being positive sentiment? These are just a few and there are even blog posts describing common wisdom.[2] In answering such questions, you will move from the science of NLP to the art of NLP.

This assignment has the following learning goals:

- Familiarize you with the idea of few-shot and zero-shot learning and see how a model learns relative to how much data it is trained on.

[1] Many have speculated that prompt engineering will be a future branch of NLP, so check back in next year to hear if I've had to majorly overhaul the NLP course slides again.

[2] `https://zapier.com/blog/gpt-prompt/`

- Learn how to use a tokenizer for a large language model.
- Improve your NLP skills when working with cutting-edge code with examples.
- Become able to train a prompt-based model using limited data.
- Become able to train a zero-shot model using no training.

This last assignment is aimed at giving you one more skill in your arsenal for when you need an NLP classifier but don't have labeled data and can't create a large dataset of labeled examples.

2 Prompt-based and Few-shot Learning for Toxic Language

In Homework 4, we'll try using Jigsaw's Toxic Language dataset for identifying toxic language. The dataset is provided on canvas and **you should use the "toxic" column for training your classifiers**. Your development work has three parts: (1) train a prompt-based classifier using OpenPrompt and (2) write instructions for classifying toxicity using a zero-shot setup with Flan-T5, a very recent encoder-decoder model trained on instructions, and (3) a generic classifier just like you built in homework 3, for which you can use the same code.

For all three of classifiers, you'll evaluate them in different settings to see whether you can get the few-shot or zero-shot classifiers to match the performance.

2.1 Prompt-based Learning

For prompt-based learning, you will use the OpenPrompt library[3] to train the classifier. Your tasks will be to (1) write your own custom verbalizer and patterns and (2) train your model by modifying one of their example scripts. The OpenPrompt repository has good documentation on how to set up their model, train it, and use the code.

Like in Homework 3, in this assignment we will use a much smaller but nearly-as-performant version of BERT, `https://huggingface.co/microsoft/MiniLM-L12-H384-uncased`, as the base language model to train our pattern-filling model. While prompt-based learning can work on any LLM, MiniLM will make the homework much faster to finish.

One small hitch to writing a verbalizer is that they need to be a *single token* in the LLM's vocabulary. For word-piece based tokenerizers, this means you can't use phrases like "super awesome" or longer words that the tokenizer will break up. For example, the word "tokenize" is broken up into the tokens "token" and "##ize" (the ## part lets the model know the token is connected to the preceding one). While the prompt-based learning model can support multi-token words, using them requires more work and coding, which is not needed in this assignment (we want to keep it simple!). Since MiniLM is trained as a drop-in version of BERT, we can use BERT's tokenizer to check whether a word is entirely in the vocabulary. Note that you do *not* need to load in the BERT model to check this; instead, you can load in its pretrained tokenizer, which is available on HuggingFace.[4] Note that all the BERT-like models will use this tokenizer, so it's helpful to see how it works!

[3] `https://github.com/thunlp/OpenPrompt`

[4] `https://huggingface.co/docs/transformers/fast_tokenizers`

■ **Problem 1.** (10 points) Write a simple piece code that takes a single word as input and then tokenizes it with the BERT tokenizer in huggingface and returns the word's corresponding tokens (or token IDs) in the BERT vocabulary. You'll want to use this piece of code in the next task to check that your verbalizer is using only single-token words.

■ **Problem 2.** (20 points) Write 10 different prompts that can be used to classify toxic speech. Prompts should be relatively different (not just adding/changing one word). For each, come up with at least 2 verbalizations of each class (toxic/non-toxic). You can share verbalizations across prompts if needed. We *really* want to see some creativity across your prompts (this will also help the model learn more too).

2.2 Zero-shot Learning

For zero-shot learning, you should use the very recent Flan-T5 model Chung et al. [2022], which is available on HuggingFace.[5]. The full model is unlikely to fit in the GPU memory we have available (it's *big*) but we can probably use the flan-t5-small model.[6].

■ **Problem 3.** (20 points) Write 10 different zero-shot prompts that can be used to classify toxic speech. Prompts should be relatively different (not just adding/changing one word). We want you to try exploring the space of how to create a prompt. Some of the questions above can be used for inspiration, but there is a lot of room to explore here.

2.3 Regular Classifier

■ **Problem 4.** (10 points) For comparison with prompt-based learning models, train a regular classifier using `Trainer` and the MiniLM parameters on all the training data (very similar to what you did in Homework 3!). You should train your model for at least two epochs, but you're not required to do any hyperparameter tuning (you just need a score). Predict the toxicity of the provided test data and calculate the F1.

3 Comparison and Evaluation

■ **Problem 5.** (20 points) Using your patterns and verbalizers, train separate prompt-based learning models on 10, 50, 100, and 500 instances of data. Your data should be randomly sampled from the training data but **be sure to have examples of each class**. You are free to choose which instances you use and what distribution of toxic/non-toxic labels are in your training data (provided you have at least one example of each). For each model, predict the scores for the provided development data and calculate the Binary F1 (toxic is the positive class).

[5] `https://huggingface.co/docs/transformers/model_doc/flan-t5`
[6] `https://huggingface.co/google/flan-t5-small`

■ **Problem 6.** (10 points) Let's compare our two prompt-based models (the few-shot and zero-shot) and our regular all-data MiniLM model. Plot the score for each prompt-based learning model and your full-data MiniLM model using Seaborn. Use the training data size as the x-axis and Binary F1 as the y-axis. For the zero-shot models, you can draw these as different horizontal lines or points (since they don't have training data amounts) If you are feeling curious, feel free to train regular models on different sizes/distributions of data and include those too. Write your guess on how many instances you think you need to train a prompt-based learning model that will reach the performance of a MiniLM model trained on all the data.

■ **Problem 7.** (10 points) How good can we get our zero-shot model? Pick your favorite zero-shot prompt and predict the labels for the provided test data. Upload these predictions to the Kaggle competition posted about on Piazza. Scores within some reasonable range (e.g., not zero or near-zero) will receive full credit, though aim high.

4 What to submit

You should submit the following parts to Canvas.

- Your code for all parts
- A write-up showing your plot with models' scores and a sentence saying how many instances you think prompt needs to reach the performance of a MiniLM model trained on all the data.

5 Academic Honesty

Unless otherwise specified in an assignment all submitted work must be your own, original work. Any excerpts, statements, or phrases from the work of others must be clearly identified as a quotation, and a proper citation provided. Any violation of the University's policies on Academic and Professional Integrity may result in serious penalties, which might range from failing an assignment, to failing a course, to being expelled from the program. Violations of academic and professional integrity will be reported to Student Affairs. Consequences impacting assignment or course grades are determined by the faculty instructor; additional sanctions may be imposed.

Annotating in your group in a non-independent way is considered grounds for violation of Academic Integrity and will receive a zero for that part of the assignment.

References

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.