01. BERT

Why was BERT way ahead of its time?

.

.

.

Because it was a masked language model even during pre-covid days!

Source: Ramsri Goutham

CS 2731
Introduction to Natural Language Processing

Session 13: LLMs, BERT and GPT

Michael Miller Yoder

October 9, 2024

School of Computing and Information

Course logistics

- No class on Monday (Fall Break)
- Project proposal presentations **next Wed Oct 16 during class**
  - Aim for **7 min max** presentations
  - There will be Q+A after each presentation
  - Add your slides here: Session 14 Project proposal presentations (CS 2731 Fall 2024).pptx
  - Instructions are on the project webpage and in the slide deck
- Project proposal and literature review **due next Thu Oct 17**
  - Instructions are on the project webpage
- Homework 3 has been released. Is **due Thu Oct 24**

Midterm OMET survey results (~40% response rate)

- What helps learning
  - Readings + lectures + quiz for review
  - Conversations/questions in class, visuals in slides
- What could be improved
  - More practical coding examples in class!
    - Lectures are theoretical and then homework assignments are all coding
  - Homework takes too long, especially if you aren't familiar with ML packages and frameworks
  - What are transformers again? Textbook just provides piles of Greek letters
- What I will change
  - Add in more coding examples, especially to get people started on the homework assignments
  - Consider lightening homework

Lecture overview: LLMs, BERT & GPT

- Intro to LLMs
- Pretraining and finetuning
- Pretraining 3 ways
  - Encoders (BERT)
  - Encoder-decoders (T5)
  - Decoders (GPT)
    - Sampling from decoder-only LLMs
    - (Briefly) in-context learning

Intro to large language models (LLMs): pretraining and finetuning

Large language models

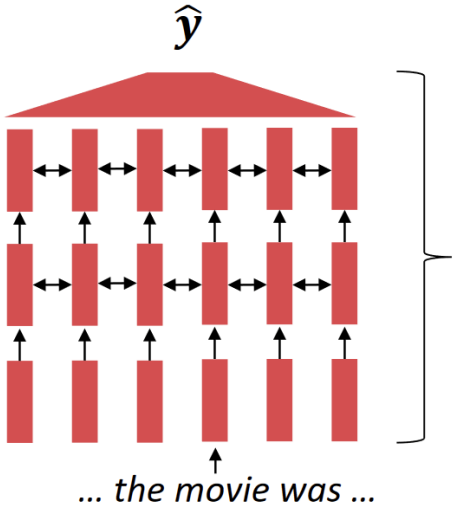
- Even through pretrained only to predict words
- Learn a lot of useful language knowledge
- Since training on a **lot** of text

Slide adapted from Jurafsky and Martin

Pretraining **whole models**

In contemporary NLP:

- All (or almost all) parameters in NLP networks are initialized via **pretraining**.
- Pretraining methods **hide parts of the input** from the model, and train the model to reconstruct those parts.
- This has been exceptionally effective at building strong:
  - representations of language
  - parameter initializations for strong NLP models
  - probability distributions over language that we can sample from



[This model has learned how to represent entire sentences through pretraining]

Slide adapted from John Hewitt

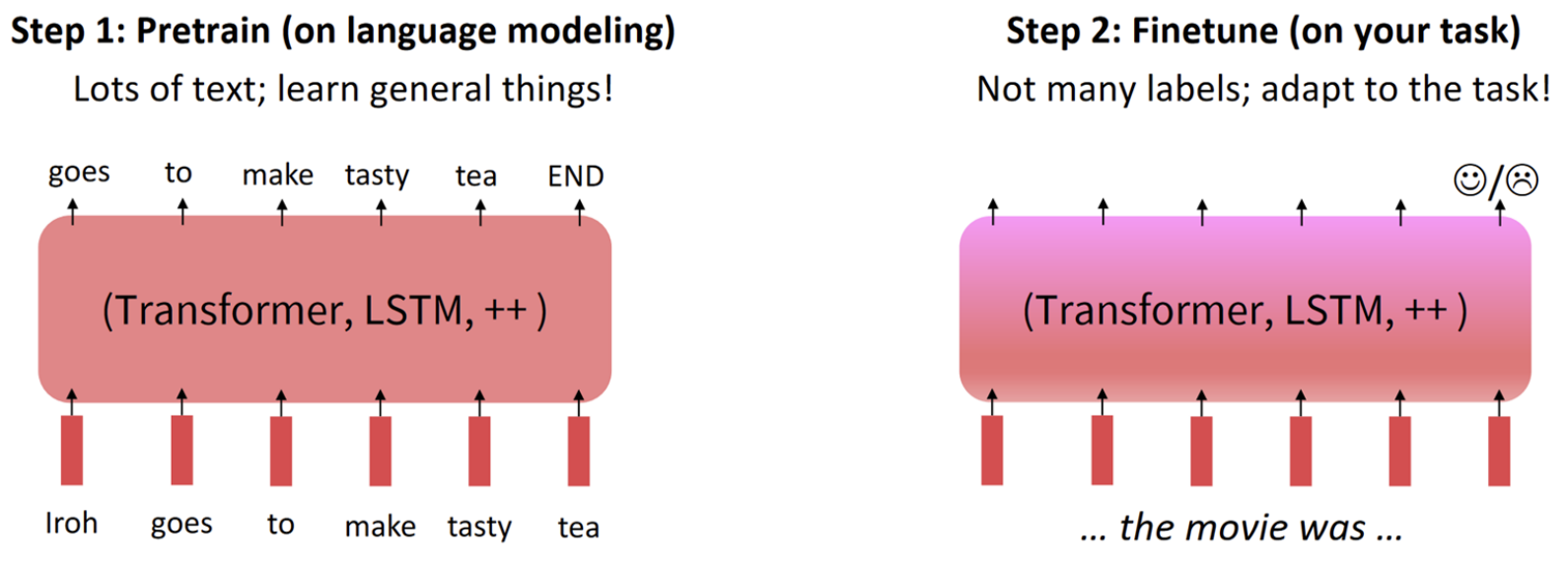
What can we learn from reconstructing the input?

