CS 2731
Human Language Technologies

Session 14: Transformers part 2, introduction to LLMs

Michael Miller Yoder

October 13, 2025

School of Computing and Information

Course logistics: project

- This **Wed Oct 15**: project peer group feedback
  - Michael has not completely planned this yet, but come with ideas for your proposal and lingering questions you have that classmates may be able to help with
- Next project deliverable: project proposal due **this Thu Oct 16**
  - Will include plans for **task, data, methods, evaluation**
  - Compare multiple approaches, including an LLM-based approach
  - Literature review of at least 3 related papers
    - Baselines to compare your approach to
  - Feel free to email or book office hours with Michael to discuss

Course logistics: project

- Project proposal presentations in-class **next Mon Oct 20**
  - Michael will start a shared PowerPoint that you will add slides to, along with instructions
  - Ungraded
- LLM access
  - We have $150 total as a class to use on OpenAI LLM credits
  - Access to open-source LLM set up on School of Computing and Information servers for API access is coming soon
    - Gemma, LLaMa, Deepseek

Course logistics: Homework 2

- Congrats to Surabhi, Chase, Nate and Lucy!
- Michael is planning on grading HW2 this week

HW2 Deception classification, CS 2731 Fall 2025

Late Subm

Overview | Data | Discussion | Leaderboard | Rules | Team | Submissions | Settin

The private leaderboard is calculated with approximately 65% of the test data.
This competition has completed. This leaderboard reflects the final standings.

#	△	Team	Members	Score	Entries
1	▲ 2	Surabhi Raghavan		0.681034	3
2	▲ 7	Chase Lahner		0.672413	1
3	▲ 5	Nathaniel Ginck		0.663793	1
4	▲ 1	lucy		0.663793	2

How to do a literature review

- Look for NLP papers related to your topic in ACL Anthology, Semantic Scholar, and Google Scholar
- For each paper, note:
  - What they cite in their related work sections (find those papers, iterate)
  - Data
  - Methods
  - Findings
- For at least 3 papers, **organize them into themes of approaches, datasets, findings**
- Ok: X paper did this, Y paper did this, Z paper did that
- Good: X and Y papers did this, while Z improved with that
- Best: X and Y papers did this, Z improved, nobody has yet to do...

Example literature review: removing private health information (PHI)

3 Literature Review

Over the years, several methods have been developed to automate the process of de-identification of PHI tags in EHRs, ranging from rule-based systems to the more recent advances in machine learning, especially large language models (LLMs).

3.1 Earlier methods for NER detection

Early NER methods for PHI identification relied heavily on rule-based approaches or feature-engineered machine learning models. These relied heavily on manual feature extraction, where common features included part-of-speech tags, word boundaries, and gazetteer lists for common PHI terms. These had limitations in handling the complexity and variability inherent in clinical narratives, especially in identifying PHI in unstructured or noisy data.

Deep learning models, particularly Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs), became widely adopted for NER tasks. These models were capable of capturing sequential dependencies in text, which is essential for understanding context and relationships between entities in clinical notes. (Liu et al., 2017) demonstrated that a combination of LSTMs and CRFs could significantly improve the identification of PHI in clinical records achieving overall precision score of 95% when applied to the i2b2 dataset.

3.2 Transformer based models for PHI De-identification

Transformer-based architectures have significantly advanced PHI de-identification by leveraging contextual embeddings and attention mechanisms. These models have consistently outperformed traditional rule-based or statistical methods in recognizing and anonymizing sensitive data in clinical texts.

BERT (Bidirectional Encoder Representations from Transformers) (Devlin, 2018) and its variants, such as BioBERT (Lee et al., 2020) and ClinicalBERT (Huang et al., 2019), have been foundational in PHI de-identification. BERT achieves F1 scores of over 94% on the i2b2 2014 dataset by leveraging bidirectional context. BioBERT, pre-trained on PubMed and PMC data, enhances this performance, reaching up to 95.6% F1 scores for PHI recognition tasks. RoBERTa (Robustly Optimized BERT Pretraining) (Liu, 2019) outperforms BERT through better training optimization techniques, achieving F1 scores of 96% on PHI deidentification datasets. ELECTRA (Clark, 2020) further improves computational efficiency and maintains comparable performance, achieving an F1 score of approximately 95.4% by focusing on discriminative learning.

Domain-specific models like ClinicalBERT and KeBioLM (Yuan et al., 2021) achieve even higher accuracy for clinical text. ClinicalBERT, fine-tuned on MIMIC-III, achieves F1 scores of up to 96.5% on de-identification tasks . KeBioLM integrates knowledge-based embeddings, achieving F1 scores of 97% for nested and complex entities, showcasing its adaptability to intricate PHI categories. Additionally, BioBART (Yuan et al., 2022), a pre-trained biomedical transformer model based on BART (Bidirectional and Auto-Regressive Transformers), achieves state-of-the-art performance for both entity recognition and text summarization in the biomedical domain, with competitive F1 scores of 96.8% on MIMIC dataset.

3.3 Large Language Models (LLMs) for PHI De-identification and NER tasks

The rise of large language models (LLMs) has further pushed the boundaries of de-identification tasks by leveraging extensive pretraining on diverse datasets and task-specific fine-tuning.

MedPaLM-2 (Singhal et al., 2023) and DeID-GPT (Liu et al., 2023) are among the most prominent models in this domain. MedPaLM-2, fine-tuned on medical texts, achieves F1 scores of 96.2% by utilizing task-specific enhancements such as better pretraining corpora and fine-grained token-level recognition. DeID-GPT, specifically developed for de-identification, surpasses MedPaLM-2 with an F1 score of 97.1%, attributed to its prompt-based approach for handling sensitive entities. GatorTron (Yang et al., 2022), a transformer designed for medical applications, excels in multi-task NLP, including semantic similarity and NER. On the i2b2 dataset, it achieves an F1 score of 96.8%, demonstrating its robustness for clinical text processing. Similarly, BioGPT, a biomedical GPT model, performs well in NER and relation extraction tasks, showcasing versatility in multiple downstream applications with an F1 score of 96.5%.

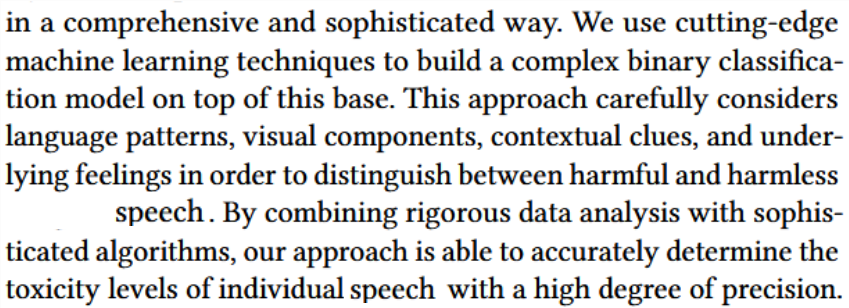
ClinicalT5 (Lu et al., 2022), an encoder-decoder architecture, has been shown to excel in classification and de-identification tasks, achieving F1 scores of 96% on i2b2, benefiting from its capability to capture both contextual and sequential dependencies effectively. Recent domain-specific adaptations of popular LLM architectures like GPT and LLama have further elevated their performance. For instance, MedGPT(Kraljevic et al., 2021) adapts GPT-3 for biomedical tasks, achieving state-of-the-art results in PHI redaction. Similarly, BioLLaMA, a fine-tuned variant of LLaMA-2, focuses on tasks like PHI de-identification and clinical summarization, with F1 scores of over 96% on the i2b2 dataset.

