Practical prompt engineering

CS 780/880 Natural Language Processing Lecture 23

Samuel Carton, University of New Hampshire

Last lecture

Key idea: Download **pretrained** transformer model, then **fine-tune** to particular task

Pretrained transformer models

- BERT, RoBERTa, XLNet, RoBERTa, DistilBERT, T5, GPT-X

Encoder-decoder, encoder-only, decoder-only

How to choose?

Looking forward:

- Zero- and few-shot learning
- Prompt engineering

Lecture 17 prompt engineering

High-level introduction covering common problems with LLMs and common solutions

- No code, no practical details

Various problems:

- Hallucination
- Reasoning errors
- Ungrounded outputs
  - Problem for robotics
- Boring output
- Biased outputs

Various approaches:

- Retrieval-augmented generation
- Atlas
- Recitation-augmented generation
- Chain of thought prompting
- Faithful chain of thought prompting
- Do as I Can, not as I Say
- ReAct

Today: practical prompt engineering

How to actually construct prompts and use proprietary APIs

- Connecting to the OpenAI API
- Zero vs few-shot learning
- Exemplar choice
- Prompt design

Training a (very) large language model

Four distinct steps in training a model like GPT3.5/4, GEMINI, Claude, etc.

1. Pick architecture
2. Language modeling
3. Instruction tuning
4. Reinforcement learning from human feedback (RLHF)

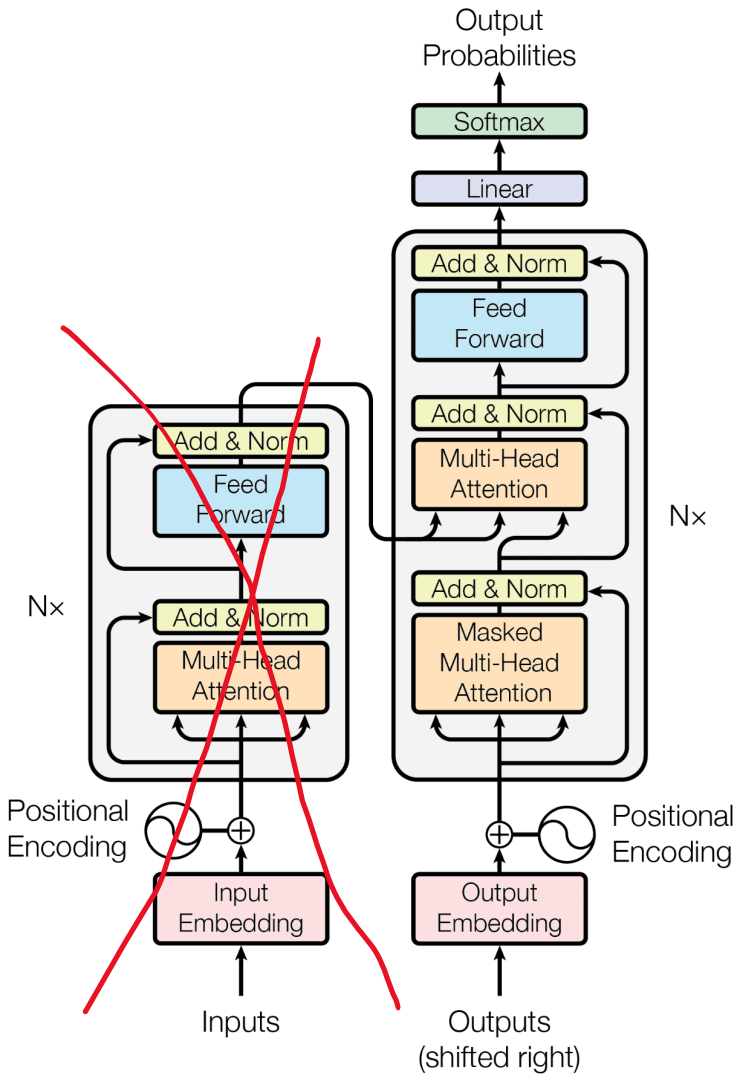
Architecture

Contemporary models mostly **decoder-only**

- Pretty much just deep stacks of transformer self-attention layers

Lots of parameters

- BERT, GPT-2: 110 million
- GPT-3: 175 billion
- GPT-4: 1 trillion (?)



