Zero- and Few-Shot Learning

CS 780/880 Natural Language Processing Lecture 21

Samuel Carton, University of New Hampshire

Last lecture

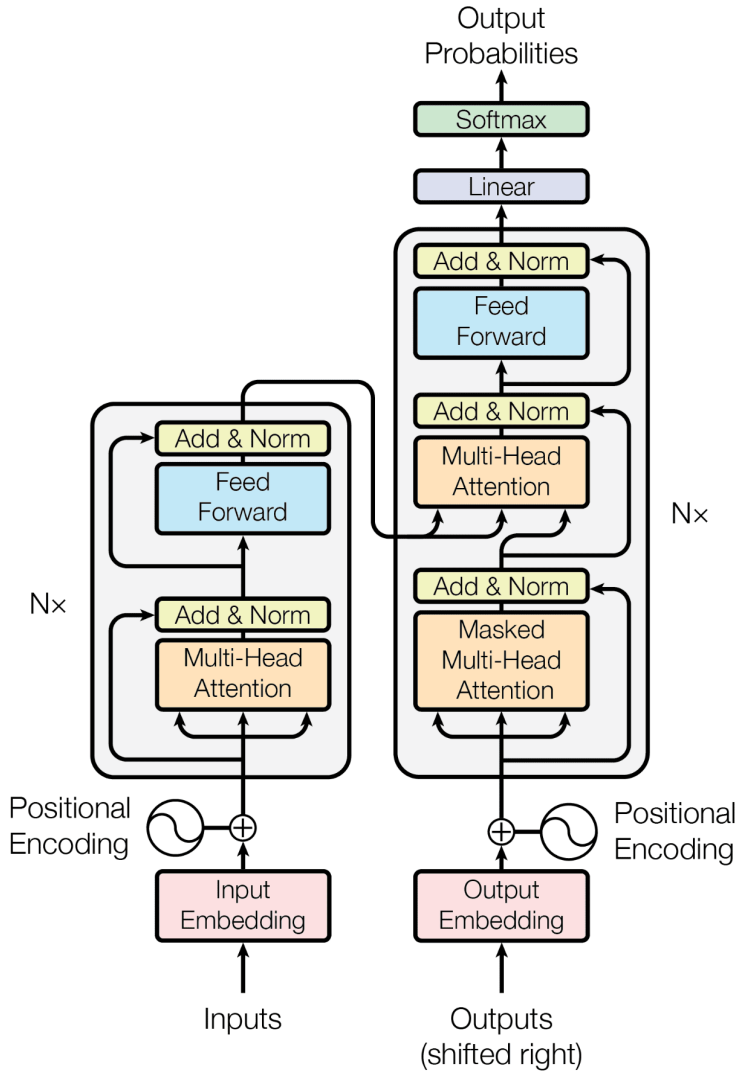
Pretrained transformer models

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

Encoder-decoder, encoder-only, decoder-only

How to choose?

- Generally, use the biggest you can train efficiently
- But use domain-specific models if appropriate
  - SciBERT
  - MatSciBERT
  - Galactica



Basic model training procedure

1. Pick your favorite model (BERT, T5, DistilBERT, etc.)
2. Download pretrained version of it from HuggingFace
3. Build an appropriate LightningModule around it
    1. Text classification, sequence tagging, translation, etc.
4. Find or construct a training/test dataset and preprocess it appropriately
5. Train the model using gradient descent
6. Evaluate the model
7. Profit

Transformer-using LightningModule

```
class BertClassifier(pl.LightningModule):
  def __init__(self,
               learning_rate:float,
               num_classes:int,
               freeze_bert:bool=False,
               **kwargs):
    super().__init__(**kwargs)

    # Like with the LSTM, we'll define a central BERT we're gonna use
    # Again, this will download this from Hugging Face in the background
    self.bert = BertModel.from_pretrained('bert-base-uncased')

    # If we want to speed up training, we can freeze the BERT module and train
    # just the output layer
    if freeze_bert:
      for param in self.bert.parameters():
        param.requires_grad = False

    # Then the only other thing we need is an output layer, whose input size will
    # be the BERT's output size (768), which can can find as follows:
    self.output_layer = torch.nn.Linear(self.bert.config.hidden_size, num_classes)

    self.learning_rate = learning_rate
    self.train_accuracy = Accuracy(task='multiclass', num_classes=num_classes)
    self.val_accuracy = Accuracy(task='multiclass', num_classes=num_classes)
```