- MIT is located in _____, Massachusetts.
- I put ___ fork down on the table.
- The woman walked across the street, checking for traffic over ___ shoulder.
- I went to the ocean to see the fish, turtles, seals, and _____.
- Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was ___.
- Iroh went into the kitchen to make some tea. Standing next to Iroh, Zuko pondered his destiny. Zuko left the _____.
- I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, ____

Slide adapted from John Hewitt

The pretraining + finetuning paradigm

Pretraining can improve NLP applications by serving as parameter initialization.



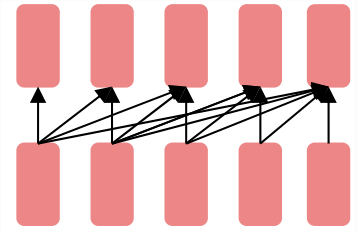
Slide adapted from John Hewitt

A reminder: Byte Pair Encoding (BPE, Sennrich+ 2016)

- LLMs generally use **subword tokenization**
- Merges frequently seen sequences of characters together into tokens
- Repeat:
  - Choose the two symbols that are most frequently adjacent in the training corpus (say 'A', 'B')
  - Add a new merged symbol 'AB' to the vocabulary
  - Replace every adjacent 'A' 'B' in the corpus with 'AB'.
  - Until *k* merges have been done.
- Allows them to generalize to unseen words, handle misspelling, novel words

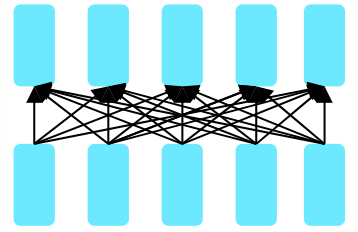
Slide adapted from Jurafsky & Martin

Three architectures for large language models



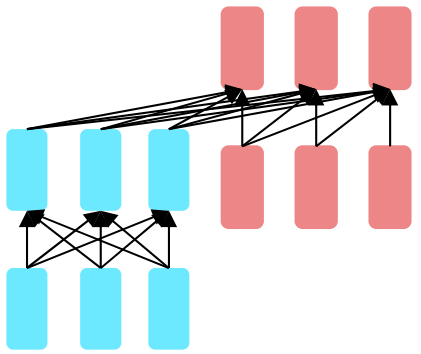
Decoders

GPT, Claude, Llama, Mixtral



Encoders

BERT family, HuBERT



Encoder-decoders

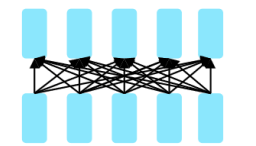
Flan-T5, Whisper

Slide adapted from Jurafsky and Martin

3 types of LLMs:
encoders, encoder-decoders, decoders

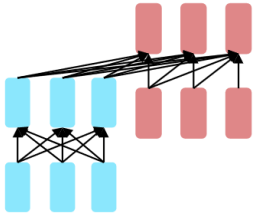
Pretraining for 3 types of architectures

The neural architecture influences the type of pretraining and natural use cases



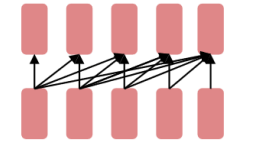
Encoders

- Gets bidirectional context – can condition on future!



Encoder-Decoders

- Good parts of decoders and encoders?



Decoders

- Nice to generate from; can't condition on future words

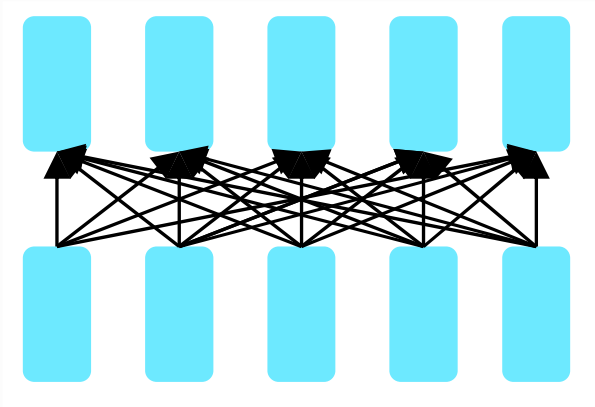
Slide adapted from John Hewitt

Encoders (BERT)

Encoders

Many varieties!

- Popular: Masked Language Models (MLMs)
- BERT family
- Trained by predicting words from surrounding words on both sides
- Are usually **finetuned** (trained on supervised data) for classification tasks.



Slide adapted from Jurafsky and Martin

Pretraining encoders: what pretraining objective to use?

- So far, we've looked at language model pretraining.
- But encoders can access to bidirectional context.
- Let's use it in training!
- BERT is pretrained with 2 objectives
  - Masked language modeling
  - Next sentence prediction

Slide adapted from John Hewitt

The Cloze Task

- The cloze task comes from psycholinguistics (the branch of linguistics and cognitive science that uses experimental methods to study how language works in human brains).
- It is a fill-in-the-blank task:

  He drove the yellow _____ into the front of our house.

- Subjects are presented with these frames and asked to fill in the missing words
- This allows experimenters to assess what a speaker understands about grammar, semantics, etc.
- According to the original BERT paper, this task provided the inspiration for BERT's masked language modeling (MLM) training task.
- But compare various kinds of denoising algorithms.