Language modeling

1. Choose **very** large corpus of text, code, etc.
    - Main companies very tight-lipped about exact pretraining dataset
    - Often use some variant of the Common Crawl (https://commoncrawl.org/)
        - Contents of most of the internet

2. Apply standard language modeling objective: $L(S) = -\sum_i \log P(s_i | s_0, \ldots, s_{i-1}; \theta)$
- I.e. teacher forcing

Instruction tuning

Basic idea: Take trained language model and fine-tune it on relatively small dataset of (instruction, response) pairs

- Bridges the gap between learning the language and learning how to take instructions

Examples:

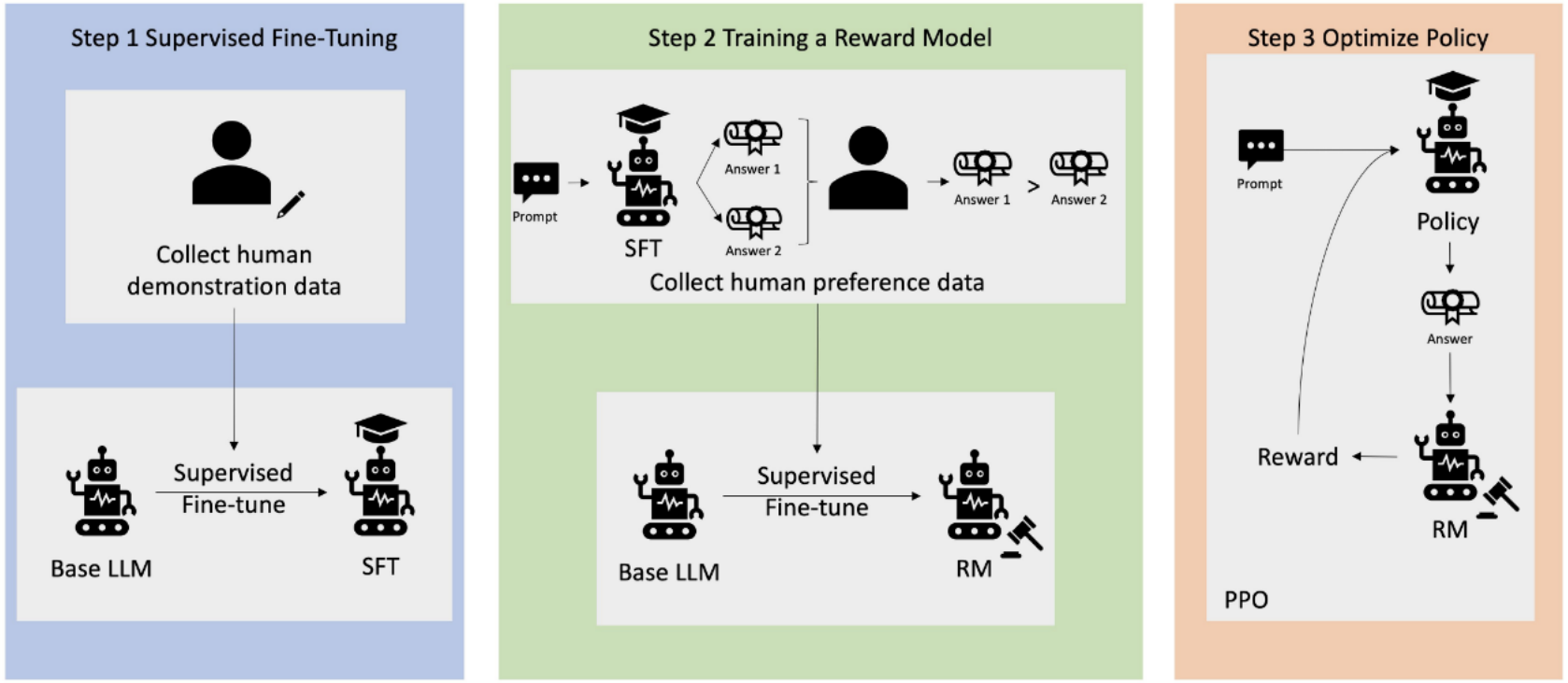
- Natural Instructions: 193k instances derived from 61 existing NLP datasets
  - Mishra, Swaroop, et al. "Cross-task generalization via natural language crowdsourcing instructions." *arXiv preprint arXiv:2104.08773* (2021).
- Flan 2021: 15M examples of 1836 tasks derived from 62 existing NLP datasets (including SST02!)
  - Longpre, Shayne, et al. "The flan collection: Designing data and methods for effective instruction tuning." *International Conference on Machine Learning*. PMLR, 2023.