Hugging Face model classes

Not only does HuggingFace make base models available, but they also have variants of each model for different kinds of tasks

E.g. BertForSequenceClassification

```
1517  class BertForSequenceClassification(BertPreTrainedModel):
1518      def __init__(self, config):
1519          super().__init__(config)
1520          self.num_labels = config.num_labels
1521          self.config = config
1522
1523          self.bert = BertModel(config)
1524          classifier_dropout = (
1525              config.classifier_dropout if config.classifier_dropout is not None else config.hidden_dropout_prob
1526          )
1527          self.dropout = nn.Dropout(classifier_dropout)
1528          self.classifier = nn.Linear(config.hidden_size, config.num_labels)
1529
1530          # Initialize weights and apply final processing
1531          self.post_init()
```

https://github.com/huggingface/transformers/blob/v4.28.0/src/transformers/models/bert/modeling_bert.py

Hugging Face model classes

My forward function

```
30  def forward(self, y:torch.Tensor, input_ids:torch.Tensor,
31                attention_mask:torch.Tensor):
32    # And then the forward function is pretty simple--
33    # way simpler than with the LSTM
34    bert_result = self.bert(input_ids=input_ids,
35                            attention_mask=attention_mask)
36    # Typically we just use the pooler output for classification
37    cls_output = bert_result['pooler_output']
38
39    py_logits = self.output_layer(cls_output)
40    py = torch.argmax(py_logits, dim=1)
41    loss = torch.nn.functional.cross_entropy(py_logits, y,
42                                             reduction='mean')
43    return {'py':py,
44            'loss':loss}
```

Their forward function

```
1562        outputs = self.bert(
1563            input_ids,
1564            attention_mask=attention_mask,
1565            token_type_ids=token_type_ids,
1566            position_ids=position_ids,
1567            head_mask=head_mask,
1568            inputs_embeds=inputs_embeds,
1569            output_attentions=output_attentions,
1570            output_hidden_states=output_hidden_states,
1571            return_dict=return_dict,
1572        )
1573
1574        pooled_output = outputs[1]
1575
1576        pooled_output = self.dropout(pooled_output)
1577        logits = self.classifier(pooled_output)
```

Hugging Face model classes

E.g. BertForTokenClassification

- __init__ pretty much identical to BertForSequenceClassification

```
1714  class BertForTokenClassification(BertPreTrainedModel):
1715      _keys_to_ignore_on_load_unexpected = [r"pooler"]
1716
1717      def __init__(self, config):
1718          super().__init__(config)
1719          self.num_labels = config.num_labels
1720
1721          self.bert = BertModel(config, add_pooling_layer=False)
1722          classifier_dropout = (
1723              config.classifier_dropout if config.classifier_dropout is not None else config.hidden_dropout_prob
1724          )
1725          self.dropout = nn.Dropout(classifier_dropout)
1726          self.classifier = nn.Linear(config.hidden_size, config.num_labels)
1727
1728          # Initialize weights and apply final processing
1729          self.post_init()
```

Hugging Face model classes

Difference is in forward functions.

- BertForSequenceClassification output layer operates on [CLS] token hidden vector, while BertForTokenClassification operates on **all** hidden vectors

BertForSequenceClassification

```
1562        outputs = self.bert(
1563            input_ids,
1564            attention_mask=attention_mask,
1565            token_type_ids=token_type_ids,
1566            position_ids=position_ids,
1567            head_mask=head_mask,
1568            inputs_embeds=inputs_embeds,
1569            output_attentions=output_attentions,
1570            output_hidden_states=output_hidden_states,
1571            return_dict=return_dict,
1572        )
1573
1574        pooled_output = outputs[1]
1575
1576        pooled_output = self.dropout(pooled_output)
1577        logits = self.classifier(pooled_output)
```