Slide credit: David Mortensen

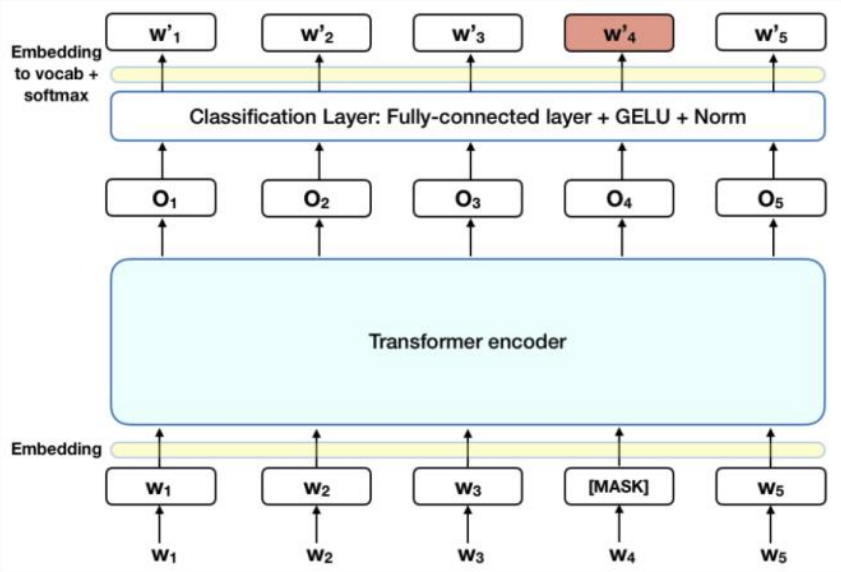
BERT is Trained to Perform Masked Language Modeling

- Since, in BERT's innards, everything is literally connected to everything, each word would indirectly see itself
- Enter masking!
- In training, random word are concealed from BERT using the `[MASK]` token
- BERT is trained to guess these masked words
- Take the sentence: "the girl was playing outside."
  - Suppose that a `[MASK]` token is inserted in place of *outside*, thus masking it
  - We then have "the girl was playing `[MASK]`."
  - The model now cannot "see" *outside*; instead it is trained to generate it based on context
  - This is part of why BERT can be so good at representing words according to the context in which they occur (not just the distribution of identically-spelled tokens)

Slide credit: David Mortensen

How BERT Is Trained to Perform Masked Language Modeling

1. Add classification layer
2. Multiply output vectors by embedding layer → vocabulary dimension
3. Calculate probabilities using softmax

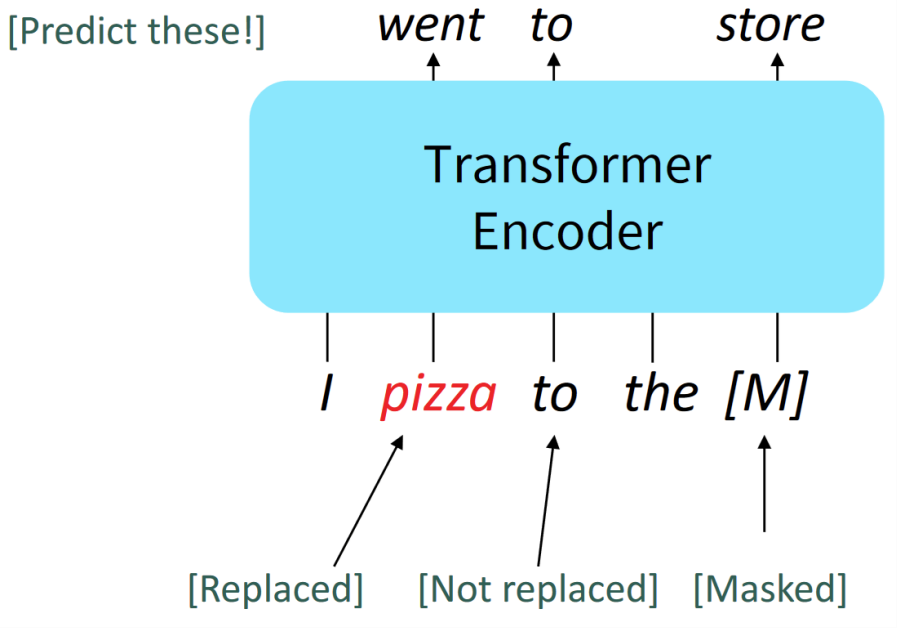


Slide credit: David Mortensen

BERT: Bidirectional Encoder Representations from Transformers

Some more details about Masked LM for BERT [Devlin et al. 2018]:

- Predict a random 15% of (sub)word tokens.
  - Replace input word with [MASK] 80% of the time
  - Replace input word with a random token 10% of the time
  - Leave input word unchanged 10% of the time (but still predict it!)
    - Why? Doesn't let the model get complacent and not build strong representations of non-masked words. (No masks are seen at fine-tuning time!)



Slide adapted from John Hewitt

BERT: Bidirectional Encoder Representations from Transformers

Details about BERT

- Two models were released:
  - BERT-base: 12 layers, 768-dim hidden states, 12 attention heads, 110 million params.
  - BERT-large: 24 layers, 1024-dim hidden states, 16 attention heads, 340 million params.
- Trained on:
  - BooksCorpus (800 million words)
  - English Wikipedia (2,500 million words)
- Pretraining is expensive and impractical on a single GPU.
  - BERT was pretrained with 64 TPU chips for a total of 4 days.
  - (TPUs are special tensor operation acceleration hardware)
- Finetuning is practical and common on a single GPU
  - “Pretrain once, finetune many times.”

Slide adapted from John Hewitt

BERT: Bidirectional Encoder Representations from Transformers

- BERT was massively popular and hugely versatile
- Finetuning BERT led to new state-of-the-art results on a broad range of tasks.

- QQP: Quora Question Pairs (detect paraphrase questions)
- QNLI: natural language inference over question answering data
- SST-2: sentiment analysis