For more details: Zhang, Shengyu, et al. "Instruction tuning for large language models: A survey." *arXiv preprint arXiv:2308.10792* (2023).

Reinforcement learning from human feedback

Basic idea: Have model generate multiple possible responses to a prompt, have humans label which one they prefer, use RL to encourage model to respond like that in the future.

- Uses **reinforcement learning**, an alternative learning paradigm to supervised learning



https://aws.amazon.com/what-is/reinforcement-learning-from-human-feedback/

Final result

Once we do all these steps, we end up with a model that:

- Is very good at sounding right
- Can do lots of tasks without any actual training
- Only “knows” things that it happened to memorize from language modeling objective
  - E.g. “New Hampshire’s nickname is The _____ State”
  - Best at stuff with lots of support in the training data
- Has been “encouraged” via RLHF to not be racist or teach people how to make bombs
  - But this is a pretty soft guardrail

In summary: very (surprisingly) powerful, but also very flawed, unreliable, and breakable

Practical prompt engineering

There’s an increasing proliferation of LLMs available out there, for use with either graphical interfaces or APIs

- OpenAI/GPT-X: https://platform.openai.com/
- Google/GEMINI: https://ai.google.dev/
- Anthropic/Claude: https://www.anthropic.com/api

Some free, some not. OpenAI API very much NOT free. But it’s the most popular and the one I am familiar with, so that’s what I’ll teach.

Other APIs will be conceptually similar.

SST-2

The last time we'll see it! Say a tearful goodbye.

Doing two things different from usual:

- Truncating to 100 items
- Converting label to text

```
1  display(dev_df)
```

	index	sentence	label
0	797	it 's not original , and , robbed of the eleme...	negative
1	198	this is a winning ensemble comedy that shows c...	positive
2	769	an infectious cultural fable with a tasty bala...	positive
3	472	you will emerge with a clearer view of how the...	positive
4	572	the film 's tone and pacing are off almost fro...	negative
...	...	...	...
95	355	there is no pleasure in watching a child suffe...	negative
96	233	i 'd have to say the star and director are the...	negative
97	849	trademark american triteness and simplicity ar...	positive
98	438	the movie has an infectious exuberance that wi...	positive
99	92	you wo n't like roger , but you will quickly r...	negative