BertForTokenClassification

```
1758        outputs = self.bert(
1759            input_ids,
1760            attention_mask=attention_mask,
1761            token_type_ids=token_type_ids,
1762            position_ids=position_ids,
1763            head_mask=head_mask,
1764            inputs_embeds=inputs_embeds,
1765            output_attentions=output_attentions,
1766            output_hidden_states=output_hidden_states,
1767            return_dict=return_dict,
1768        )
1769
1770        sequence_output = outputs[0]
1771
1772        sequence_output = self.dropout(sequence_output)
1773        logits = self.classifier(sequence_output)
```

Hugging Face

Hugging Face (as accessed through the transformers Python library) is a fantastic resource.

- It's never been easier to grab powerful NLP models and begin customizing them to suit your own needs

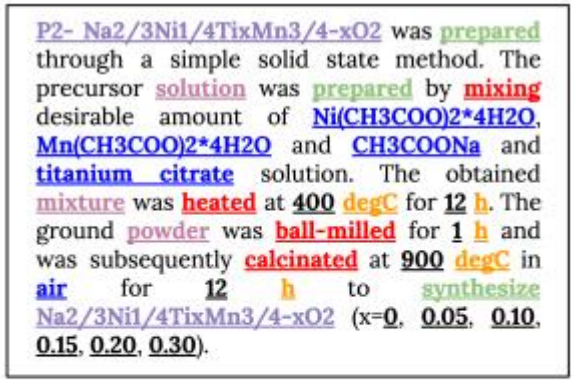
BUT.

The whole concept of "training" NLP models is beginning to go out the window in favor of **zero-** and **few-shot learning** with massive pretrained large language models (GPT-3, ChatGPT, GPT-4, etc.)

Example: Materials information extraction

O'Gorman et al., 2021 collect a nice dataset for **entity extraction** from the text of materials science papers

- 595 papers, 160k tokens
- Took a PhD student and 2 postdocs 3 months to do
- Trained BERT gets 89% token-by-token accuracy
  - (You'd use BertForTokenClassification for this)
- Would probably work even better with a Matsci-specific BERT variant
  - E.g. MatBERT, MatSciBERT, etc.



P2- Na2/3Ni1/4TixMn3/4-xO2 was prepared through a simple solid state method. The precursor solution was prepared by mixing desirable amount of Ni(CH3COO)2*4H2O, Mn(CH3COO)2*4H2O and CH3COONa and titanium citrate solution. The obtained mixture was heated at 400 degC for 12 h. The ground powder was ball-milled for 1 h and was subsequently calcinated at 900 degC in air for 12 h to synthesize Na2/3Ni1/4TixMn3/4-xO2 (x=0, 0.05, 0.10, 0.15, 0.20, 0.30).

Figure 1: Part of an example synthesis procedure included in the dataset with entity annotations from Zhao et al. (2015). Colors represent entity types and underlines represent span boundaries. Colors: Target, Nonrecipe-operation, Unspecified-Material, Operation, Material, Condition-Unit, Number.

MS-Mentions: consistently annotating entity mentions in materials science procedural text

T O'Gorman, Z Jensen, S Mysore... - Proceedings of the ..., 2021 - aclanthology.org

... Table 1 outlines the resultant size of our corpus, **MS-MENTIONS**. We also list the corpus statistics for the Materials Science Procedural Text (MSPT) corpus described in Mysore et al. (...

☆ Save 99 Cite Cited by 1 Related articles All 5 versions »

Example: Materials information extraction

Compare and contrast that with a prompt that Satanu came up with for getting GPT-3 to extract information from the abstract of a paper about alloys

Task is to extract information from the given paragraph based on the question.
####
Answer the question concisely only by extracting exact words from the paragraph.
#####
Question: What is the high entropy alloy system explored in the above text?