Clarity and using generative AI tools for writing

- Writing **clarity is what I grade** on homework and project reports, not grammar and spelling
  - Computer science writing values clarity and conciseness
- Writing from generative AI is often vague, abstract, wordy, and non-specific to what you did in your project. It isn't recommended
- Generative AI can generate claims that aren't backed up by your project
- ChatGPT and other LLMs don't know what you specifically did or are planning on doing in your project or homework. You could tell them, but at that point just write that down directly in your report!

Clarity and using generative AI tools for writing

- If you use generative AI tools for writing, aim for **machine-in-the-loop** writing where you as the human bear most of the rhetorical load (Knowles 2024)
  - AI is more like an assistant than a co-author
- Example of unclear project writing



in a comprehensive and sophisticated way. We use cutting-edge machine learning techniques to build a complex binary classification model on top of this base. This approach carefully considers language patterns, visual components, contextual clues, and underlying feelings in order to distinguish between harmful and harmless speech. By combining rigorous data analysis with sophisticated algorithms, our approach is able to accurately determine the toxicity levels of individual speech with a high degree of precision.

Review: Describe self-attention in transformers

Overview: Transformers part 2, intro to LLMs

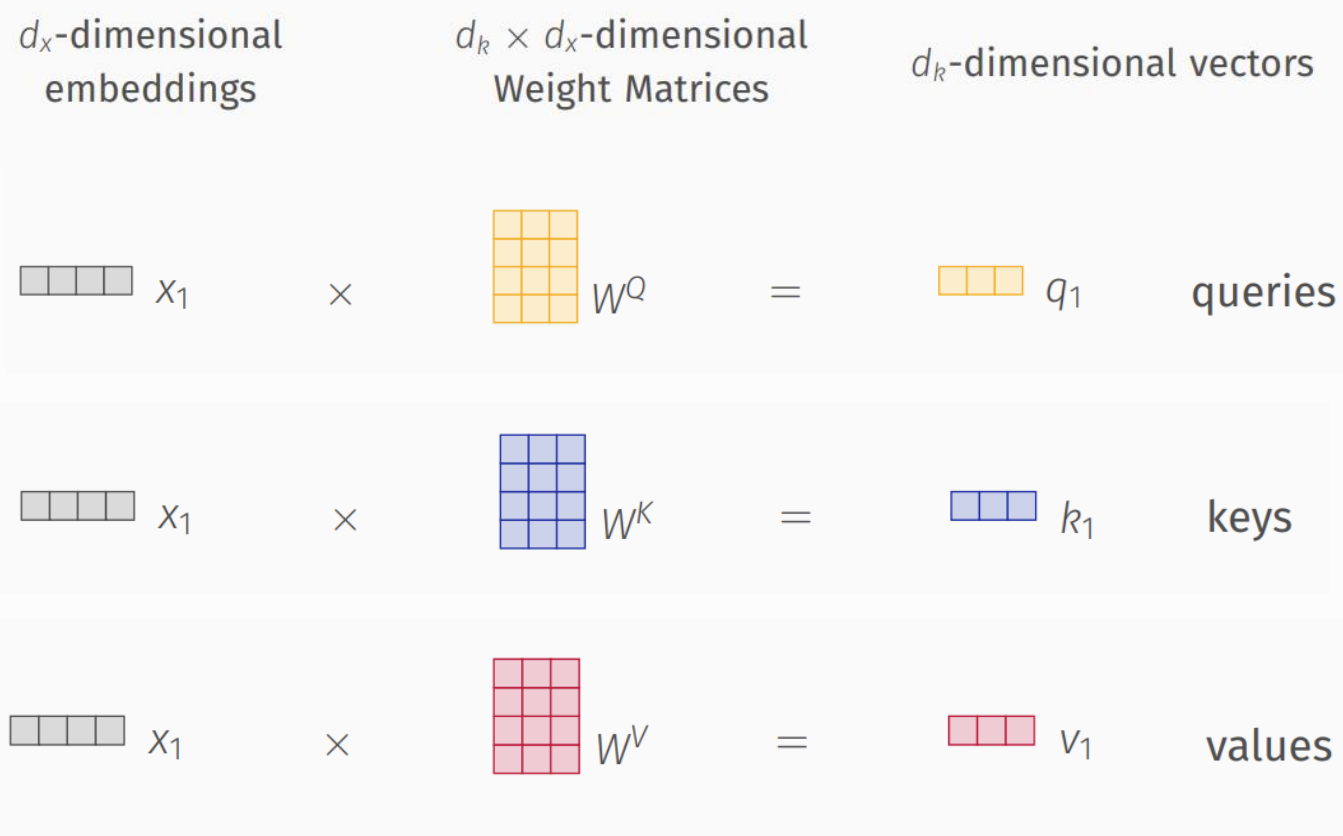
- Activity: work through self-attention
- Transformer input and output details
  - Position embeddings
  - Language modeling head
- Intro to LLMs
  - Pretraining LLMs
  - Sampling for LLM generation
  - Harms from LLMs
- Coding activity: fine-tune GPT-2

Activity: work through self-attention

Calculate transformed output for one input word

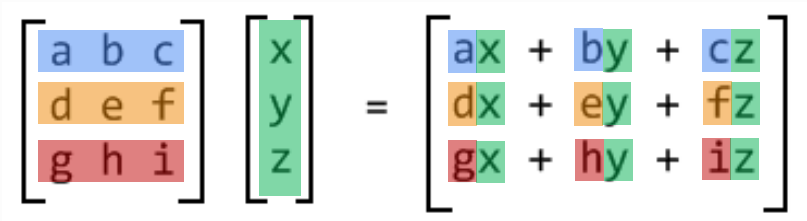
- Example sentence: “we wash our cats” (don’t ask)
- Let’s just calculate the vector output, for one input word: “we”
- High-level points to remember before you get buried in the math:
  - Each token will have an output vector that integrates contextual information from other tokens in the sentence
  - Each token can play a role as a query, key, and value
- Parameters (learned through backpropagation) are assumed given:
  - W^Q, W^K, W^V

Computing Self-Attention, Step One: Compute Key, Query, and Value Vectors

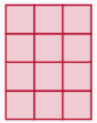


Slide credit: David Mortensen

Dot product: vector · matrix



Computing Self-Attention, Step One: Compute Key, Query, and Value Vectors

d_x-dimensional embeddings		$d_k \times d_x$-dimensional Weight Matrices		d_k-dimensional vectors	
$x_1 = [3, 0, 1, -0.5]$  x_1	$\times$	$W^Q = \begin{bmatrix} 1.5 & 1 & 2 \\ 3 & -2 & 5 \\ 1 & 2 & -2 \\ 9 & 4 & 2 \end{bmatrix}$	$=$	 q_1	queries
$x_1 = [3, 0, 1, -0.5]$  x_1	$\times$	$W^K = \begin{bmatrix} 1 & 0.5 & 2 \\ -2 & 0.5 & 3 \\ 0.5 & 2 & -3 \\ 5 & 3 & 2 \end{bmatrix}$	$=$	 k_1	keys
$x_1 = [3, 0, 1, -0.5]$  x_1	$\times$	 W^V	$=$	 v_1	values

Find q_1 and k_1

Slide credit: David Mortensen

Computing Self-Attention, Step Two: Weighted Sum of Value Vectors

	We	wash	our	cats
	x_1	x_2	x_3	x_4
query-key dot product	$q_1 \cdot k_1 =$	$q_1 \cdot k_2 =$	$q_1 \cdot k_3 =$	$q_1 \cdot k_4 =$
divide by $\sqrt{d_k}$				
softmax				
$\times$ value	$\times v_1$	$\times v_2$	$\times v_3$	$\times v_4$
sum	z_1			

Assume $d_k = 64$

$k_2 = [3, 4, 3]$
$k_3 = [5, 2, 3]$
$k_4 = [3, 2, 1]$

$v_1 = [1, 0.5, -1]$
$v_2 = [4, 5, -2]$
$v_3 = [-3, 2, 2]$
$v_4 = [1, 1, 6]$

Slide credit: David Mortensen

Transformer input and output

Token and Position Embeddings

- The matrix X (of shape $[N \times d]$) has an embedding for each word in the context.
- This embedding is created by adding two distinct embedding for each input: **token** and **position** embeddings
- Since self-attention doesn't build in order information, we need to encode the order of the sentence in our keys, queries, and values

Slide adapted from Jurafsky & Martin

Token Embeddings

Embedding matrix E has shape $|V| \times d$

- One row for each of the $|V|$ tokens in the vocabulary.
- Each word is a row vector of d dimensions

Given: string "*Thanks for all the*"

1. Tokenize with BPE and convert into vocab indices

w = [5,4000,10532,2224]

2. Select the corresponding rows from E, each row an embedding

(row 5, row 4000, row 10532, row 2224).