100 rows × 3 columns

BERT baseline

Just as a basis for comparison, we can look at how a BERT-based classifier does on our 100-item validation set sample:

```
bert_trainer.fit(model=bert_model,
                 train_dataloaders=train_dataloader,
                 val_dataloaders=dev_dataloader)
```

```
INFO:pytorch_lightning.utilities.rank_zero:GPU available: True (cuda), used: True
INFO:pytorch_lightning.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO:pytorch_lightning.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO:pytorch_lightning.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO:pytorch_lightning.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:pytorch_lightning.callbacks.model_summary:
  | Name           | Type               | Params
-------------------------------------------------
0 | bert           | BertModel          | 109 M
1 | output_layer   | Linear             | 1.5 K
2 | train_accuracy | MulticlassAccuracy | 0
3 | val_accuracy   | MulticlassAccuracy | 0
-------------------------------------------------
109 M     Trainable params
0         Non-trainable params
109 M     Total params
437.935   Total estimated model params size (MB)
Validation accuracy: tensor(0.6100, device='cuda:0')
Epoch 0: 100%
Validation accuracy: tensor(0.9400, device='cuda:0')
Validation accuracy: tensor(0.9500, device='cuda:0')
Validation accuracy: tensor(0.9500, device='cuda:0')
Validation accuracy: tensor(0.9500, device='cuda:0')
Validation accuracy: tensor(0.9500, device='cuda:0')
INFO:pytorch_lightning.utilities.rank_zero:`Trainer.fit` stopped: `max_epochs=1` reached.
Training accuracy: tensor(0.9179, device='cuda:0')
```

Installing OpenAI

```
# The OpenAI API keeps changing, so I'm teaching a slightly older version

!pip install openai==1.3.7

# You can find up-to-date documentation at https://platform.openai.com/docs/introduction
```

```
Collecting openai==1.3.7
  Downloading openai-1.3.7-py3-none-any.whl (221 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 221.4/221.4 kB 4.9 MB/s eta 0:00:00
Requirement already satisfied: anyio<4,>=3.5.0 in /usr/local/lib/python3.10/dist-packages (from openai==1.3.7) (3.7.1)
Requirement already satisfied: distro<2,>=1.7.0 in /usr/lib/python3/dist-packages (from openai==1.3.7) (1.7.0)
Collecting httpx<1,>=0.23.0 (from openai==1.3.7)
  Downloading httpx-0.27.0-py3-none-any.whl (75 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 75.6/75.6 kB 8.0 MB/s eta 0:00:00
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from openai==1.3.7) (2.7.0)
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (from openai==1.3.7) (1.3.1)
Requirement already satisfied: tqdm>4 in /usr/local/lib/python3.10/dist-packages (from openai==1.3.7) (4.66.2)
Requirement already satisfied: typing-extensions<5,>=4.5 in /usr/local/lib/python3.10/dist-packages (from openai==1.3.7) (4.11.0)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages (from anyio<4,>=3.5.0->openai==1.3.7) (3.7)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio<4,>=3.5.0->openai==1.3.7) (1.2.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.23.0->openai==1.3.7) (2024.2.2)
Collecting httpcore==1.* (from httpx<1,>=0.23.0->openai==1.3.7)
  Downloading httpcore-1.0.5-py3-none-any.whl (77 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 77.9/77.9 kB 9.3 MB/s eta 0:00:00
Collecting h11<0.15,>=0.13 (from httpcore==1.*->httpx<1,>=0.23.0->openai==1.3.7)
  Downloading h11-0.14.0-py3-none-any.whl (58 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.3/58.3 kB 6.6 MB/s eta 0:00:00
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1.9.0->openai==1.3.7) (0.6.0)
Requirement already satisfied: pydantic-core==2.18.1 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1.9.0->openai==1.3.7) (2.18.1)
Installing collected packages: h11, httpcore, httpx, openai
Successfully installed h11-0.14.0 httpcore-1.0.5 httpx-0.27.0 openai-1.3.7
```

Authenticating with OpenAI

```
# I'm going to have to delete my credentials before I make this notebook public
# because it costs me a little money every time I want to use them.

# Sign up for your own OpenAI account here: https://platform.openai.com/signup

import os
import openai
organization = None
api_key = None

client = OpenAI(
    # defaults to os.environ.get("OPENAI_API_KEY")
    api_key=api_key,
)
```