Paragraph: […] Here, we present a first-principles investigation of non-equimolar chromium-manganese-iron-cobalt-nickel (CrMnFeCoNi) HEAs and effects of molybdenum (Mo) and niobium (Nb) substitutions on cost, phase stability and solubility, and mechanical and thermal performance up to 1000 K operational temperature. […] Lower Ni concentration leads to lower thermal conductivity, indicating better thermal insulation, while reducing Mn concentration significantly increases the thermal conductivity, indicating better performing heat sinks. [...]
Answer: CrMnFeCoNi

[…] According to the paragraph, what impact does concentration of Ni have on the system? […]
Answer: Lower Ni concentration leads to lower thermal conductivity.

[…] According to the paragraph, what impact does concentration of Mn have on the system? […]
Answer: Lowering Mn concentration significantly increases the thermal conductivity.

[…] What are the target property explored according to the paragraph? […]
Answer: Thermal and mechanical properties, cost, phase stability and solubility, thermal performance, mechanical performance, thermal insulation, thermal conductivity, thermal expansion coefficient.

How is this possible?

Think back to why BERT & co turn out to be a good starting point for fine-tuning NLP models to do specific things (like information extraction)

- Analogous to teaching someone English before teaching them to do specific task

But what happens if you teach someone to be **really good** at English before teaching them to do the specific task?

- Then maybe you don't actually need to "teach" them to do the task, maybe you can just ask them to do it

Example: Materials information extraction

No real materials-science knowledge needed to do this task—just reading comprehension

Task is to extract information from the given paragraph based on the question.
####
Answer the question concisely only by extracting exact words from the paragraph.
#####
Question: What is the high entropy alloy system explored in the above text?

Paragraph: […] Here, we present a first-principles investigation of non-equimolar chromium-manganese-iron-cobalt-nickel (CrMnFeCoNi) HEAs and effects of molybdenum (Mo) and niobium (Nb) substitutions on cost, phase stability and solubility, and mechanical and thermal performance up to 1000 K operational temperature. […] Lower Ni concentration leads to lower thermal conductivity, indicating better thermal insulation, while reducing Mn concentration significantly increases the thermal conductivity, indicating better performing heat sinks. [...]
Answer: CrMnFeCoNi

[…] According to the paragraph, what impact does concentration of Ni have on the system? […]
Answer: Lower Ni concentration leads to lower thermal conductivity.

[…] According to the paragraph, what impact does concentration of Mn have on the system? […]
Answer: Lowering Mn concentration significantly increases the thermal conductivity.

[…] What are the target property explored according to the paragraph? […]
Answer: Thermal and mechanical properties, cost, phase stability and solubility, thermal performance, mechanical performance, thermal insulation, thermal conductivity, thermal expansion coefficient.

Language modeling versus world knowledge

And these language models can pick up world knowledge as well!

Remember that the training objective for language modeling is: given words {w0, w1, …,wt-1}, predict word wt.

So what happens when we apply our own internal language model to the following prompt:

The state nickname of New Hampshire is

Language modeling versus world knowledge

And these language models can pick up world knowledge as well!

Remember that the training objective for language modeling is: given words {w0, w1, …,wt-1}, predict word wt.

So what happens when we apply our own internal language model to the following prompt:

> The state nickname of New Hampshire is
> The Granite State

GPT-3 got it right! So… is that language modeling or is it world knowledge?

- Both!

GPT-3

- Published in 2020
- Decoder-only transformer model (same architecture as GPT-1 and 2)
- Trained on Common Crawl
  - 40 TB of text scraped from the web
- 175 billion parameters
- Blew a bunch of NLP benchmarks out of the water, particularly in **zero** and **few-shot** settings

Language models are **few-shot learners**
T Brown, B Mann, N Ryder... - Advances in neural ..., 2020 - proceedings.neurips.cc
... up **language models** greatly improves task-agnostic, **few-shot** ... GPT-3, an autoregressive **language model** with 175 billion ... **language model**, and test its performance in the **few-shot** ...
☆ Save 99 Cite Cited by 9123 Related articles All 18 versions »

Zero-shot learning

Basic idea: rather than fine-tune a language model to do a specific task (the way we would do with BERT, XLNet, etc.), we just **ask it to do what we want** and rely on its intrinsic capability to do it correctly.