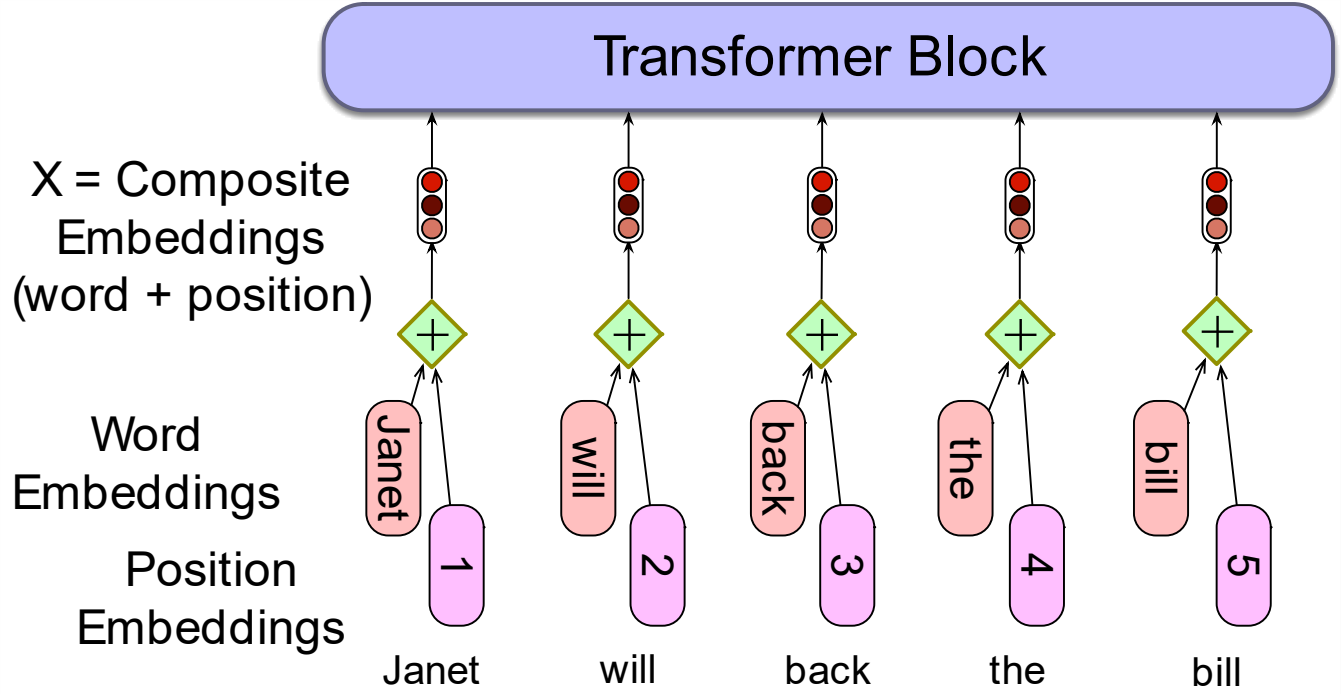
Slide adapted from Jurafsky & Martin

Position Embeddings

- There are many methods, but we'll just describe the simplest: absolute position.
- Goal: learn a position embedding matrix E_{pos} of shape $1 \times N$
- Start with randomly initialized embeddings
  - one for each integer up to some maximum length.
  - i.e., just as we have an embedding for the word *fish*, we'll have an embedding for position 3 and position 17.
- As with word embeddings, these position embeddings are learned along with other parameters during training.