Zero-shot prompt

The simplest possible prompt structure consists of: 1) instructions; 2) input document; and 3) space for generated output

```python
# We begin with a function that will take in an SST-2 instance and create a prompt from it

def create_zero_shot_prompt(text:str):
  prompt = f'''
Label the following short movie review as either positive or negative.

Review: {text}

Label:
  '''.strip()

  return prompt

first_dev_prompt = create_zero_shot_prompt(dev_df.iloc[0]['sentence'])
print(first_dev_prompt)
```

```
Label the following short movie review as either positive or negative.

Review: it 's not original , and , robbed of the element of surprise , it does n't have any huge laughs in its story of irresponsible cops who love to play pranks .

Label:
```

Calling the OpenAI API

```python
# Then we make a function to send the prompt to the OpenAI API and get the response back
from openai import OpenAI

def prompt_gpt_4(prompt:str):
  chat_completion = client.chat.completions.create(
    messages=[
        {
            "role": "user",
            "content": prompt,
        }
    ],
    model="gpt-4",
  )
  return chat_completion.choices[0].message.content
```

```python
first_dev_response_1 = prompt_gpt_4(first_dev_prompt)
print(first_dev_response_1)
```

```
Negative
```

```python
first_dev_response_2 = prompt_gpt_4(first_dev_prompt)
print(first_dev_response_2)
```

```
Negative
```

```python
first_dev_response_3 = prompt_gpt_4(first_dev_prompt)
print(first_dev_response_3)
```

```
Negative
```

```json
{
    "id": "chatcmpl-abc123",
    "object": "chat.completion",
    "created": 1677858242,
    "model": "gpt-3.5-turbo-0613",
    "usage": {
        "prompt_tokens": 13,
        "completion_tokens": 7,
        "total_tokens": 20
    },
    "choices": [
        {
            "message": {
                "role": "assistant",
                "content": "\n\nThis is a test!"
            },
            "logprobs": null,
            "finish_reason": "stop",
            "index": 0
        }
    ]
}
```

https://platform.openai.com/docs/api-reference/making-requests

Running 0-shot learning on our data

```python
# Then we simply iterate through the items in our (shortened) dev set, and run
# the API on each one

zero_shot_responses  = []

for index, row in dev_df.iterrows(): #one of several ways to iterate through rows of a DF
  if index % (dev_df.shape[0]//10)==1: print(index,'...')
  prompt = create_zero_shot_prompt(row['sentence'])
  response = prompt_gpt_4(prompt)
  zero_shot_responses.append(response)
```

```
1 ...
11 ...
21 ...
31 ...
41 ...
51 ...
61 ...
71 ...
81 ...
91 ...
```

```python
zero_shot_responses
```

```
['Negative',
 'Positive',
 'Positive',
 'Positive',
 'Positive',
 'Negative',
 'Positive',
 'Positive',
 'Negative',
 'Positive',
 'Positive',
 'Negative',
 'Negative',
 'Positive',
 'Positive',
 'Positive',
 'Negative',
 'Negative',
 'Negative',
 'Positive',
 'Negative',
 'Negative',
 'Positive',
 'Positive',
 'Neutral']
```

Running 0-shot learning on our data

```
dev_df['zero_shot_prediction'] = zero_shot_responses
# GPT-4 capitalized all its responses so we need to lower-case them
dev_df['zero_shot_prediction'] = dev_df['zero_shot_prediction'].apply(lambda s:s.lower())
display(dev_df)
```

	index	sentence	label	zero_shot_prediction
0	797	it 's not original , and , robbed of the eleme...	negative	negative
1	198	this is a winning ensemble comedy that shows c...	positive	positive
2	769	an infectious cultural fable with a tasty bala...	positive	positive
3	472	you will emerge with a clearer view of how the...	positive	positive
4	572	the film 's tone and pacing are off almost fro...	negative	negative
...	...	...	...	...
95	355	there is no pleasure in watching a child suffe...	negative	negative
96	233	i 'd have to say the star and director are the...	negative	negative
97	849	trademark american triteness and simplicity ar...	positive	positive
98	438	the movie has an infectious exuberance that wi...	positive	positive
99	92	you wo n't like roger , but you will quickly r...	negative	neutral

