Transformers – Part 3

Summary of Chapter 10 from
Speech and Language Processing,
Jurafsky and Martin, Feb. 3, 2024 draft
Michael Wollowski

Language modeling head

- Language models, from a simple n-gram model to the feedforward and RNN language models are word predictors.
- Given a context of words, they assign a probability to each possible next word.
- In Transformer architectures, there is a **language modeling head** designed for this purpose.

Language modeling head

- If the preceding context is *"Thanks for all the"* and we want to know the likelihood of the next word being *"fish"* we would compute:
  $P(fish|Thanks\ for\ all\ the)$
- Language models assign such a conditional probability to every possible next word.
- In other words, they give a distribution over the entire vocabulary.
- An n-gram language models computes such probabilities given the $n-1$ prior words as context.
- For transformer language models, the context can be quite large: up to 2048 or even 4096 tokens.

Language modeling head

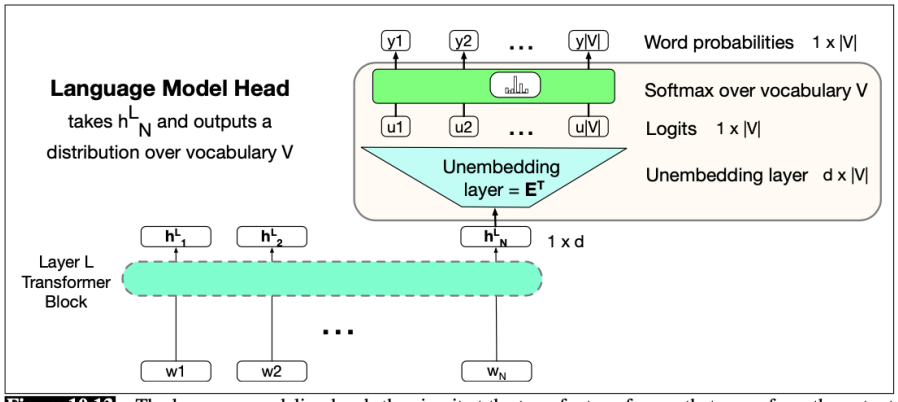
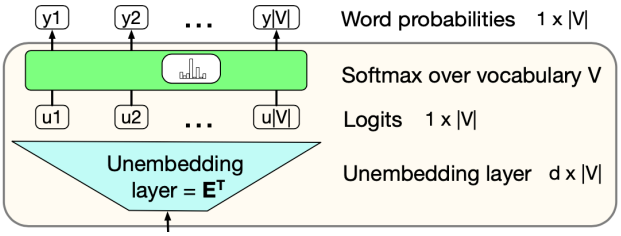


Figure 10.13 The language modeling head: the circuit at the top of a transformer that maps from the output embedding for token N from the last transformer layer ($\mathbf{h}^L_N$) to a probability distribution over words in the vocabulary V.

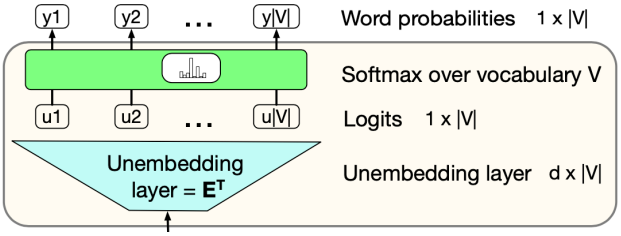
- The language modeling head takes the output of the final transformer layer from the last token N and use it to predict the upcoming word at position $N + 1$.

Language modeling head



- The first module is a linear layer.
- It projects from the output h^L_N to the logit vector, or score vector.
- The score vector has a single score for each of the words in the vocabulary *V* .
- Commonly this matrix is the transpose of the embedding matrix E.
- The transpose E^T is called the *unembedding* layer, because it performs the reverse of the embedding that occurs at the input stage of the transformer.

