ECE 1786 Lecture #9

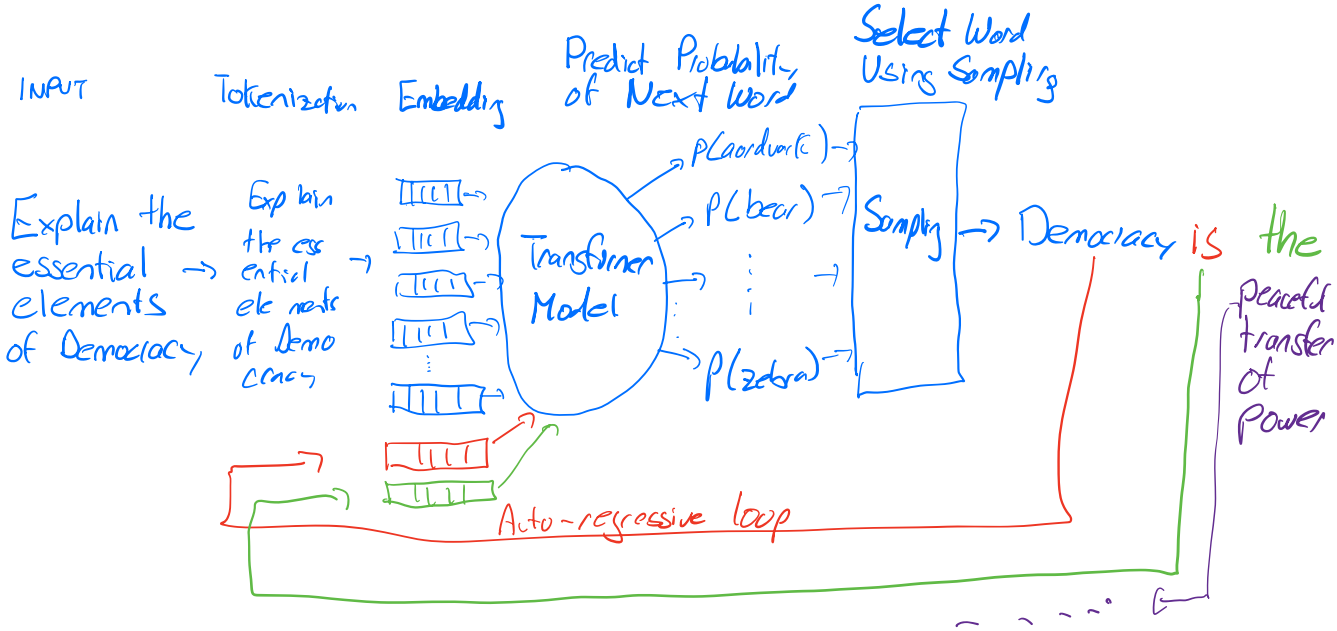
Work-in-Flight:

- Assignment 4 - Decoding for Generation, Prompt Engineering for Different Tasks; due November 15
- Project interim report - due Monday November 20 at 9pm
- Git Hub Request for access: only 25/36 teams have asked/fill form!!!

Last Day: Project Proposals! (Lecture 8++)
Before That: Prompt Engineering (Lecture 7)

Today: How the Large Language Models Became Good at doing what you ask: Reinforcement Learning from Human Feedback (RLHF)

In lectures 4,5 and 6 we covered the architecture of a transformer, the process of training it, and then using it for text generation. Lecture 7 discussed how to prompt a transformer generator that has been well-trained. When you put it all together it has some key moving parts:



I did leave out one thing about that training that is important: What makes the LLMs good at doing what you ask them to do? I've implied that simply training a model to predict the next word is all that it needs to become so smart, but that isn't quite all that is needed; there is a second layer of training.

That said, I do believe (but don't have proof) that much of the higher-level comprehension apparent in the models does come from this first level training.

However, if you were to use a model that is only trained the first way, you would find that its output would be messy and unsatisfactory in many ways - poor output (random characters and repetition and off-topic answers) would appear more often, and the model's ability to comprehend what you wanted would be worse than you've experienced with chatGPT and GPT-4.