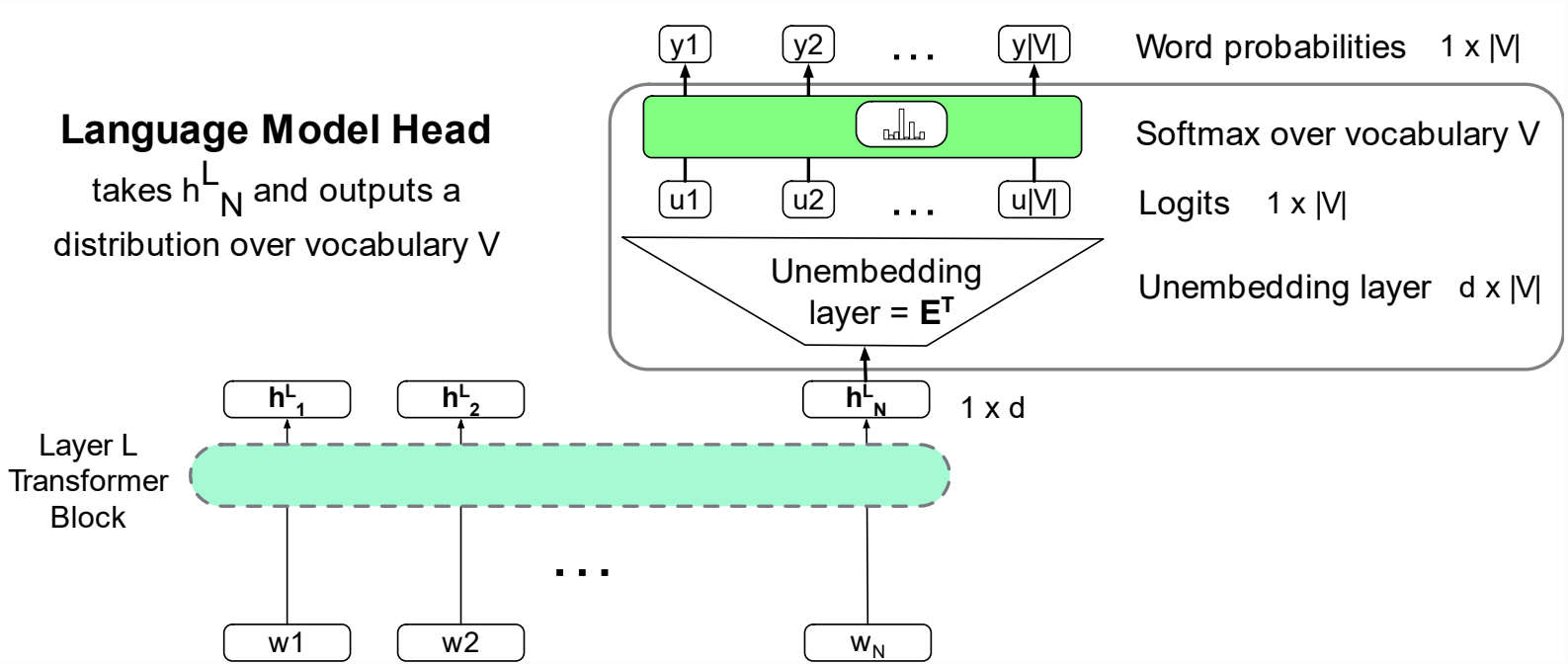
Slide adapted from Jurafsky & Martin

Each **x** is just the sum of word and position embeddings



Slide adapted from Jurafsky & Martin

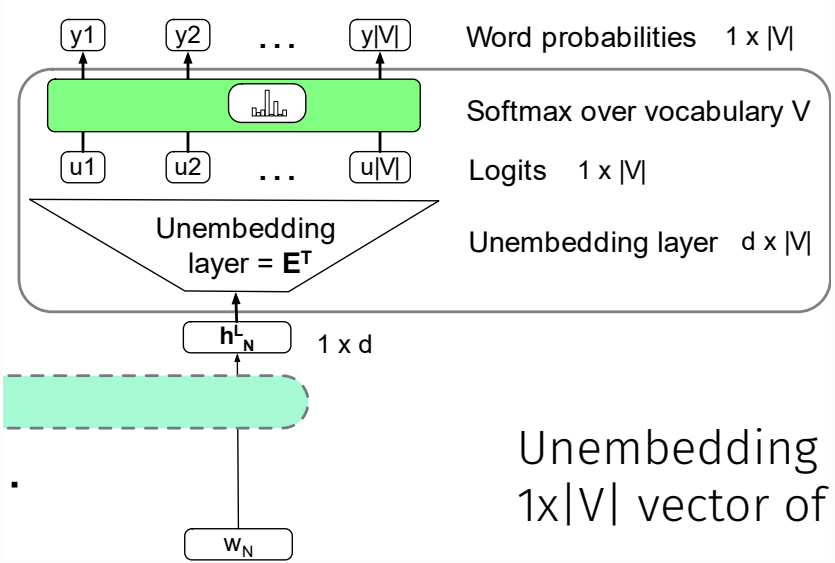
Language modeling head



Slide adapted from Jurafsky & Martin

Language modeling head

Unembedding layer: FFN layer projects from h^L_N (shape $1 \times d$) to probability distribution vector over the vocabulary



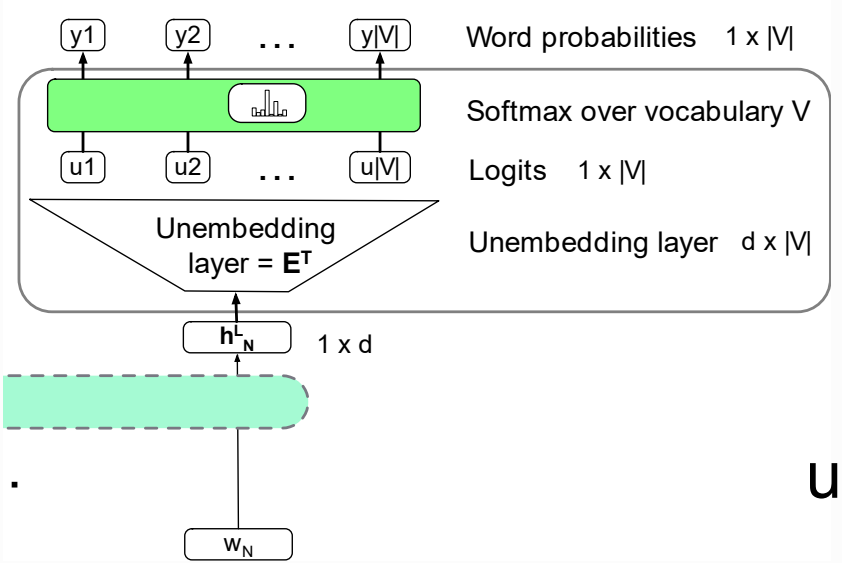
Why "unembedding"? **Tied** to E^T

Weight tying, we use the same weights for two different matrices

Unembedding layer maps from an embedding to a 1x|V| vector of logits

Slide adapted from Jurafsky & Martin

Language modeling head



Logits, the score vector u

One score for each of the $|V|$ possible words in the vocabulary V. Shape $1 \times |V|$.

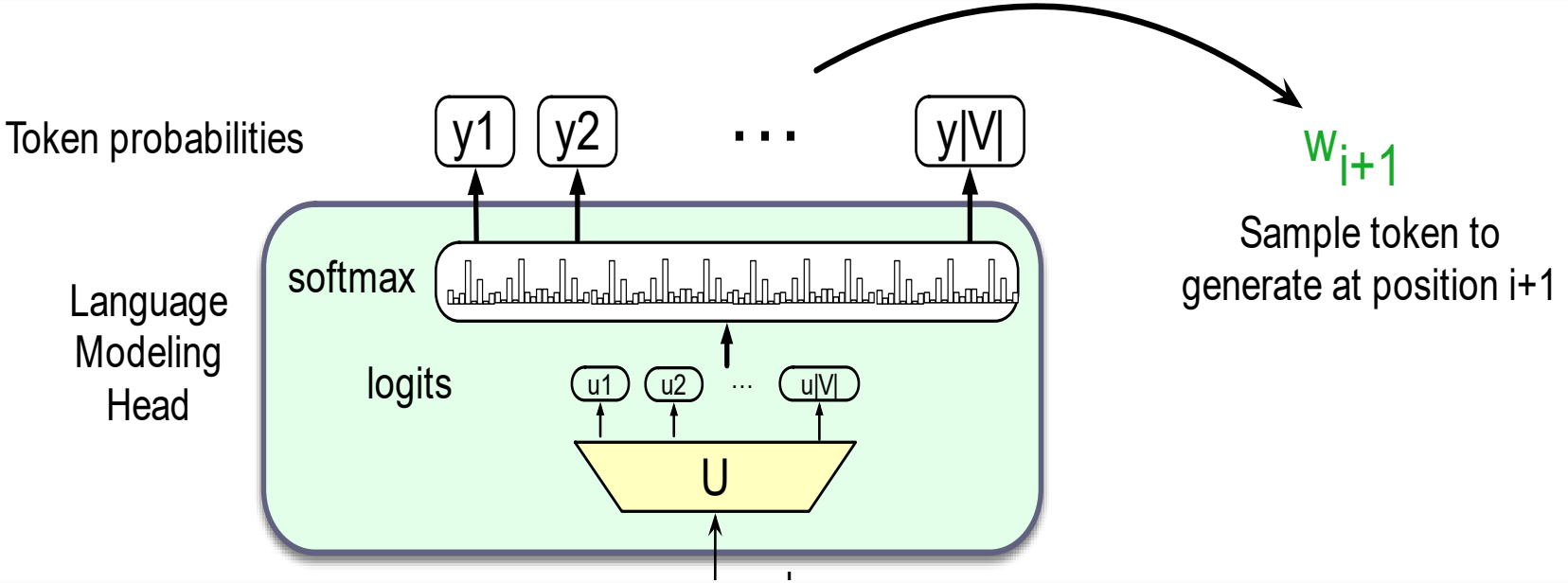
Softmax turns the logits into probabilities over vocabulary. Shape $1 \times |V|$.

$$u = h_N^L E^T$$

$$y = \text{softmax}(u)$$

Slide adapted from Jurafsky & Martin

The final transformer language model



Slide adapted from Jurafsky & Martin

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

Language models

- Remember the simple n-gram language model
  - Assigns probabilities to sequences of words
  - Generate text by sampling possible next words
  - Is trained on counts computed from lots of text
- Large language models are similar and different:
  - Assigns probabilities to sequences of words
  - Generate text by sampling possible next words
  - **Are trained by learning to guess the next word**