100 rows × 4 columns

Calculating 0-shot performance

```
# And now we can use scikit-learn functions to assess the accuracy

from sklearn.metrics import accuracy_score
print(accuracy_score(dev_df['label'], dev_df['zero_shot_prediction']))

# And it's good!
```

```
0.96
```

```
# We can take a look at the few (4) cases where the model disagreed with the true label
pd.options.display.max_colwidth=0
dev_df[dev_df['label'] != dev_df['zero_shot_prediction']]

# And they are all pretty borderline.
```

	index	sentence	label	zero_shot_prediction
20	260	/ but daphne , you 're too buff / fred thinks he 's tough / and velma - wow , you 've lost weight !	negative	positive
51	102	does paint some memorable images ... , but makhmalbaf keeps her distance from the characters	positive	negative
91	143	a solid film ... but more conscientious than it is truly stirring .	positive	negative
99	92	you wo n't like roger , but you will quickly recognize him .	negative	neutral

Few-shot prompt

```
# What we mostly need to do few-shot learning is a new function that will take in
# an SST item and create a prompt with X exemplars in it

# First we make a function that will create the text of an exemplar
def create_exemplar_text(text:str, label:str):
  exemplar = f'''
Review: {text}

Label: {label}
'''.strip()
  return exemplar


first_exemplar = create_exemplar_text(train_df.iloc[0]['sentence'], train_df.iloc[0]['label'])
print(first_exemplar)
```

```
Review: hide new secretions from the parental units

Label: negative
```

Few-shot prompt

```
# Then a function that will sample exemplars from a dataframe and add them into an overall prompt
def create_few_shot_prompt(text:str,
                           num_exemplars:int,
                           random_seed:int,
                           exemplar_df:pd.DataFrame):
  # Construct the prompt piece by piece
  prompt = 'Label the following short movie review as either positive or negative.\n'

  # Add text of each exemplar
  exemplar_rows = exemplar_df.sample(n=num_exemplars, random_state=random_seed)
  for index, row in exemplar_rows.iterrows():
    exemplar_text = create_exemplar_text(row['sentence'], row['label'])
    prompt = prompt + '\n'+exemplar_text+'\n'

  # Add final prompt
  prompt =prompt + f'''
Review: {text}

Label:
  '''
  return prompt

first_dev_few_prompt = create_few_shot_prompt(dev_df.iloc[0]['sentence'],
                                              3,
                                              0,
                                              train_df)
print(first_dev_few_prompt)
```

Few-shot prompt

An archetypal few-shot prompt looks something like the following:

1. Instructions
2. Exemplars
3. Final prompt (should have identical format to exemplars)

```
Label the following short movie review as either positive or negative.

Review: very pleasing at its best moments

Label: positive

Review: , american chai is enough to make you put away the guitar , sell the amp , and apply to medical school .

Label: negative

Review: too much like an infomercial for ram dass 's latest book aimed at the boomer

Label: negative

Review: it 's not original , and , robbed of the element of surprise , it does n't have any huge laughs in its story of irresponsible cops who love to play pranks .

Label:
```