- I'll post an example query that shows this of what every other model looked like this time last year compared to GPT-3.5, which was what became chatGPT (maybe show)

OpenAI led the way on this second layer of training.They call it "Reinforcement Learning with Human Feedback." (RLHF)

- It is possible to describe the essentials of what is going on first without using the structure of Reinforcement Learning (RL), although there is a part of it that needs specific RL techniques.
- The core of the second layer RLHF is described in a paper by Ouyang et. Al, (https://arxiv.org/abs/2203.02155 also posted in the course notes) but also described in a more readable, higher-level form in a Huggingface blog post (https://huggingface.co/blog/rlhf that is also posted). The Llama2 paper from Meta also speaks to some interesting issues around RLHF and safety vs. quality the Llama2 training paper (https://arxiv.org/abs/2307.09288, also posted).

Before we begin, let's define a little terminology: A prompt is the text that is input to the large model. A completion is the full sequence of text produced in the auto-regressive loop - i.e. the sequence of tokens that become words.

9-24.

Cohere - Toronto start up

co:here

Dashboard Documentation Playground Community

Generate Embed Classify

Export code Share

PRESETS

EXAMPLE PRESETS

Content Creation
- Blog Posts
- Email Copy
- Hashtag Generator
- Product Descriptions

Summarization
- Chat Summarization
- Article Summarization

Paraphrasing
- Spelling & Grammar Check
- Correct Errors in Voice to Text Tran...

Information Extraction
- Extract Entities from Legal Agreem...
- Extract Entities from Invoices

Suggestions for project ideas in a course called Creative Applications of Natural Language Processing: **Using the Classroom as an Experimental Lab.**

Bibliography

Books

Journal articles

See also

Computational linguistics
Corpus linguistics
Computational statistics
Corpus-based computational linguistics
Data mining
Dialectology
Digital humanities
Digital literacy

Language documentation
Lexicography
Lexical semantics
Linguistic geography
Linguistic typology
Metaphor
Natural language processing
NLP: Using the Classroom as an Experimental Lab

Phonetics
Pragmatics
Psycholinguistics
Semantics
Sociolinguistics
Speech recognition
Syntax
Text analytics
Text corpus
Text mining

Notes

Further reading

PARAMETERS

MODEL: xlarge-20220609 (xlarge)

NUMBER OF TOKENS: 160

TEMPERATURE: 0.7

STOP SEQUENCES: -- ×

TOP-K: 0

TOP-P: 1

FREQUENCY PENALTY: 0

PRESENCE PENALTY: 0

CONTROL PANEL

Generate

Clear all

Save

9-26

huggingface.co/bigscience/bloom?text=Suggestions+for+project+ideas+in+a++

PapernotRSC · MiBot · Learn Python in Y Minutes · Bazen Papers · LutherQinDocs · NFREF · ECE 1786 · ECE 243 · 2022/3 Undegrad Capstone · Massey Senior-Junior · ECE Iris · Learning · TCAIREM · CBL

Hugging Face | Search models, datasets, users... | Models | Datasets | Spaces | Docs | Solutions | Pricing

Hugging Face is way more fun with friends and colleagues! Join an organization | Dismiss this message

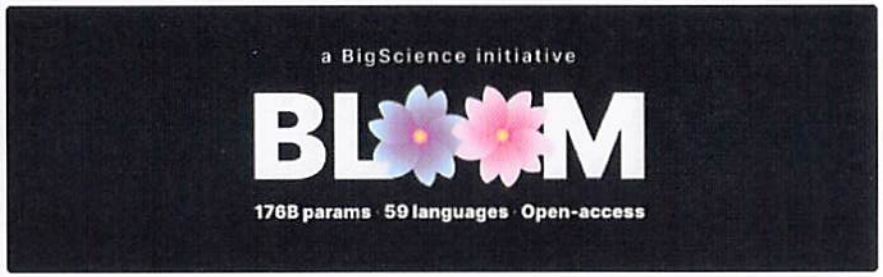
bigscience / **bloom** ♡ like 1.68k

Text Generation · PyTorch · TensorBoard · Transformers · 46 languages · arxiv:1909.08053 · arxiv:2110.02861 · arxiv:2108.12409 · doi:10.57967/hf/0003 · bloom

feature-extraction · Eval Results · License: bigscience-bloom-rail-1.0

Model card | Files and versions | Training metrics | Community 138 | Deploy | Use in Transformers

Edit model card



BigScience Large Open-science Open-access Multilingual Language Model
Version 1.3 / 6 July 2022

Current Checkpoint: **Training Iteration 95000**

Total seen tokens: **366B**

Model Details

BLOOM is an autoregressive Large Language Model (LLM), trained to continue text from a prompt on vast amounts of text data using industrial-scale computational resources. As such, it is able to output coherent text in 46 languages and 13 programming languages that is hardly distinguishable from text written by humans. BLOOM can also be instructed to perform text tasks it hasn't been explicitly trained for, by casting them

Downloads last month
12,512

Hosted inference API

Text Generation · Groups · Examples

Suggestions for project ideas in a course called Creative Applications of Natural Language Processing: "find a good phrase/sentence and create a book about it".