Slide adapted from Jurafsky & Martin

Large language models

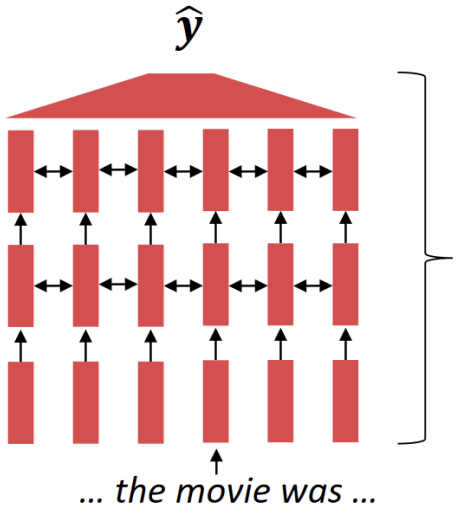
- Even through pretrained only to predict words
- Learn a lot of useful language knowledge
- Since training on a **lot** of text

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

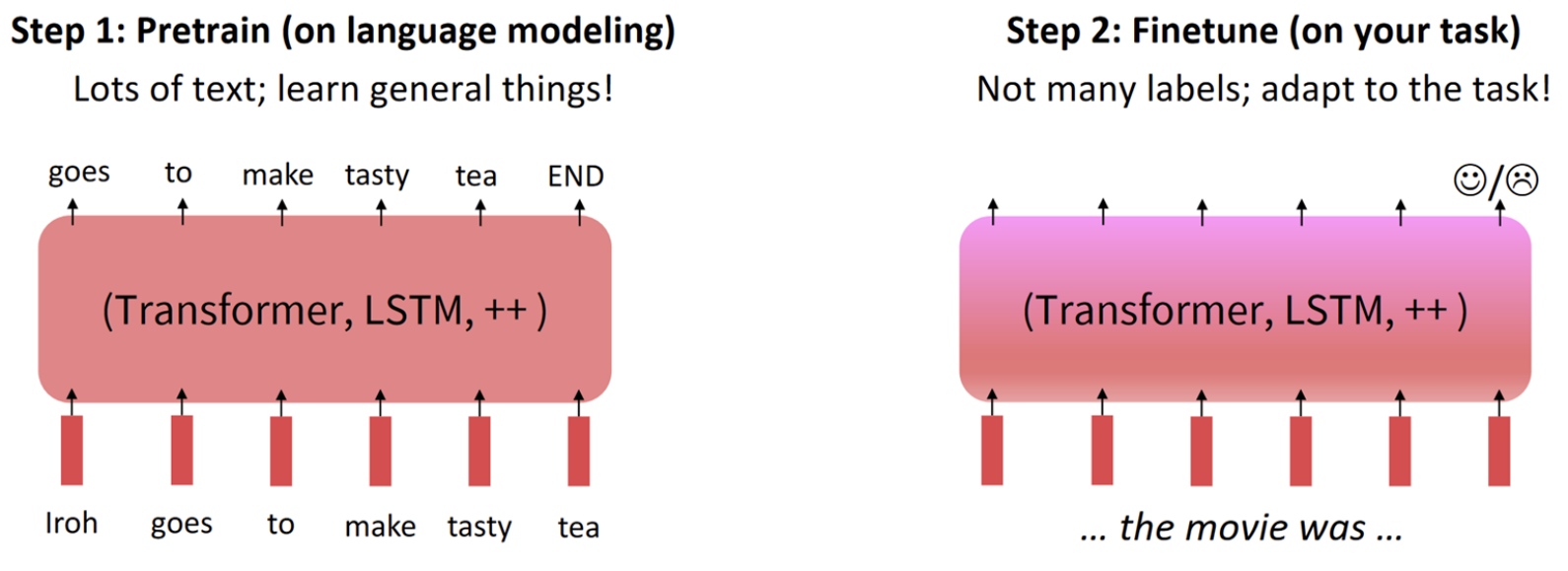
What can we learn from reconstructing the input?

- 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

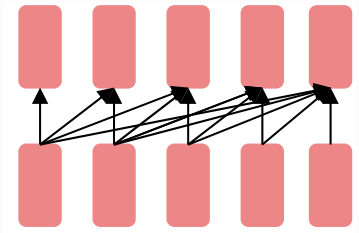
Pretraining can improve NLP applications by serving as parameter initialization.



Slide adapted from John Hewitt

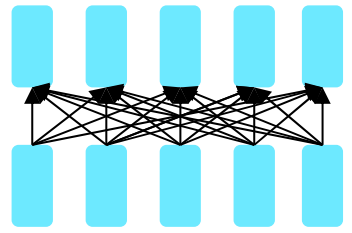
3 types of LLMs:
encoders, encoder-decoders, decoders

Three architectures for large language models



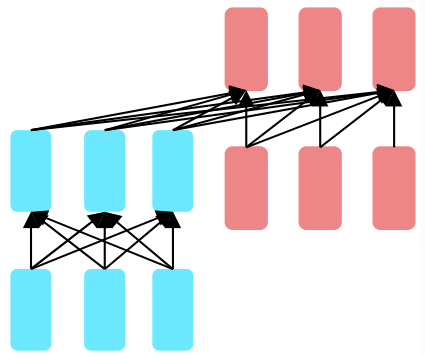
Decoders

GPT, Claude, Llama, Mixtral



Encoders

BERT family, HuBERT



Encoder-decoders

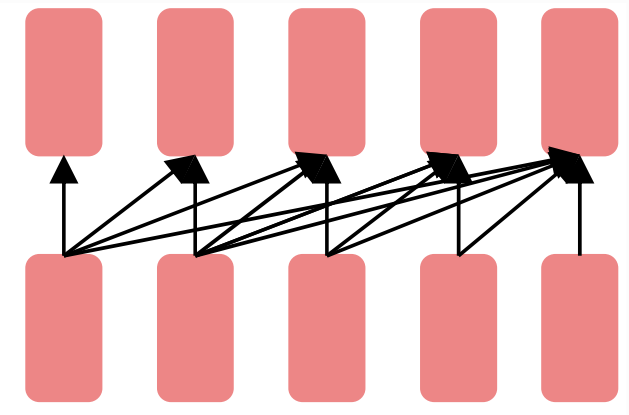
Flan-T5, Whisper

Slide adapted from Jurafsky and Martin

Decoder-only models

Also called:

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



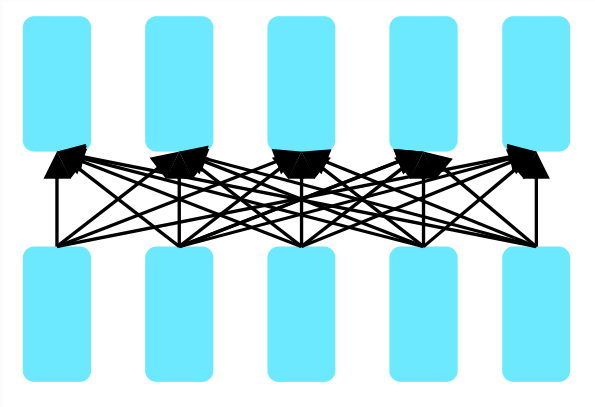
- Predict words left to right

Slide adapted from Jurafsky and Martin

Encoders

Many varieties!

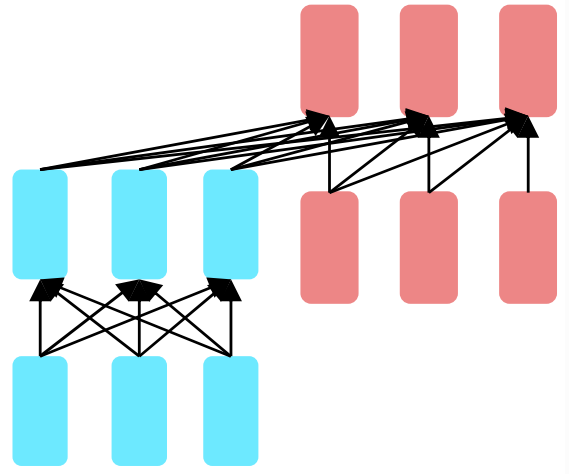
- Popular: Masked Language Models (MLMs)
- BERT family
- Trained by predicting words from surrounding words on both sides
- Are usually **finetuned** (trained on supervised data) for classification tasks.