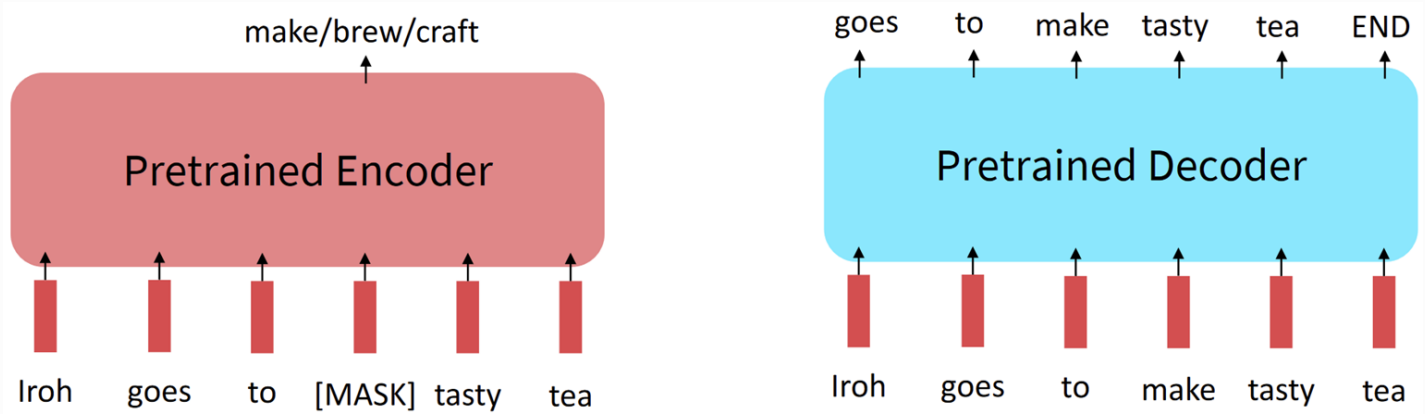
- CoLA: corpus of linguistic acceptability (detect whether sentences are grammatical.)
- STS-B: semantic textual similarity
- MRPC: Microsoft paraphrase corpus
- RTE: a small natural language inference corpus

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Slide adapted from John Hewitt

BERT: Bidirectional Encoder Representations from Transformers

- Those results looked great! Why not used pretrained encoders for everything?
- If your task involves generating sequences, consider using a pretrained decoder; BERT and other pretrained encoders don't naturally lead to nice autoregressive (1-word-at-a-time) generation methods.



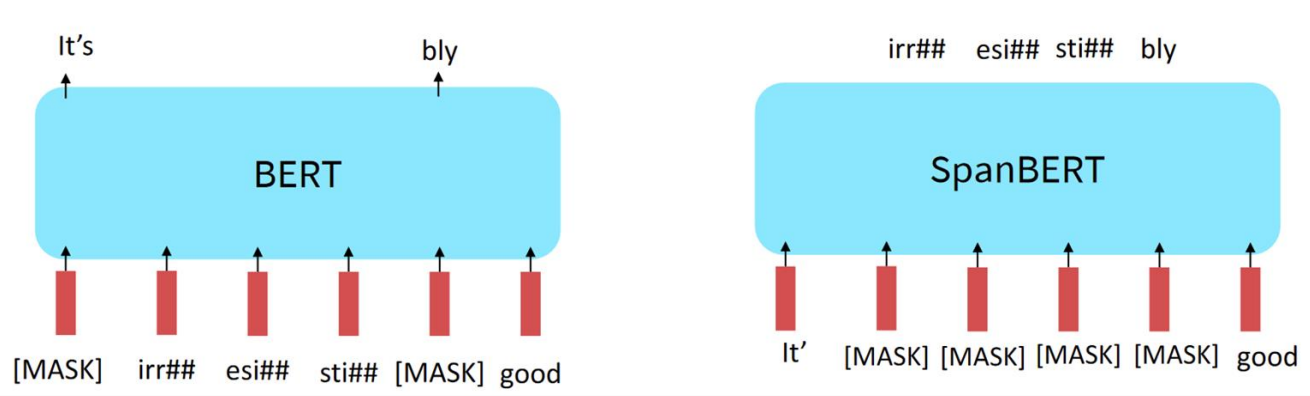
Slide adapted from John Hewitt

Extensions of BERT

You'll see a lot of BERT variants like RoBERTa, SpanBERT, etc

Some generally accepted improvements to the BERT pretraining formula:

- RoBERTa [Liu et al. 2019]: mainly just train BERT for longer and remove next sentence prediction!
- SpanBERT [Joshi et al. 2020]: masking contiguous spans of words makes a harder, more useful pretraining task

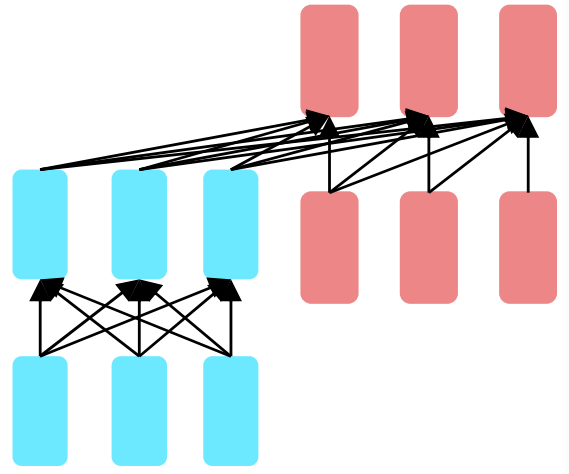


Slide adapted from John Hewitt

Encoder-decoders (T5)

Encoder-Decoders

- Trained to map from one sequence to another
- Very popular for:
  - machine translation (map from one language to another)
  - speech recognition (map from acoustics to words)



Slide adapted from Jurafsky and Martin

Pretraining encoder-decoders: what pretraining objective to use?

What Raffel et al. 2020 found to work best was span corruption. Their model: T5.

Replace different-length spans from the input with unique placeholders; decode out the spans that were removed!

This is implemented in text preprocessing: it's still an objective that looks like language modeling at the decoder side

Slide adapted from John Hewitt

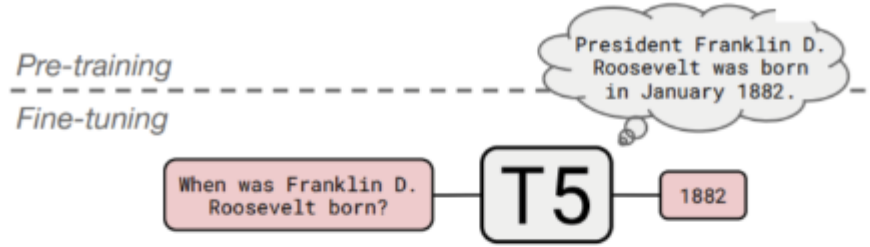
Finetuned T5

A fascinating property of T5: it can be finetuned to answer a wide range of questions, retrieving knowledge from its parameters.