Running 1-shot learning on our data

```
one_shot_responses = []
for index, row in dev_df.iterrows():
  if index % (dev_df.shape[0]//10)==1: print(index,'...')
  prompt = create_few_shot_prompt(row['sentence'],
                                  num_exemplars=1,
                                  random_seed = index+5646,
                                  exemplar_df=train_df)
  response = prompt_gpt_4(prompt)
  one_shot_responses.append(response)
```

```
1 ...
11 ...
21 ...
31 ...
41 ...
51 ...
61 ...
71 ...
81 ...
91 ...
```

```
one_shot_responses
```

```
['negative',
 'positive',
 'positive',
 'negative',
 'negative',
 'negative',
 'positive',
 'positive',
 'positive',
 'negative',
 'positive',
 'positive',
 'negative',
 'positive',
 'positive',
 'negative',
 'negative',
 'positive',
 'positive',
 'positive',
 'positive',
 'negative',
 'negative',
 'positive',
 'negative',
 'negative',
 'positive',
 'positive',
 'positive']
```

Running 1-shot learning on our data

```
pd.options.display.max_colwidth = 50
dev_df['one_shot_prediction'] = one_shot_responses
# Note that because we demonstrated the output format, we don't need to lowercase the predicted labels
display(dev_df)
```

	index	sentence	label	zero_shot_prediction	one_shot_prediction
0	797	it 's not original , and , robbed of the eleme...	negative	negative	negative
1	198	this is a winning ensemble comedy that shows c...	positive	positive	positive
2	769	an infectious cultural fable with a tasty bala...	positive	positive	positive
3	472	you will emerge with a clearer view of how the...	positive	positive	negative
4	572	the film 's tone and pacing are off almost fro...	negative	negative	negative
...	...	...	...	...	...
95	355	there is no pleasure in watching a child suffe...	negative	negative	negative
96	233	i 'd have to say the star and director are the...	negative	negative	negative
97	849	trademark american triteness and simplicity ar...	positive	positive	positive
98	438	the movie has an infectious exuberance that wi...	positive	positive	positive
99	92	you wo n't like roger , but you will quickly r...	negative	neutral	positive

100 rows × 5 columns

Calculating 1-shot performance

```
# And again we can calculate the accuracy
print(accuracy_score(dev_df['label'], dev_df['one_shot_prediction']))

# And it's equally good as zero-shot
```

```
0.96
```

```
# And it gets mostly the same ones wrong as zero-shot, except for the first row
pd.options.display.max_colwidth=0
display(dev_df[dev_df['label'] != dev_df['one_shot_prediction']])
```

	index	sentence	label	zero_shot_prediction	one_shot_prediction
3	472	you will emerge with a clearer view of how the gears of justice grind on and the death report comes to share airtime alongside the farm report .	positive	positive	negative
20	260	/ but daphne , you 're too buff / fred thinks he 's tough / and velma - wow , you 've lost weight !	negative	positive	positive
51	102	does paint some memorable images ... , but makhmalbaf keeps her distance from the characters	positive	negative	negative
99	92	you wo n't like roger , but you will quickly recognize him .	negative	neutral	positive

Calculating 5-shot performance

```
# Marginally better
print(accuracy_score(dev_df['label'], dev_df['five_shot_prediction']))

```

```
0.97
```

```
pd.options.display.max_colwidth=0
display(dev_df[dev_df['label'] != dev_df['five_shot_prediction']])
```

	index	sentence	label	zero_shot_prediction	one_shot_prediction	five_shot_prediction
20	260	/ but daphne , you 're too buff / fred thinks he 's tough / and velma - wow , you 've lost weight !	negative	positive	positive	positive
51	102	does paint some memorable images ... , but makhmalbaf keeps her distance from the characters	positive	negative	negative	negative
99	92	you wo n't like roger , but you will quickly recognize him .	negative	neutral	positive	positive

Evaluating LLMs

As we can see from these high accuracy scores, we've essentially exhausted SST-2 as a useful benchmark for model accuracy.

- Only "errors" we're seeing are truly borderline cases where humans would disagree
- All subjective tasks (e.g. sentiment, moderation) have some of these, meaning accuracy ceiling is never truly 100%

So how do we come up with more meaningful evaluations of LLMs?