Task is to extract information from the given paragraph based on the question.
####
Answer the question concisely only by extracting exact words from the paragraph.
#####
Question: What is the high entropy alloy system explored in the above text?

Paragraph: […] Here, we present a first-principles investigation of non-equimolar chromium-manganese-iron-cobalt-nickel (CrMnFeCoNi) HEAs and effects of molybdenum (Mo) and niobium (Nb) substitutions on cost, phase stability and solubility, and mechanical and thermal performance up to 1000 K operational temperature. […] Lower Ni concentration leads to lower thermal conductivity, indicating better thermal insulation, while reducing Mn concentration significantly increases the thermal conductivity, indicating better performing heat sinks. [...]
Answer: CrMnFeCoNi

[…] According to the paragraph, what impact does concentration of Ni have on the system? […]
Answer: Lower Ni concentration leads to lower thermal conductivity.

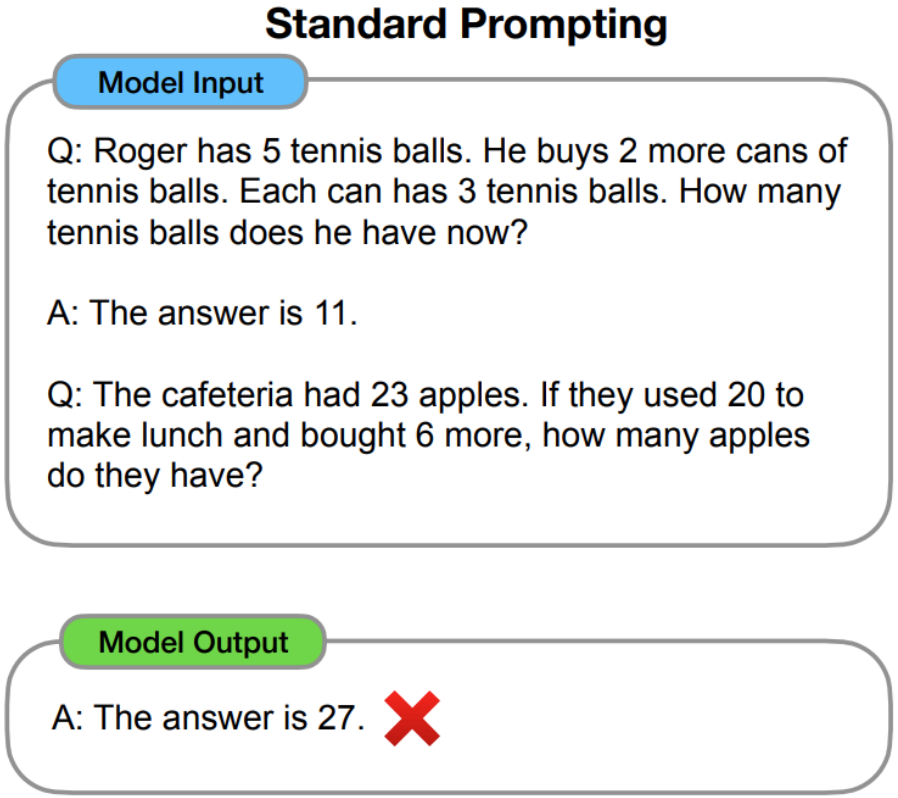
[…] According to the paragraph, what impact does concentration of Mn have on the system? […]
Answer: Lowering Mn concentration significantly increases the thermal conductivity.

[…] What are the target property explored according to the paragraph? […]
Answer: Thermal and mechanical properties, cost, phase stability and solubility, thermal performance, mechanical performance, thermal insulation, thermal conductivity, thermal expansion coefficient.