A:

Look at

sampling | greedy | BLOOM prompting tips

Switch to "greedy" for more accurate completion e.g. math/history/translations (but which may be repetitive/less inventive)

Compute ⌘+Enter 0.9

Powered by AzureML | The model is loaded and running on Intel Xeon Ice Lake CPU

JSON Output | Maximize

Spaces using bigscience/bloom 53

huggingface/bloom_demo

ysharma/Talk_to_Multilingual_AI_WhisperBloomCoqui

Meta/facebook

92c

Free, Unlimited OPT-175B Text Generation

Warning: This model might generate something offensive. No safety measures are in place as a free service.

W Fact | Chatbot | Airport Code | Translation | Cryptocurrency | Code | Math

Suggestions for project ideas in a course called Creative Applications of Natural Language Processing:

Generate

Suggestions for project ideas in a course called Creative Applications of Natural Language Processing: - Sentiment analysis of reddit comments - Machine translation of reddit comments - Sentiment classification of reddit comments I think I'll go with the last one.

> Sentiment classification of reddit comments I think you're going to need a lot of data for that.

I'm sure there's a lot of data on reddit.

Like the results? ☆ Support Alpa development by staring Alpa on GitHub

Screenshot | Tweet it! #alpa

9-24

beta.openai.com/playground

PapernotRSC MiBot Learn Python in Y Minutes Bazen Papers LutherQinDocs NFREF ECE 1786 ECE 243 2022/3 Undegrad Capstone Massey Senior-Junior ECE Iris Learning TCAIREM CBL

Overview Documentation Examples Playground

Help University of Toronto

Get started

Enter an instruction or select a preset, and watch the API respond with a completion that attempts to match the context or pattern you provided.

You can control which model completes your request by changing the model.

KEEP IN MIND

- Use good judgment when sharing outputs, and attribute them to your name or company. Learn more.
- Requests submitted to our models may be used to train and improve future models. Learn more.
- Our default models' training data cuts off in 2021, so they may not have knowledge of current events.

Playground

Load a preset... Save View code Share ...

Suggestions for project ideas in a course called Creative Applications of Natural Language Processing:

1. Develop a part-of-speech tagger for a low-resource language.

2. Build a machine translation system for a low-resource language pair.

3. Develop a question answering system for a low-resource language.

4. Create a chatbot for a low-resource language.

5. Develop a text summarization system for a low-resource language.

Mode

Model: text-davinci-002

Temperature: 0.7

Maximum length: 256

Stop sequences
Enter sequence and press Tab

Top P: 1

Frequency penalty: 0

Presence penalty: 0

Best of: 1

Inject start text

Inject restart text

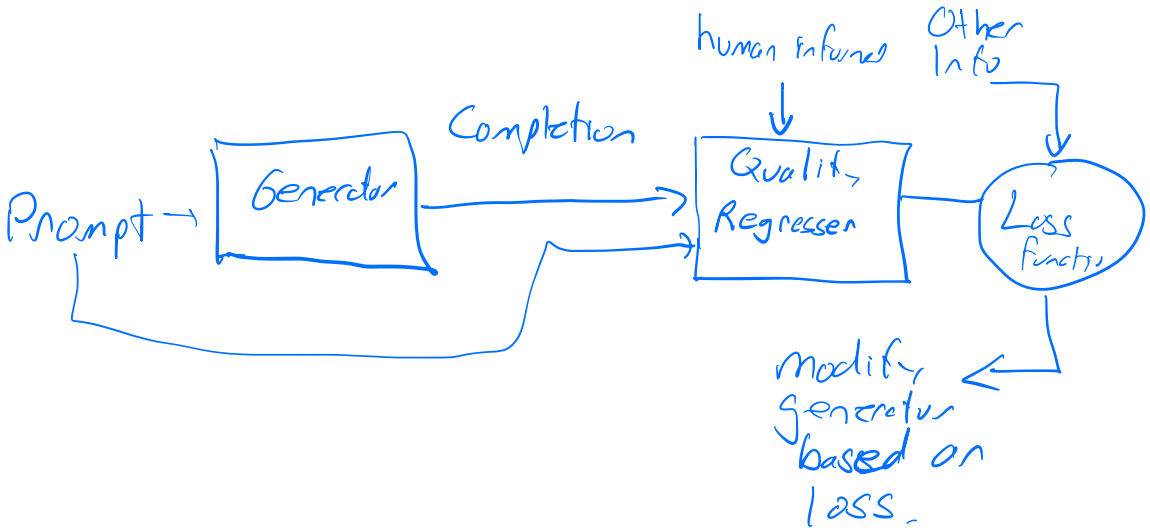
Show probabilities: Off

Submit

17

Here is a summary of that 'layer 2' method used to make the models better, after the initial, very extensive pre-training on very large corpora to predict the next word:

1. Pay humans to create a dataset of prompts + completions (questions/answer; or requests/responses) by humans, which do a good job of answering/ responding to well formed questions/requests. Pay people to do this, and train them well (Meta/llama2 says this quality really matters)
2. Fine tune the model, predicting the next word, on these specific examples
   A. Works some, but isn't the core method
3. Build and train a classifier (really regresser) that produces a rating of the quality of the prompt+completion.
4. Use the output of the classifier to compute a loss function that judges the completion given the prompt. It is used as a reward function in a reinforcement learning optimization of the generator. That is, this loss (plus another) is used to adjust the parameters in the Transformer so that the probabilities generated ultimately make better 'completions' that humans prefer.



The details are interesting, with some complexity. One of the most clever parts is that once you've got the classifier and this system going, it is possible to automatically generate many more data examples (of prompts/completions) to train on.

Step 1: Hire Humans to create good examples of prompt+completions. E.g, from Ouyang paper, both prompt & "labeler demonstration" were human-produced:

Prompt:
Create a shopping list from this recipe:
Trim the ends off zucchini. Cut zucchini in half lengthwise; scoop out pulp, leaving 1/2-in. shells. Finely chop pulp. In a skillet, cook beef, zucchini pulp, onion, mushrooms and peppers over medium heat until meat is no longer pink; drain. Remove from the heat. Add 1/2 cup cheese, ketchup, salt and pepper; mix well. Spoon into the zucchini shells. Place in a greased 13x9-in. baking dish. Sprinkle with remaining cheese.

on screen

human

Labeler demonstration
ucchini, beef, onion, mushroom, peppers, cheese, ketchup, salt, pepper

Jumping ahead & reiterating why this step is important, we can see what the non-RLHF'd "GPT3 175B" does with the prompt, and what the the RLHF'd one (called "instruct GPT" back then). The first is useless, the second answers the question.

GPT-3 175B completion:
Bake, uncovered, at 350° for 20-25 minutes or until zucchini is tender and cheese is melted.

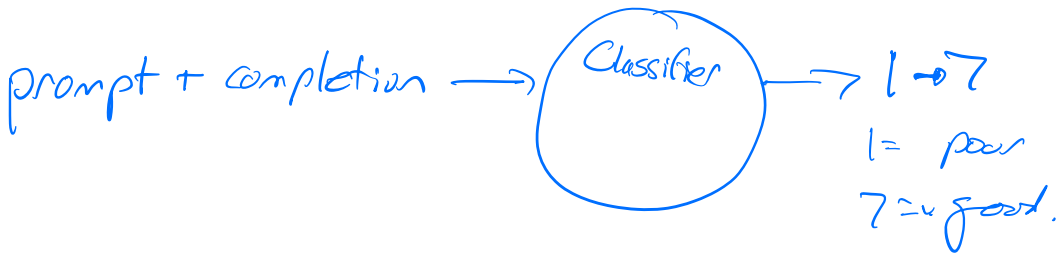
InstructGPT 175B completion:
Bake, uncovered, at 350° for 20-25 minutes.

