SI 630 Homework 4: Generation and Detection via Transformers

Due: Friday April 23, 11:59pm

Introduction

Pretrained language models like BERT and GPT have revolutionized many areas of NLP. These models have been trained on massive volumes of text and their parameters reflect a basic understanding of the structure and meaning of many kinds of text. The huge advantage for such pre-training is that models can be quickly adapted to new tasks by performing fine-tuning, where the parameters are modified to reflect a particular type of text (e.g., social media) and to use them to perform some specific task (e.g., classification). As a result, with just a limited amount of data, these models are capable of generalizing to a much wider set of instances; for example, a BERT-based classifier trained on a few thousand classifiers might perform substantially better than a logistic regression classifier trained on the same data (like we saw in the demo notebook in class!). Homework 4 is designed to get you experience with these kinds of models.

In the long long ago of Lecture 2, we learned about language models that could generate text and briefly played around with the Talk to Transformer website,[1] which can generate stories from the start of one. Wouldn't it be cool to try building and using that kind of technology yourself? In this assignment, you'll finally get face to face with some of the latest NLP deep learning libraries to do two tasks:

1. Generate the lyrics to a song
2. Detect whether a song you've been given was generated by a machine or not.

As a part of this assignment, you'll work with one of the many libraries built on top of the HuggingFace transformers library (or work with transformers itself), which has implementations of many of the latest-and-greatest NLP models and, conveniently, has included example scripts for how to effectively fine-tune pretrained models to do both of these tasks.

The overarching goal of Homework 4 is to generate the most realistic song lyrics you can from the generation model. However, you're also tasked with detecting these machine-generated lyrics, which creates a dilemma—you want your generation system to fool your classification system

[1] https://talktotransformer.com/

(making it think the lyrics are human-authored), yet you also want the classification system to do well.[2]

Most of the homework effort will consist of familiarizing yourself with how to run these models and will likely not involve writing much code (unless you want to go wild on creating cool generators/classifiers, which I fully support).

Homework 4 has the following (very practical) learning goals:

1. Learn how to use modern deep learning libraries like transformers to fine-tune pretrained models like BERT (for classification) and GPT (for generation)
2. Learn how to run your code on a GPU (as provided by Great Lakes)
3. Get a sense of training time and complexity for working with these models
4. Try your hand at tuning hyperparameters to make better models, especially as it relates to convergence

In summary, we want to help you get familiar with the basics of these models in a way that gives you skills you might use on a job, a course project, or in a research setting (e.g., fine-tuning a BERT classifier for your particular domain).

Part 1: Learning What Lyrics Look Like

Pre-trained language models are trained on a variety of text and taken off the shelf, won't generate "lyric-like text" on their own. As a result, we first need to fine-tune a language model to have it learn the structure and vocabulary of song lyrics.

For Part 1, you'll fine-tune one of the transformers from the HuggingFace library to generate text. I personally recommend the openai-gpt or gpt2 model, as they're small enough to reliably fit on the GPUs given to you on the Great Lakes clusters. Note that in deep learning terminology, for training your language generation system in Part 2, you'll want to fine-tune what are known as *causal language models* (CLMs) which are the language models we talked about in class on Week 2. GPT and GPT2 are examples of CLMs. These models capture the left-to-right ordering of words, e.g., $p(w_i|w_{i-1})$, and are trained to predict from only the prior context. CLMs are in contrast to *masked language models* (MLMs) like BERT which condition on the whole context to predict missing words.

You can use either the `transformers` library with one of its training scripts or the `simpletransformers` library and its wrappers. Both are easy to use in practice and will mostly require you to learn how to call the training functions using the provided scripts or skeleton code.

[2] We'll talk about this concept much more in class as a part of a special kind of network called a Generative Adversarial Network (GAN) but you are not implementing that here.

We have provided three large datasets for you to use on Piazza. These are real lyrics (see the copyright disclaimer below) and come with the artist and genre(s) that the artist is known for. The data is divided into train, dev, and test. You should use the training data to fine-tune your model. **You do not need to use all of the training data**. Given the time constraints (both end-of-semester and Great Lakes runtimes), you are welcome to use subsets of the data, provided that you can justify your decision and have enough data to train the model. For example, using the first 100K lyrics is fine, as is using all lyrics from a particular genre (as long as it is not rare); however, using all lyrics from a single artist is insufficient as there are likely too few examples. If you have some concern, please let us know.

The lyric data is provided to you in a JSON format, which preserves the line breaks. However, most (if not all) of the language models will read their input data as one instance per line. Since you want to generate entire songs and not *lines* of a song, you'll need to encode the lyric as a single line. There are various ways of doing this: one is to add a special token to the vocabulary to represent a newline and include this in your model; another is to simply use a regular text word to indicate a newline. You are free to use any approach for training. Recognize that your model will *generate* lyrics in this format in Part 2, which you'll then need to convert back to the actual new line.

When fine-tuning your model, you'll need to choose hyperparameters that converge to a low loss. To figure out the right hyperparameters, you should use the dev data to select the best hyperparameters that give the lowest perplexity (reminder: perplexity is the metric we discussed back in Lecture 2 that measure how surprised the model is by the words that come next; a lower perplexity indicates the language model expects to be generating this kind of word, which is what we want).