NQ: Natural Questions

WQ: WebQuestions

TQA: Trivia QA



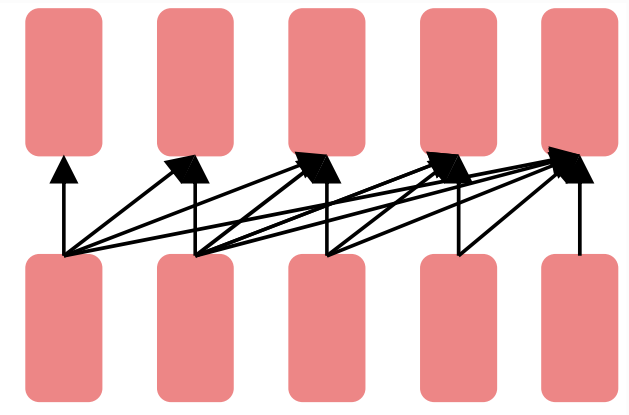
	NQ	WQ	TQA dev	TQA test	
Karpukhin et al. (2020)	**41.5**	42.4	**57.9**	–	
T5.1.1-Base	25.7	28.2	24.2	30.6	**220 million params**
T5.1.1-Large	27.3	29.5	28.5	37.2	**770 million params**
T5.1.1-XL	29.5	32.4	36.0	45.1	**3 billion params**
T5.1.1-XXL	32.8	35.6	42.9	52.5	**11 billion params**
T5.1.1-XXL + SSM	35.2	**42.8**	51.9	**61.6**	

Decoders (GPT)

Decoder-only models

Also called:

- Causal LLMs
- Autoregressive LLMs
- Left-to-right LLMs



- Predict words left to right

Slide adapted from Jurafsky and Martin

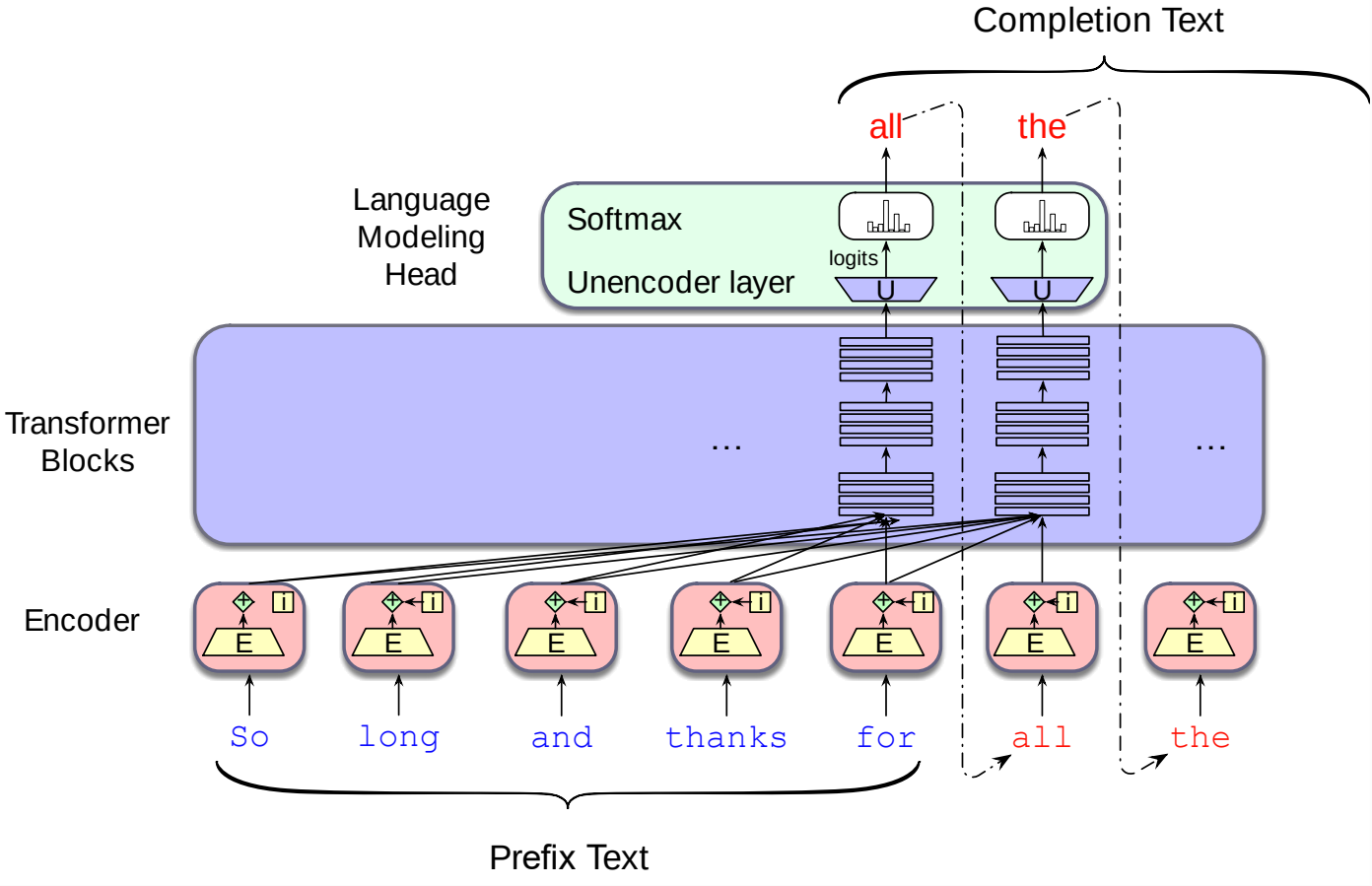
Decoder-only models can handle many tasks

- Many tasks can be turned into tasks of predicting words!

Slide adapted from Jurafsky and Martin

Conditional generation

Generating text conditioned on previous text!



Slide adapted from Jurafsky and Martin

Many practical NLP tasks can be cast as word prediction!