- 8 zucchini squash, ends trimmed
- 1 pound ground beef
- 1 cup finely chopped zucchini pulp
- 1 small onion, chopped
- 8 ounces diced mushrooms
- 1 sweet red or green pepper, diced
- 1/2 cup grated cheese
- 1/4 cup ketchup
- Salt and pepper to taste
- 1 cup shredded cheese

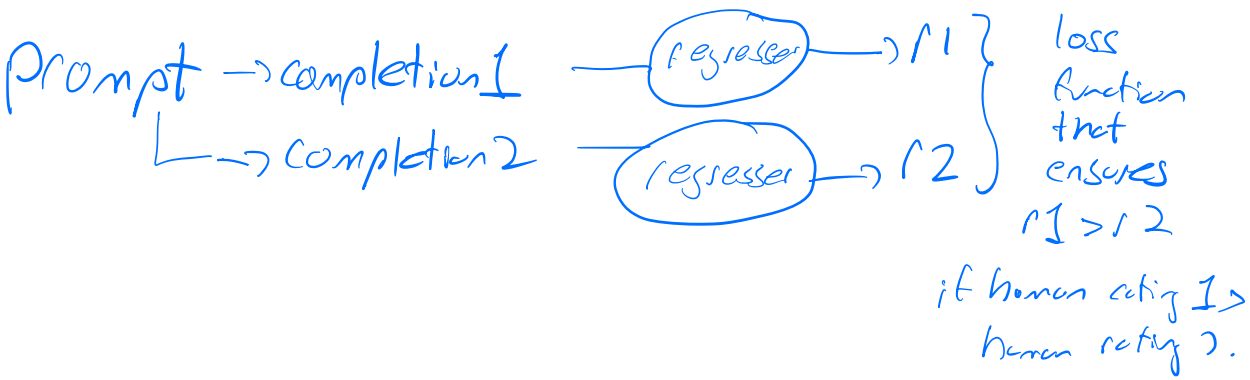
Figure 49: Labeler-written prompt from our dataset, along with the human-written demonstration, and completions from GPT-3 175B and InstructGPT175B. Prompt is lightly cherry-picked (5 selected from 15 to show a diverse range of tasks), and the completions are not cherry-picked.

See the Ouyang paper for many examples of this, in the appendices.
A total of 12,000 prompt completions were paid for back then
At first, these were used to do regular 'fine-tuning' - they were used as training input to predict-the-next-word, as you're familiar with from Lecture 5/6.
– call this "SFT" for Supervised Fine Tuning

Step 2: Train a classifier/regresser model that takes in a prompt+completion as input and produces a rating of the quality of the completion given the prompt. The rating is on a scale of 1-7.



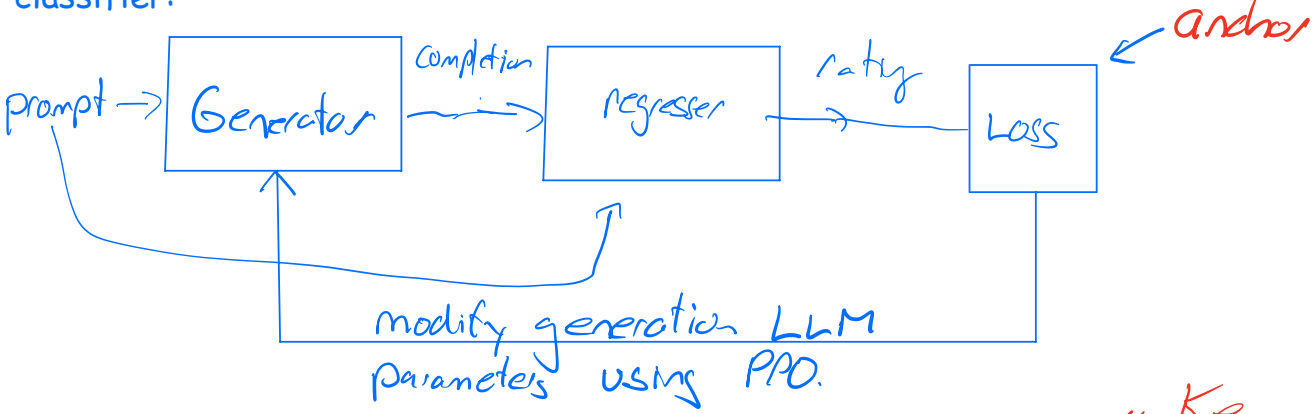
- To do this, collect human labels on a set of prompts + completions - again paying people
  - issue: humans are calibrated differently, one person's 1, is another's 5 (*)
  - So, the regresser/classifier is trained in a more clever way - present two completions for the same given prompt; aside: what is a regresser?
  - Ask humans to label both completions with on scale 1-7
  - The goal of the training is to make the model's rating such that it prefers the ones that humans prefer. It is not to make the ratings match, because of (*).



