SI 630 Homework 4:
Prompt-based NLP

Due: Wednesday April 13, 5:30pm

Last updated: April 4 (version 1.0)

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 an approach called Pattern-Exploiting Training [PET; Schick and Schütze, 2020a,b], 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. The PET system aims 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!

This assignment has the following learning goals:

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

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 Learning for Toxic Language

In Homework 4, we’ll try using Jigsaw’s Toxic Language dataset using PET to train our classifier. Conveniently the PET authors have already provided code for you to use at `https://github.com/timoschick/pet`. Your task will be to (1) write your own custom verbalizer and patterns and (2) train your model by modifying one of their example scripts. The PET 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`, to train our models. While PET 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 PET 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.[1] Note that all the BERT-like models will use this tokenizer, so it’s helpful to see how it works!

■ **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.

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

■ **Problem 2.** (40 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).

■ **Problem 3.** (10 points) For comparison with PET, 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.

■ **Problem 4.** (30 points) Using your patterns and verbalizers, train separate PET 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 test data and calculate the Macro F1.

■ **Problem 5.** (10 points) Let's compare our PET-based models and our regular all-data MiniLM model. Plot the score for each PET model and your full-data MiniLM model using Seaborn. If you are feeling curious, feel free to train 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 PET model that will reach the performance of a MiniLM model trained on all the data.

3 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 PET needs to reach the performance of a MiniLM model trained on all the data.

4 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

Timo Schick and Hinrich Schütze. Exploiting cloze questions for few-shot text classification and natural language inference. *Computing Research Repository*, arXiv:2001.07676, 2020a. URL http://arxiv.org/abs/2001.07676.

Timo Schick and Hinrich Schütze. It's not just size that matters: Small language models are also few-shot learners. *Computing Research Repository*, arXiv:2009.07118, 2020b. URL http://arxiv.org/abs/2009.07118.