Sentiment analysis: “I like Jackie Chan”

1. We give the language model this string:
   `The sentiment of the sentence "I like Jackie Chan" is:`
2. And see what word it thinks comes next:

$P(\text{positive} \mid \texttt{The sentiment of the sentence ``I like Jackie Chan" is:})$

$P(\text{negative} \mid \texttt{The sentiment of the sentence ``I like Jackie Chan" is:})$

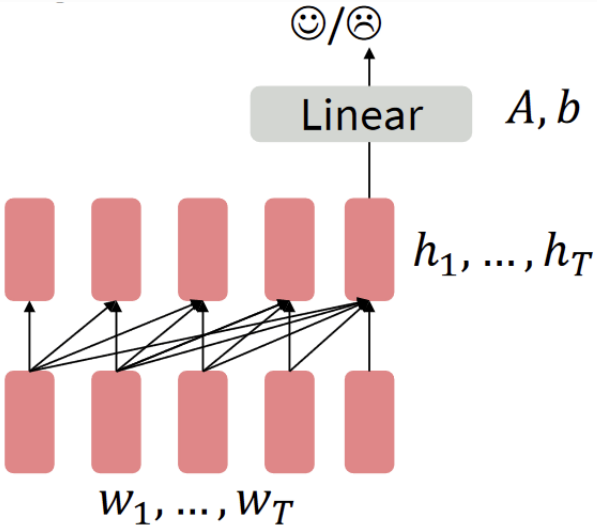
Slide adapted from Jurafsky and Martin

Pretrained decoders for classification

- When using language model pretrained decoders for classification, we can ignore that they were trained to model $p(w_t|w_{1:t-1})$.
- We can finetune them by training a classifier on the last word's hidden state.

$$h_1, \dots, h_T = \text{Decoder}(w_1, \dots, w_T)$$
$$y \sim Ah_T + b$$

- Where A and b are randomly initialized and specified by the downstream task.
- Gradients backpropagate through the whole network.



[Note how the linear layer hasn't been pretrained and must be learned from scratch.]

Slide adapted from John Hewitt

Framing lots of tasks as conditional generation

QA: “Who wrote The Origin of Species”

1. We give the language model this string:

`Q: Who wrote the book ``The Origin of Species"? A:`

2. And see what word it thinks comes next:

$P(w|$`Q: Who wrote the book ``The Origin of Species"? A:`$)$

3. And iterate:

$P(w|$`Q: Who wrote the book ``The Origin of Species"? A: Charles`$)$

Slide adapted from Jurafsky and Martin

Summarization

Original

The only thing crazier than a guy in snowbound Massachusetts boxing up the powdery white stuff and offering it for sale online? People are actually buying it. For $89, self-styled entrepreneur Kyle Waring will ship you 6 pounds of Boston-area snow in an insulated Styrofoam box – enough for 10 to 15 snowballs, he says.

But not if you live in New England or surrounding states. “We will not ship snow to any states in the northeast!” says Waring’s website, ShipSnowYo.com. “We’re in the business of expunging snow!”

His website and social media accounts claim to have filled more than 133 orders for snow – more than 30 on Tuesday alone, his busiest day yet. With more than 45 total inches, Boston has set a record this winter for the snowiest month in its history. Most residents see the huge piles of snow choking their yards and sidewalks as a nuisance, but Waring saw an opportunity.

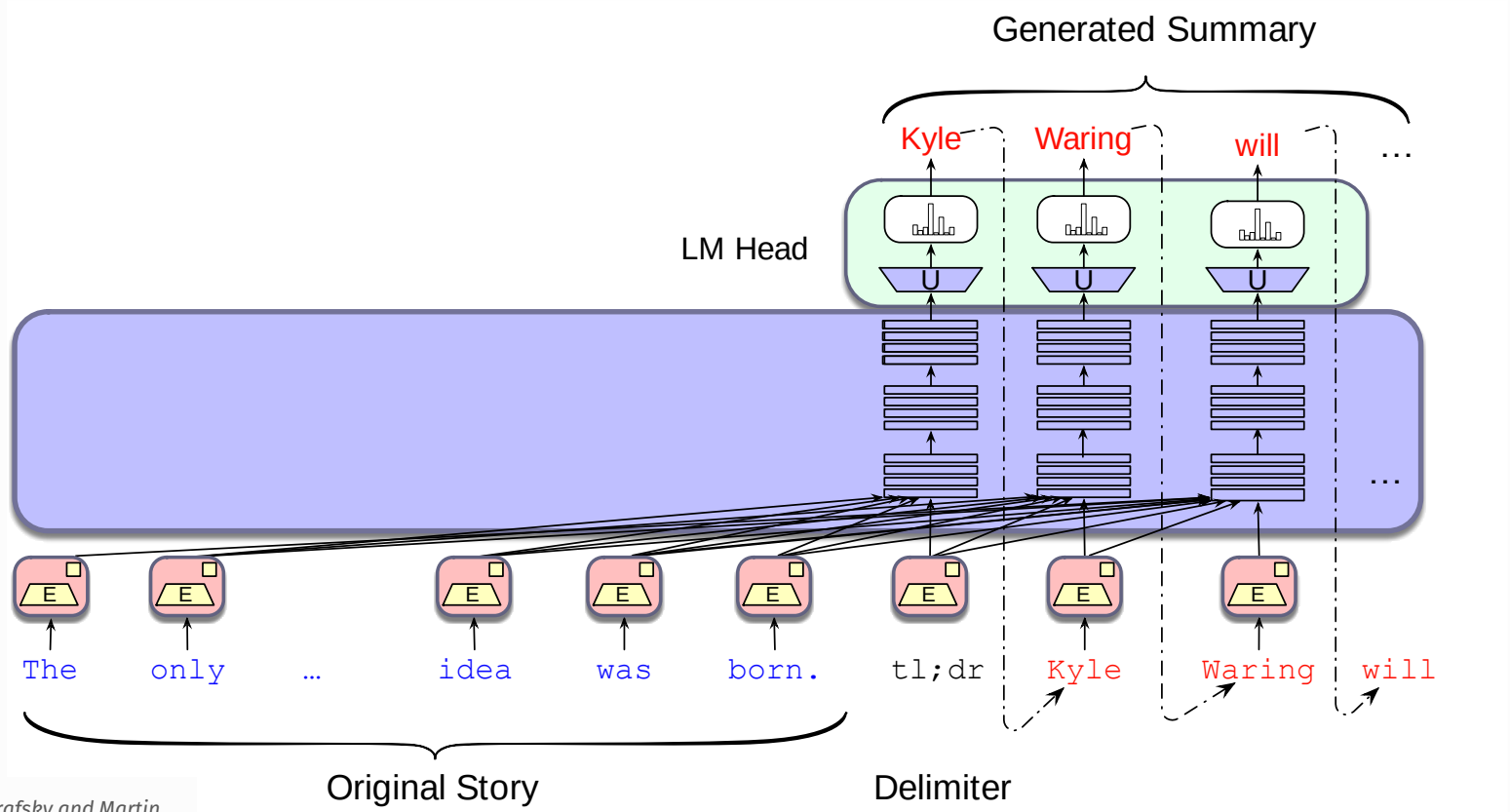
According to Boston.com, it all started a few weeks ago, when Waring and his wife were shoveling deep snow from their yard in Manchester-by-the-Sea, a coastal suburb north of Boston. He joked about shipping the stuff to friends and family in warmer states, and an idea was born. [...]