- Notice that we can get as many completions as we want, by simply re-running the generator. That's how we get completion1 and completion2 (**)
- Aside: what sort of model should we use to make the regresser?
  - Answer - a Transformer, perhaps even the same model as the LLM being, with the language head is replaced with a single regression output
    - Open AI used small GPT; Meta used largest LLAMA2.

9-6

Step 3: Create a larger dataset of prompts + completion1, 2, 3 .... N (i.e. many completions for a given prompt, using (**). Use those as training data for a new round of different training of the generator, that makes use of the step 2 classifier:



- Can generate many completions, as said, so lots of potential data
- Confession - this "PPO-RL modification of the generator is too complex for me to understand and explain

Key

What is "anchor"? It turns out that without a key addition to the loss function, this feedback/loss will cause the generator to lose its basic knowledge.

- The regresser human preference goal will override the basic knowledge
- The generator output logits/probabilities need to be 'anchored' to that knowledge, while they are also being 'pulled' by this loss to create a good quality human preferred output

Loss = f(Classifier(prompt+completion), model outputs of Unchanged SFT)

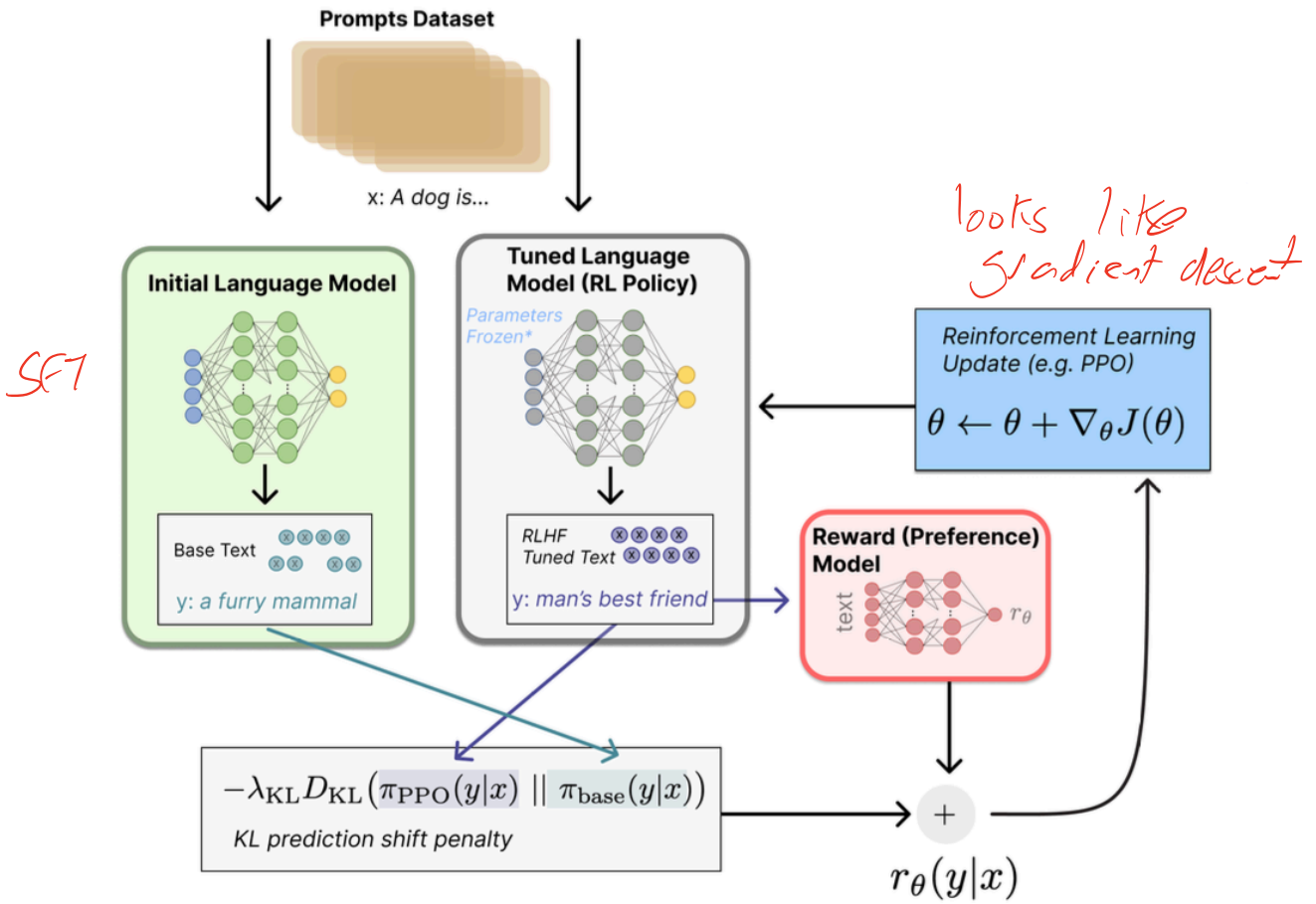
- The unchanged SFT is the model after step 1.
- Take all the logits of SFT, and the logits and of the generator, and force them to stay reasonably close.