Few-shot learning

Basic idea: for even better performance than zero-shot learning, we give the model one or more examples of what we want it to do as part of the prompt

Popularized by Brown et al., (2020)



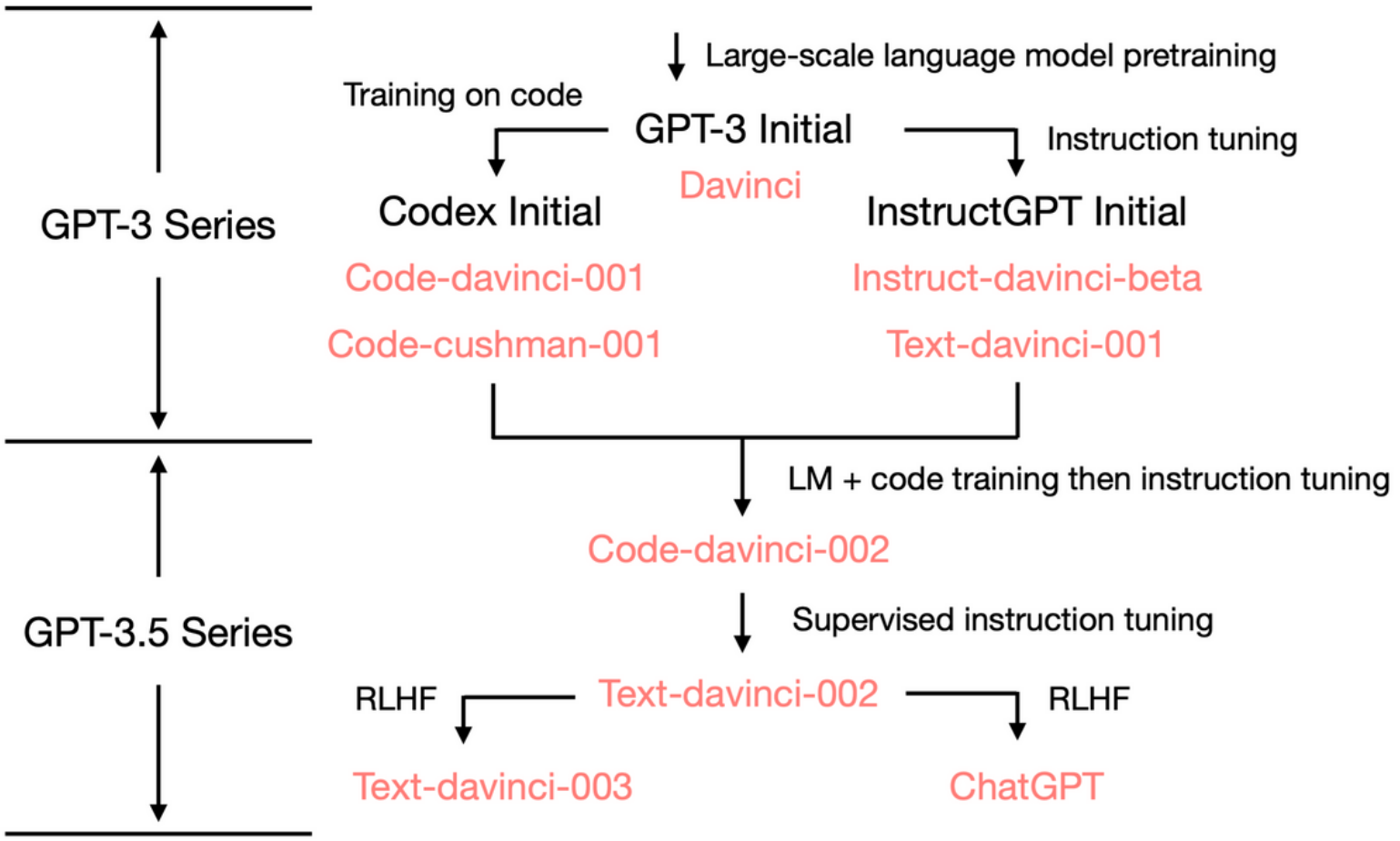
https://arxiv.org/pdf/2201.11903.pdf

ChatGPT

ChatGPT is now based on GPT-4

But when it was first released, it was based on a version of GPT-3 that had been additionally trained with:

- Code
- Instruction tuning
- Reinforcement Learning from Human Feedback (RLHF)



Concluding thoughts

GPT-3, ChatGPT, GPT-4

Traditional model training versus zero- and few-shot learning