Slide adapted from Jurafsky and Martin

Encoder-Decoders

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



Slide adapted from Jurafsky and Martin

Decoder LLMs

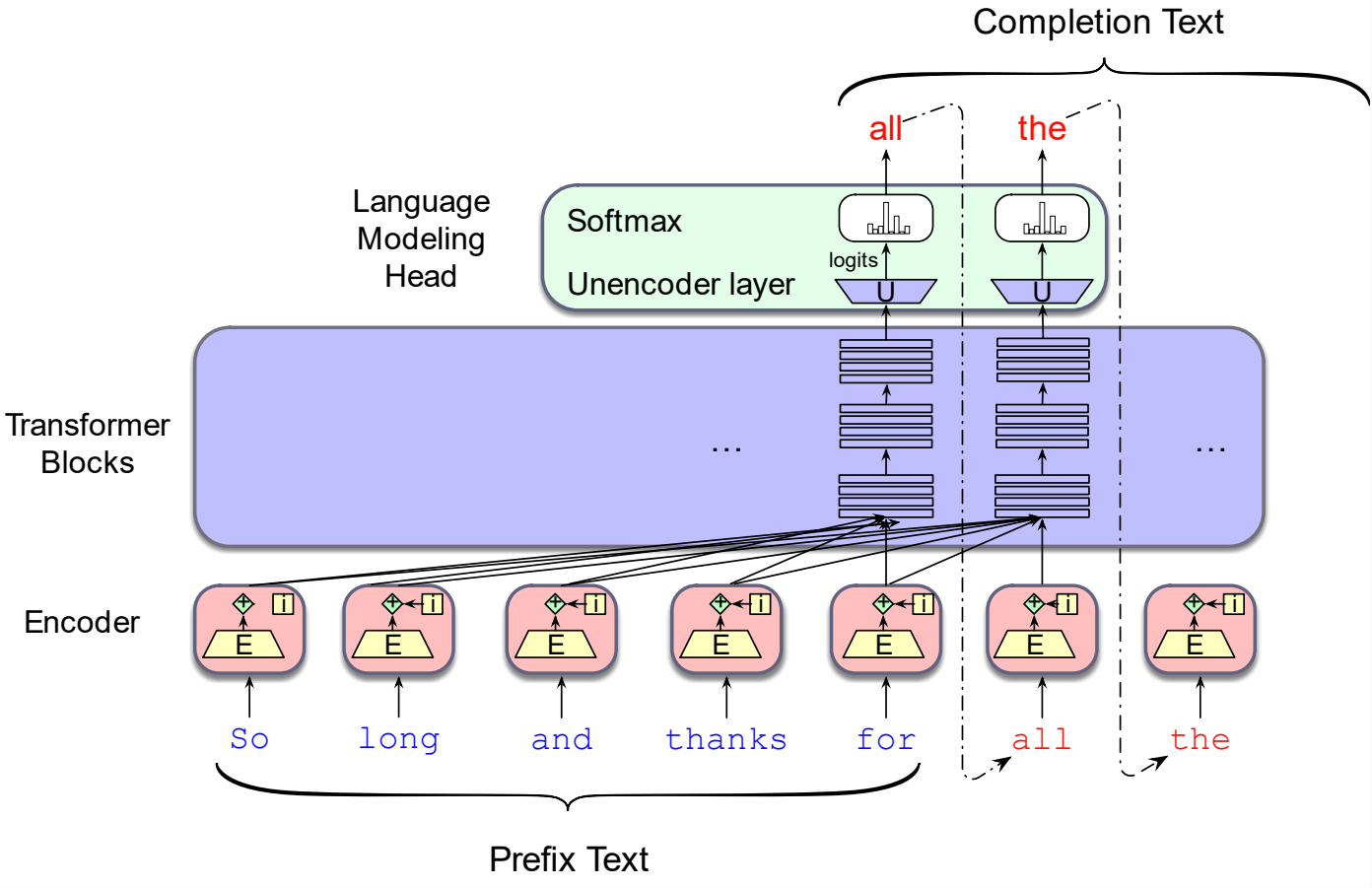
Decoder-only models can handle many tasks

- 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

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.

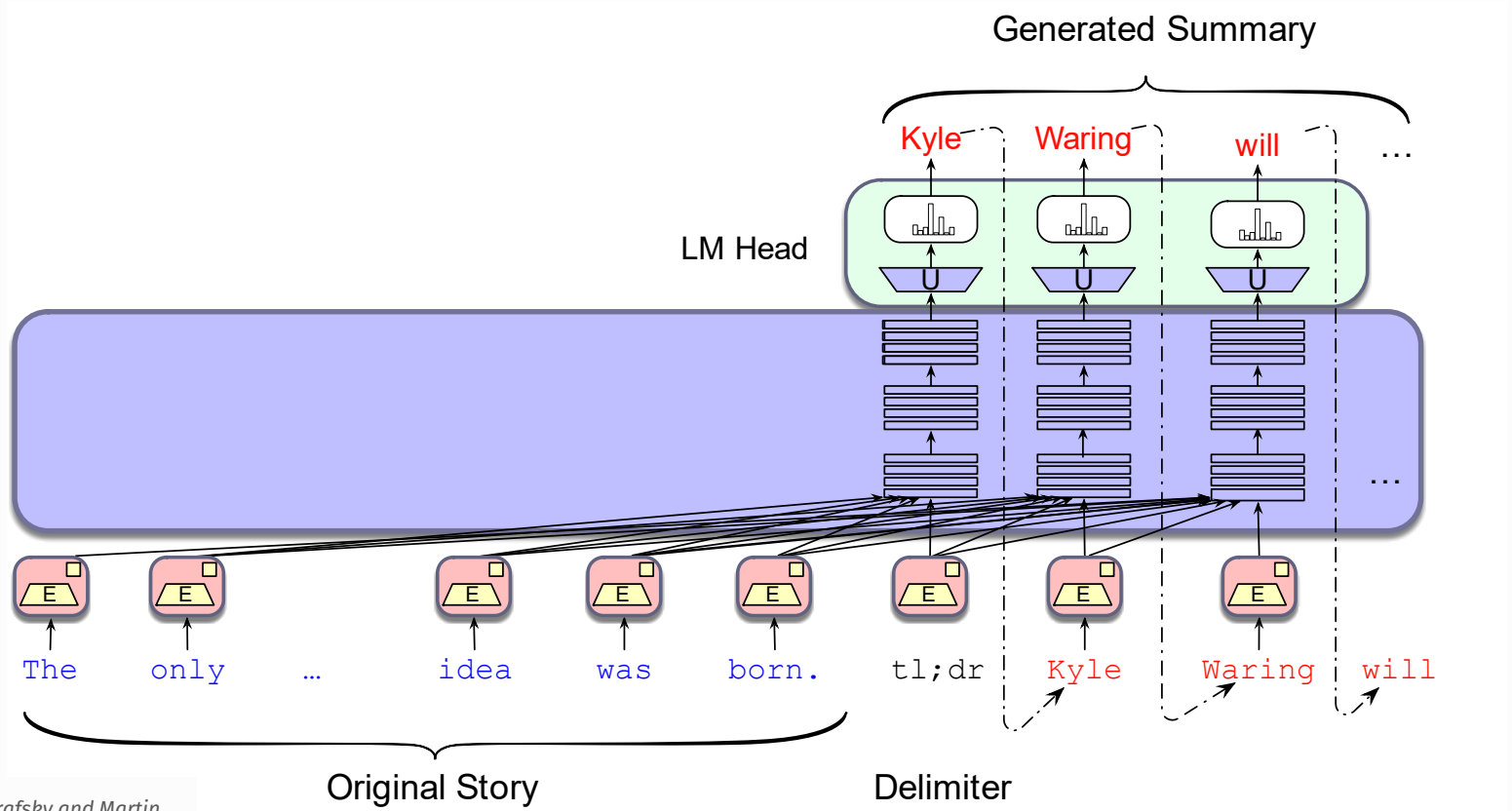
According to Boston.com, it all started a few weeks ago, when Waring and his wife were shoveling deep snow from their yard in Manchester-by-the-Sea, a coastal suburb north of Boston. He joked about shipping the stuff to friends and family in warmer states, and an idea was born. [...]