If you want to get fancy (very optional!), your model can also incorporate more structure too by including meta-data on the input. Part of the *art* of making these models is figuring out how to encode it. The encoding is like training a language model to predict $p(w_i \mid w_{i-1}$, genre=country, artist="Taylor Swift"). One way to do this is to include this kind of information as input to the model, either by using special tokens for artist/genre or even just putting it as text the model learns to use.

Important Note: Some of the lyrics in the data we provide you may contain offensive language as written by the original songwriter. The lyrics' use in the course is for instructional purposes in generating language and the instructional team does not condone their message or intent. As a part of generating language later, you may find that your model learns to reproduce such language, which is one of the potential harms addressed way back in Week 2 with the Stochastic Parrots paper. You are welcome to filter out such messages from your fine-tuning data (though the underlying language model may still be biased towards generating them to begin with!).

What to do:

- Convert the train, dev, and test data to single-line input format (one song per line)

- Train a model using the training data and save it to an output directory. You can/should train multiple models and evaluate them using dev data to pick the model with the lowest perplexity on the dev set.
- Report the following
    a. The loss on the training for your best model and its perplexity (on train.txt)
    b. the perplexity on dev data (remember you have to process it too in the same way)
    c. once you have finalized the model, the perplexity on test tata

Important Copyright Disclaimer

The lyrics contained within the provided data are the property of the original author and are provided here solely for educational purposes within the context of the SI 630 class, in accordance with the fair use exemption for education of the copyright act. Redistribution or use of the data for purposes other than for purposes of this class is strictly forbidden. This data is provided at present scale in order to facilitate the training of deep learning models for Homework 4.

Part 2: Generating Lyrics

Once you have your lyric-generating model fine-tuned (which is the hard part), now it's time to generate lyrics. You can again use `transformer`'s scripts for this or `simpletransformer`'s code (or whatever library you used to fine-tune if it was different). For both libraries, you'll need to provide a prompt to start the generation process. You should use prompts from the initial word of a song from your data. E.g., for a lyric starting "I saw the", you would begin the song with "I". You are welcome to use any of the initial words to generate your lyrics (though you need variety).

Note: if you decided to add more context to the lyrics (e.g., the genre) during fine-tuning, be sure to filter these out when generating your lyrics.

What to do:

- Using your final model and random seed 2021, generate lyrics for the following 5 prompts:
    a. My
    b. The
    c. One
    d. When
    e. If
- Generate 500 new lyrics using a sample of the initial word of songs from your data. E.g., for a lyric starting "I saw the", you would begin the song with "I". You are welcome to use any of the initial words to generate your lyrics (though you need variety).
- Use your model to generate lyrics (or pick from the 500) and pick your "favorite" to post in the Piazza post called "Put your favorite song lyrics here." Please pick something relatively clean, if possible

Since the model is originally pre-trained on lots of language, feel free to try out new words (not in training) or write longer song prompts.

Note that for the generation, you'll eventually want to convert the output of the generation back to a human-readable format that looks like lyrics (e.g., with no special newline characters).

Part 3: Training a classifier to recognize machine-generated text

Part 3 will develop a classifier to recognize machine-generated text. You'll once again use the `transformers` or `simpletransformers` library to train a model`—this time a classifier. For Part 3, you'll want to use a Masked Language Model (MLM) for classification (unlike Parts 1 and 2 which used a CLM). I personally recommend the `distilbert-base-cased` model, as it's small enough to reliably fit on the GPUs given to you on the Great Lakes clusters. This particular model is a pared-down version of BERT with fewer parameters, however you're welcomed to try out different models!

Both the `transformers` and `simpletransformers` libraries provide scripts and code to train these kinds of classification models.

In particular, your task will involve several steps:

1. Create a dataset of machine-generated vs. human-authored lyrics from the data we give you.
2. Convert the lyrics to a single-line-per-lyric format
3. (Optionally) fine-tune your classification model like you did for language generation
4. Train a classifier on the single-line-per-lyric data

For training data, we've provided 10K examples of machine generated lyrics for train, and 5K each for dev and test. (More data may potentially be available if needed). You will need to write some code to create the full training, dev, and test data from these machine-generated examples and the human-authored lyrics we've given you. You'll assign the classification label of each lyric . This step is the majority of the code you'll need to write for this part of the homework.

If you want to go wild, you can use your lyric generation from Step 2 to generate *more* machine-generated training data.

What to include in your write-up:

- Describe in a few sentences how you constructed your train, dev, and test data (e.g., which kinds of lyrics did you use? How did you decide on how many to use?)
- Report the following

    a. The accuracy on train.txt for your best model
    b. the accuracy on dev.txt
    c. once you have finalized the classification model,
        - the accuracy on test.txt
        - the accuracy on the 500 lyrics you have generated from Part 2, i.e., does the model predict your machine-generated lyrics as human or machine?
- Once finished scoring the model, describe what changes you might make to the lyric generation part to improve its ability to generate human-looking lyrics.

What to submit

1. A report with
    a. Answers to all the questions (and all the models' performances)
    b. The 5 generated lyrics that had specific prompts we asked for
2. A json file with your 500 generated lyrics (these should be in the "normal" format with newlines)
3. All the code for your generation and classification systems (including any preprocessing)