"Supervised Fine Tuning"

The picture below, from the Huggingface blog, gives some sense of the whole process. It makes use of terminology from Reinforcement Learning that I have not used.

- "Policy" = the generator language model being tuned;
- "Reward Model" = the Regresser that was trained to figure out how good a prompt+completion is

9-7



looks like gradient descent

SFT

ECE 1786 Project Progress Report

- Due on November 20
- Hard word limit - 1000, with penalty
- Goal is to make sure you're pushing for progress!
- See the assignment on line for the deeper description, also attached below

ECE 1786 Project Progress Report

The project progress report is a check-in to show that you are on track to complete your project. By the project progress date, you should have made good progress on:

- Collecting data if that was a part of your plan
- Producing a baseline model or have good clarity and done the work necessary to make a comparison that shows your approach is working or not
- Producing at least one result, including one qualitative or quantitative comparison
- Reflected upon the feedback given at proposal time

The report document demonstrates your progress. The document has a word limit of maximum of 1000 words, which is a hard limit as usual.

Some of the sections are similar to your project proposal. You may find that when you look at your previous writing of that proposal a second time, that you find ways of expressing your ideas more concisely.

The word limit is hard: There is a 1% penalty for every word more than the 1000 limit. Please count the words in your document, compute the penalty, and put it on the front page. These characters/words are not included in the word count, nor are pictures or references.

There is a penalty-free grace period of one hour past the deadline. Any work that is submitted between 1 hour and 24 hours past the deadline will receive a 20% grade deduction. No other late work is accepted.

The progress report should have the following sections:

Introduction

- Give a clear (re)statement of the goal of your project, making use of the feedback you received on the proposal, and any adjustments you've made since the proposal.

Data Processing

- Describe the data that you have collected and cleaned to date. Be clear and specific when describing what you've done, so that a classmate can reproduce your work. If at all possible, show some statistics about your cleaned data (e.g. number of examples in each class), and at least one example of a cleaned training data. Since no plan ever survives first contact with reality, this section will probably be different from what you wrote in your proposal.

- Note that class 2 projects may not need as much data as class 1 projects. However, there is almost always a need for some kind of hold-out test set, and your work in that direction should be described here.

Baseline Model or Comparison Method

- Briefly describe (again) your baseline model or comparison method that you created to compare with your neural network. This may have evolved from your proposal, so indicate what has changed if anything. The essence of this section is to show that you've progressed in having a way to determine how well you are succeeding, so far.

Architecture

- Give a new, better description of the architecture that you plan to build, and how far along you are in creating it. This description should be more detailed than in your initial proposal, and in many cases, hopefully much better as described in the feedback. There should be clarity on how information/data flows throughout the model.

Result

- At least one result or comparison between your system and either the baseline or using the comparison method described above. You are not measured on how well it is working at this point, just if you're able to make a sensible comparison. For some problems. Quantitative measures are preferred, but if you can make a case for a qualitative comparison, that's okay too.

Discussion

- Discuss your results, including at least one set of training curves if applicable, or otherwise use some other metrics. Do you think your system is performing well? Base your discussion on both the results that you have shown, and the interpretation of your training curve. What issues, particular to your project, will you have to overcome?

Team Work and Progress

- Describe how well your team is working together. Take a look at the divided tasks and deadlines you set earlier. How is each person doing? What has each person accomplished?