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

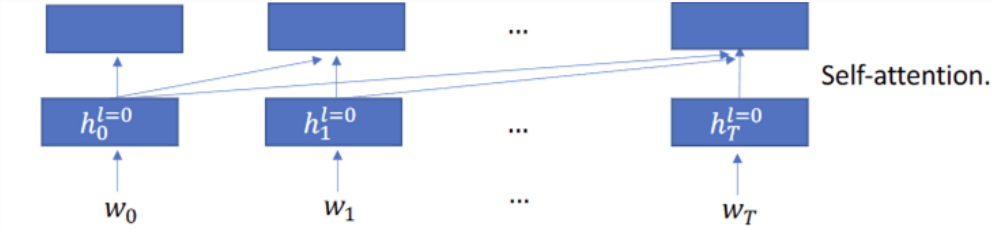
Pretraining decoder LLMs

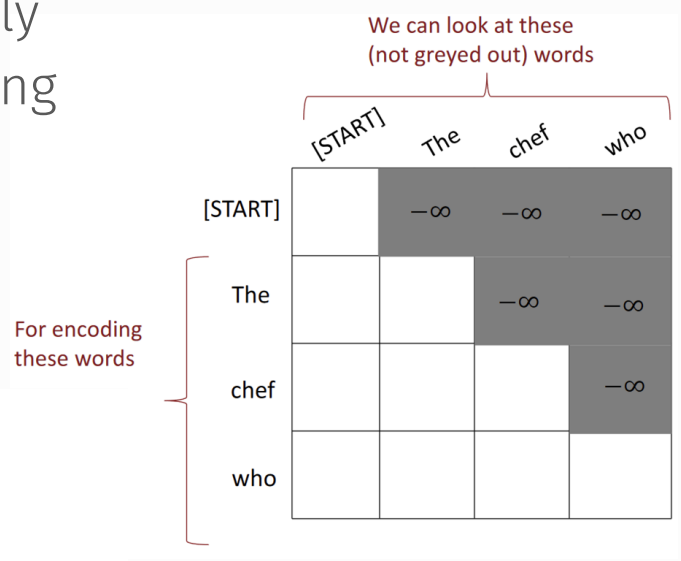
- Take a corpus and ask the model to predict the next word!
- Train the model using gradient descent to minimize the error
- Same loss function as other neural models: cross-entropy loss
- Move the weights in the direction that assigns a higher probability to the true next word

Slide adapted from Jurafsky and Martin

Decoding: apply a “causal mask” for self-attention

- To do auto-regressive LM, we need to apply a “causal” mask to self-attention, forbidding it from getting future context.
- At timestep t, we set a_i = 0 for $i > t$

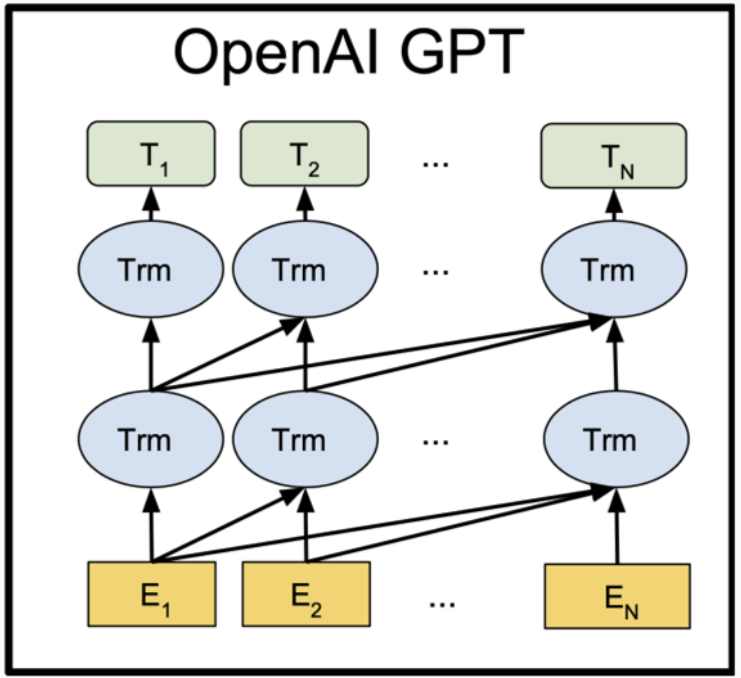




Slide adapted from Tianxing He, John Hewitt

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

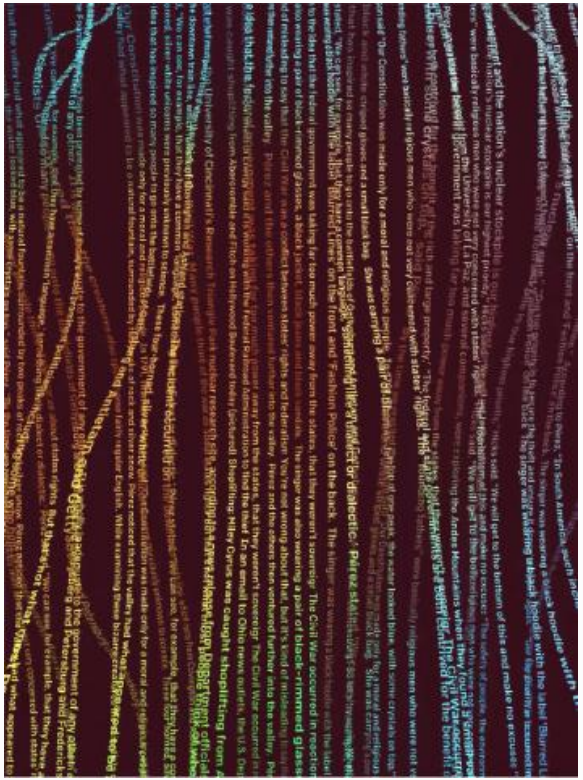
- 2018's GPT was a big success in pretraining a decoder!
- Transformer decoder with 12 layers, 117M parameters.
- 768-dimensional hidden states, 3072-dimensional feed-forward hidden layers.
- Trained on BooksCorpus: over 7000 unique books.
    - Contains long spans of contiguous text, for learning long-distance dependencies.



Slide adapted from John Hewitt, David Mortensen

GPT-2, GPT-3, GPT-4, GPT-5 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

Slide adapted from Jurafsky and Martin

Random sampling

$\mathrm{i} \leftarrow 1$

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

while w_i != EOS

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

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

Slide adapted from Jurafsky and Martin

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

Slide adapted from Jurafsky and Martin

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

Slide adapted from Jurafsky and Martin

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 $p(w_t \mid w_{<t})$
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.

Slide adapted from Jurafsky and Martin

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.