Summary

Kyle Waring will ship you 6 pounds of Boston-area snow in an insulated Styrofoam box – enough for 10 to 15 snowballs, he says. But not if you live in New England or surrounding states.

Slide adapted from Jurafsky and Martin

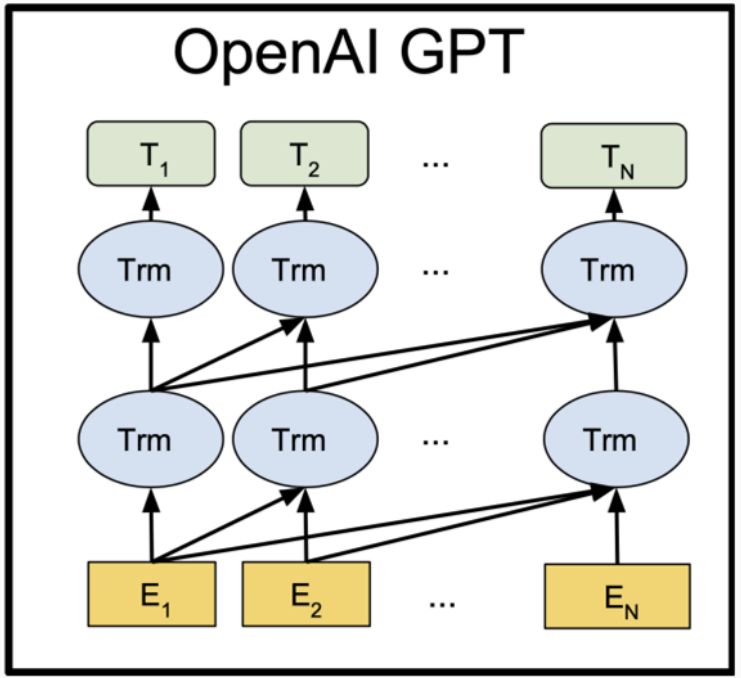
LLMs for summarization (using tl;dr)



Slide adapted from Jurafsky and Martin

Generative Pretrained Transformer (GPT; Radford et al. 2018)

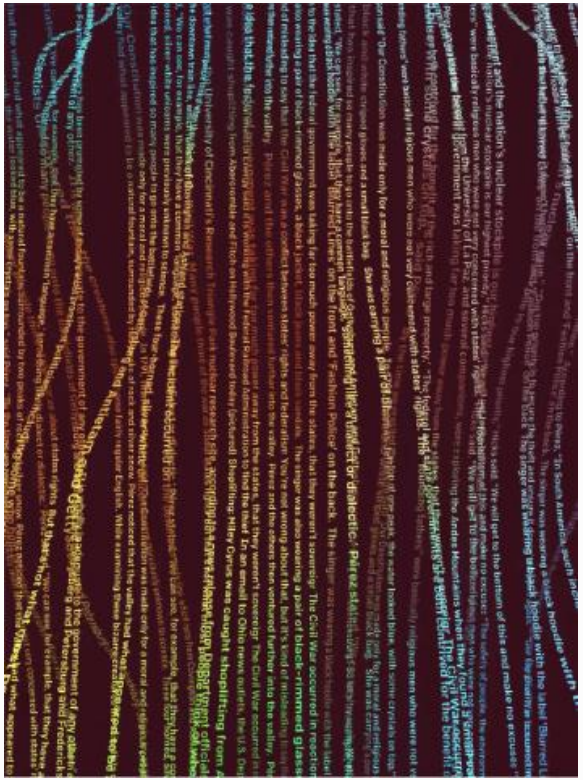
- 2018's GPT was a big success in pretraining a decoder!
- Transformer decoder with 12 layers, 117M parameters.
- 768-dimensional hidden states, 3072-dimensional feed-forward hidden layers.
- Byte-pair encoding with 40,000 merges
- Trained on BooksCorpus: over 7000 unique books.
  - Contains long spans of contiguous text, for learning long-distance dependencies.
- The acronym "GPT" never showed up in the original paper; it could stand for "Generative PreTraining" or "Generative Pretrained Transformer"



Slide adapted from John Hewitt, David Mortensen

GPT-2, GPT-3, GPT-4 from OpenAI

- They are basically larger and larger autoregressive transformer LMs trained on larger and larger amounts of data
- They have shown amazing language generation capability when you give it a prompt (aka. prefix, the beginning of a paragraph)



Slide adapted from Tianxing He

Generation example from the GPT-2 model

SYSTEM PROMPT (HUMAN-WRITTEN)	*In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.*
MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES)	The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved. Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

A sample from GPT2 (with top-k sampling)

Slide credit: Tianxing He

Sampling for LLM generation

Decoding and Sampling

- This task of choosing a word to generate based on the model's probabilities is called **decoding**.
- The most common method for decoding in LLMs: **sampling**.
- Sampling from a model's distribution over words:

• choose random words according to their probability assigned by the model.