- More challenging tasks
- Main topic of next lecture

Advanced prompting methods

In lecture 17 I went over a few advanced prompting techniques that have been introduced in the literature, including:

- **Chain-of-thought prompting:** improve logical reasoning
- **Retrieval-augmented generation:** reduce hallucination
- **Recitation-augmented generation:** reduce hallucination without doing actual retrieval

And so on. Lots of things people are experimenting with these days. See the following for more details:

Liu, Pengfei, et al. "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing." *ACM Computing Surveys* 55.9 (2023): 1-35.

Role-setting

Some people swear by “role-setting”, where before you give it any instructions, you give the model an “identity” to abide by

- This is how people have been “jailbreaking” the models.

Example from my submitted paper:

Role Setting:

I am a helpful assistant capable of extracting information from text. I will not generate any new tokens... return the information in format of the schema.

Exemplar:

{{Prompt instructions}}
{{Exemplar XML}}
{{Exemplar output JSON}}

Prompt:

Extract all information relating to every High Entropy Alloys (HEA) from the following text: {{Paper XML}} using the following schema: {{JSON schema}}. Return a list of schemas in JSON format for every unique compound.

Additional context: HEA composition sometimes have variable ratio... not available then return 'No information'.

Restriction: Do not extract any tokens if a HEA... not available then return 'No information'. Do not violate the schema.

Choice of exemplars

Sometimes the exemplars can actually sabotage the model! So exemplar choice is a big deal

- Understudied area. No good survey I can find.

A) Rationalized prediction prompt with exemplars

Based on the document between the <doc> tags, classify the claim between the <claim> tags as SUPPORT, CONTRADICT, or NOINFO, with respect to the document. Respond in json format, with a 'label' field for the classification and an 'explanation' field listing the parts of the document that support the label.

{{NOINFO exemplar}}

<doc> BACKGROUND Adoption of new and underutilized ... **CONCLUSIONS This study substantiates previous findings related to vaccine price and presents new evidence to suggest that GAVI eligibility is associated with accelerated decisions to adopt Hib vaccine.** ... </doc>
<claim> **A country's Vaccine Alliance (GAVI) eligibility is not indictivate of accelerated adoption of the Hub vaccine.** </claim>
{"label":"CONTRADICT", "explanation":["**CONCLUSIONS This study substantiates previous findings related to vaccine price and presents new evidence to suggest that GAVI eligibility is associated with accelerated decisions to adopt Hib vaccine.**"]}

<doc> Chronic hepatitis C is the leading cause ... **Policies requiring discontinuation of methadone in 32% of all programs contradict the evidence base for efficacy of long-term replacement therapies and potentially result in relapse of previously stable patients.** </doc>
<claim> **32% of liver transplantation programs required patients to discontinue methadone treatment in 2001.** </claim>
{"label":"SUPPORT", "explanation":["**Policies requiring discontinuation of methadone in 32% of all programs contradict the evidence base for efficacy of long-term replacement therapies and potentially result in relapse of previously stable patients**"]}

<doc> **Genetically identical cells sharing an environment can display markedly different phenotypes**... allows a trial commitment to multicellularity that external signals could extend. </doc>
<claim> **Gene expression does not vary appreciably across genetically identical cells**. </claim>

{"label":"NOINFO", "explanation":[]} ✗

B) Evidence-only prompt (no rationalization)

{{Prompt instructions}}

<doc> Genetically identical cells sharing an environment can display markedly different phenotypes. </doc>
<claim> Gene expression does not vary appreciably across genetically identical cells. </claim>

{"label": "CONTRADICT"} ✓

C) No-exemplar prompt

{{Prompt instructions}}

{{Document and claim}}

{"label": "CONTRADICT", "explanation": ["Genetically identical cells sharing an environment can display markedly different phenotypes."]} ✓

Concluding thoughts

Zero and few shot learning

Anatomy of a prompt

Exemplar choice

Role-setting