Slide adapted from Jurafsky and Martin

Temperature sampling

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

Instead of

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

We do

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

A lower τ pushes high-probability words higher and low probability word lower due to the way softmax works

Slide adapted from Jurafsky and Martin

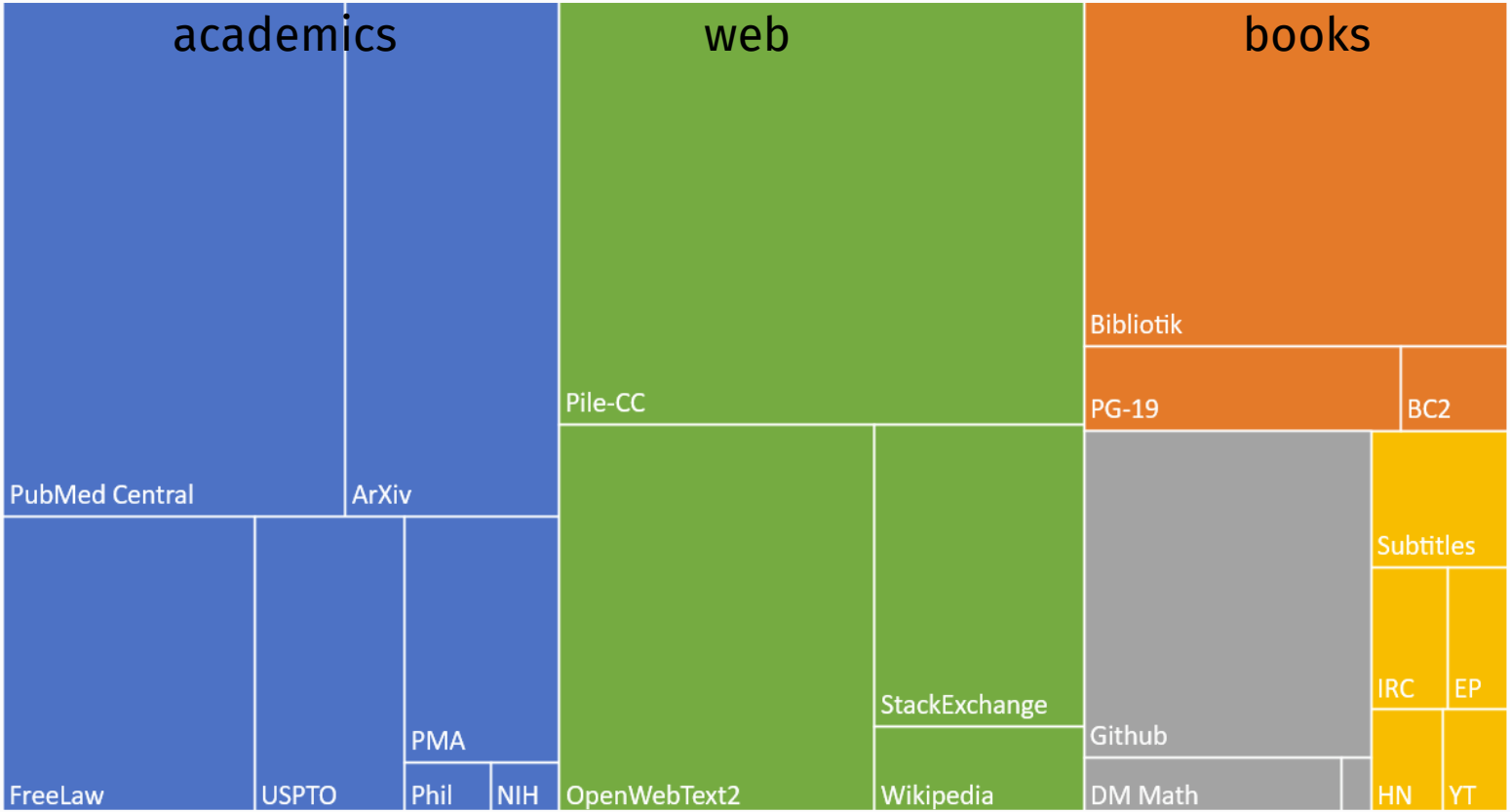
Pretraining data and harms of LLMs

LLMs are mainly trained on the web

- Common crawl, snapshots of the entire web produced by the non- profit Common Crawl with billions of pages
- Colossal Clean Crawled Corpus (C4; Raffel et al. 2020), 156 billion tokens of English, filtered
- What's in it? Mostly patent text documents, Wikipedia, and news sites

Slide adapted from Jurafsky and Martin

The Pile: a pretraining corpus



dialog

Slide adapted from Jurafsky and Martin

Big idea

- Text contains enormous amounts of knowledge
- Pretraining on lots of text with all that knowledge is what gives language models their ability to do so much

Slide adapted from Jurafsky and Martin

But there are problems with scraping from the web

- Copyright: much of the text in these datasets is copyrighted
  - Not clear if fair use doctrine in US allows for this use
  - This remains an open legal question
- Data consent
  - Website owners can indicate they don't want their site crawled
- Privacy:
  - Websites can contain private IP addresses and phone numbers

Slide adapted from Jurafsky and Martin

Harms from LLMs

Hallucination

What Can You Do When A.I. Lies About You?

People have little protection or recourse when the technology creates and spreads falsehoods about them.

Air Canada loses court case after its chatbot hallucinated fake policies to a customer

The airline argued that the chatbot itself was liable. The court disagreed.

Copyright

Authors Sue OpenAI Claiming Mass Copyright Infringement of Hundreds of Thousands of Novels

Privacy

Slide adapted from Jurafsky and Martin

Harms from LLMs

Toxicity and abuse

The New AI-Powered Bing Is Threatening Users.

Cleaning Up ChatGPT Takes Heavy Toll on Human Workers

Contractors in Kenya say they were traumatized by effort to screen out descriptions of violence and sexual abuse during run-up to OpenAI's hit chatbot

Misinformation

Chatbots are generating false and misleading information about U.S. elections

Slide adapted from Jurafsky and Martin

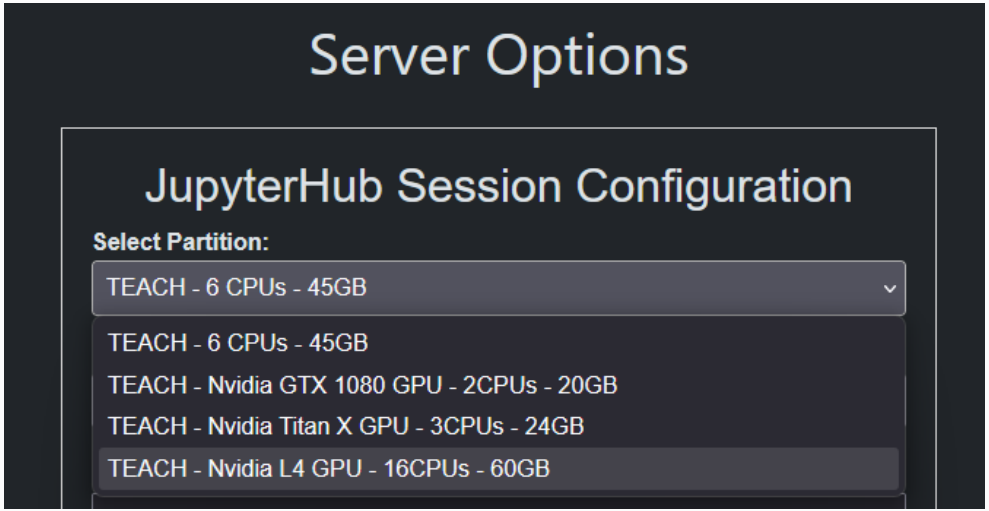
Conclusion

- 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
- Harms from LLMs include hallucinating false information, leaking private information from training data, generating abuse and misinformation

Coding activity

Notebook: finetune GPT-2 on Shakespeare

- Click on this nbgitpuller link or find the link on the course website
- **Important difference from normal:** Open a '**TEACH – Nvidia L4 GPU – 16 CPUs – 60GB**' server



- Open **session14_gpt2_shakespeare.ipynb**