●After each token we'll sample words to generate according to their probability *conditioned on our previous choices*,

• A transformer language model will give the probability

Random sampling

$\mathrm{i} \leftarrow 1$

$w_i \sim \mathrm{p}(w)$

while w_i != EOS

  $\mathrm{i} \leftarrow \mathrm{i} + 1$

  $w_i \sim \mathrm{p}(w_i \mid w_{<i})$

Random sampling doesn't work very well

- Even though random sampling mostly generate sensible, high-probable words,
- There are many odd, low- probability words in the tail of the distribution
- Each one is low- probability but added up they constitute a large portion of the distribution
- So they get picked enough to generate weird sentences

Factors in word sampling: **quality** and **diversity**

Emphasize **high-probability** words
+ **quality**: more accurate, coherent, and factual,
- **diversity**: boring, repetitive.

Emphasize **middle-probability** words
+ **diversity**: more creative, diverse,
- **quality**: less factual, incoherent

Top-k sampling:

1. Choose # of words *k*
2. For each word in the vocabulary *V* , use the language model to compute the likelihood of this word given the context $p(wt | w_{<t})$
3. Sort the words by likelihood, keep only the top *k* most probable words.
4. Renormalize the scores of the *k* words to be a legitimate probability distribution.
5. Randomly sample a word from within these remaining *k* most-probable words according to its probability.

Top-p sampling (= nucleus sampling)

Holtzman et al., 2020

Problem with top-*k*: *k* is fixed so may cover very different amounts of probability mass in different situations

Idea: Instead, keep the top p percent of the probability mass

Given a distribution $P(w_t \mid \mathbf{w}_{<t})$, the top-*p* vocabulary $V(p)$ is the smallest set of words such that

$$\sum_{w \in V^{(p)}} P(w|\mathbf{w}_{<t}) \geq p$$

Temperature sampling

Reshape the distribution instead of truncating it

Intuition from thermodynamics,

- a system at high temperature is flexible and can explore many possible states,
- a system at lower temperature is likely to explore a subset of lower energy (better) states.

In **low-temperature sampling**, ($\tau \leq 1$) we smoothly

- increase the probability of the most probable words
- decrease the probability of the rare words.

Temperature sampling

Divide the logit by a temperature parameter τ before passing it through the softmax.

Instead of $\mathbf{y} = \text{softmax}(u)$

We do $\mathbf{y} = \text{softmax}(u/\tau)$

Temperature sampling

$$\mathbf{y} = \text{softmax}(u/\tau)$$

$0 \leq \tau \leq 1$

Why does this work?

- When τ is close to 1 the distribution doesn't change much.
- The lower of a fraction τ is, the larger the scores being passed to the softmax
- Softmax pushes high values toward 1 and low values toward 0.
- Large inputs (lower temperature) pushes high-probability words higher and low probability word lower, making the distribution more greedy.
- As τ approaches 0, the probability of most likely word approaches 1

In-context learning

GPT-3, in-context learning, and very large models

So far, we've interacted with pretrained models in two ways:

- Sample from the distributions they define (maybe providing a prompt)
- Fine-tune them on a task we care about, and take their predictions.

Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts.

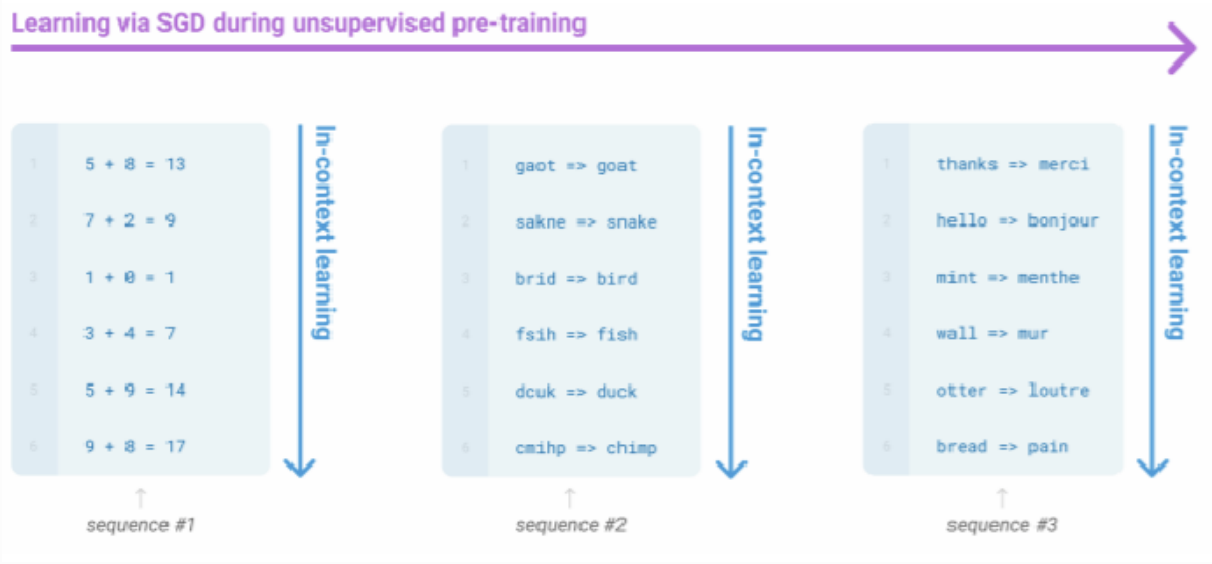
GPT-3 is the canonical example of this. The largest T5 model had 11 billion parameters.

GPT-3 has **175 billion parameters**.

Slide adapted from John Hewitt

GPT-3, in-context learning, and very large models

Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts. The in-context steps seem to specify the task to be performed.



Slide adapted from John Hewitt

Wrapping up

- Transformer-based language models pretrained on lots of text are called **large language models (LLMs)**
- LLMs can have decoder-only, encoder-only, or encoder-decoder architectures
- Decoder-only LLMs can cast many different NLP tasks as word prediction
- There are many different sampling approaches that balance diversity and quality in text generation from LLMs

Questions?