Language modeling head

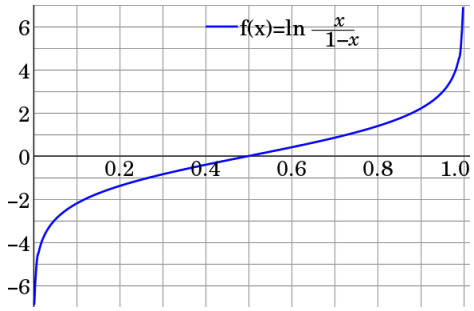


- A softmax layer turns the logits **u** into the probabilities y over the vocabulary.
- Sidebar:

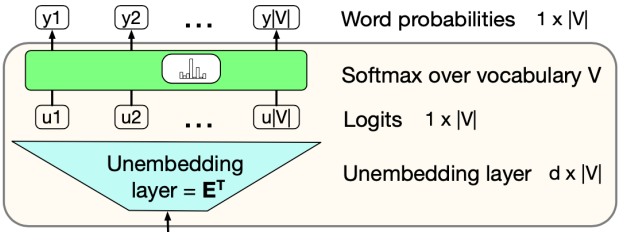
  Probability: *p* ->

  Odds: *p/(1-p)* ->

  logits: *log(p/1-p)*



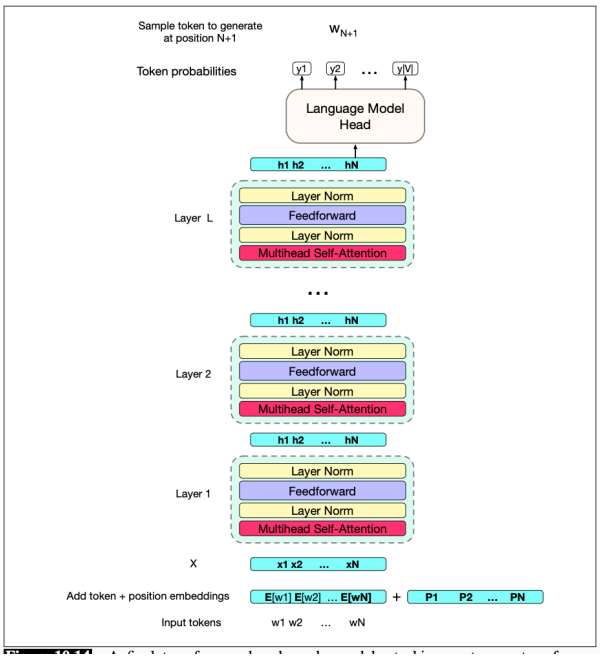
Language modeling head



- Use the word probabilities to generate text.
- Sample a word from these probabilities *y*.
- For example, sample the highest probability word, called 'greedy' decoding.
- Recall the article entitled "What kind of Mind does ChatGPT have?"
- Whatever entry y_k we choose from the probability vector y, we generate the word that has that index *k*.

Transformer Decoder-only Model

- The figure shows the total stacked architecture.
- The input to the first transformer block is represented as X.
- They are the *N* indexed word embeddings + position embeddings: E[w] + P
- The input to all the other layers is the output H from the prior layer.



Transformer Decoder-only Model

- The language model on the prior slide is called a *decoder-only* model.
- This is because this model constitutes roughly half of the encoder-decoder model for transformers.
- Confusingly, the original introduction of the transformer had an encoder-decoder architecture.
- It was only later that the standard paradigm for causal language model was defined by using only the decoder part of this original architecture.

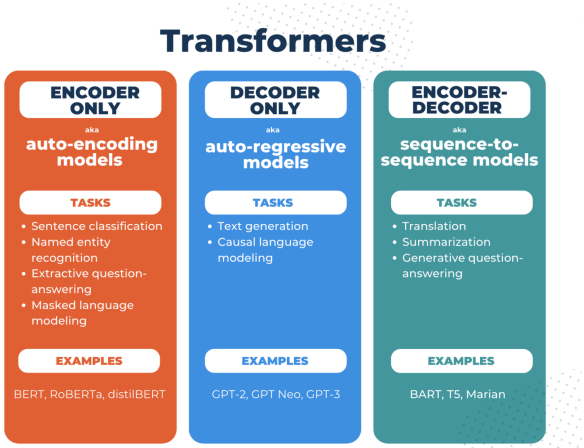
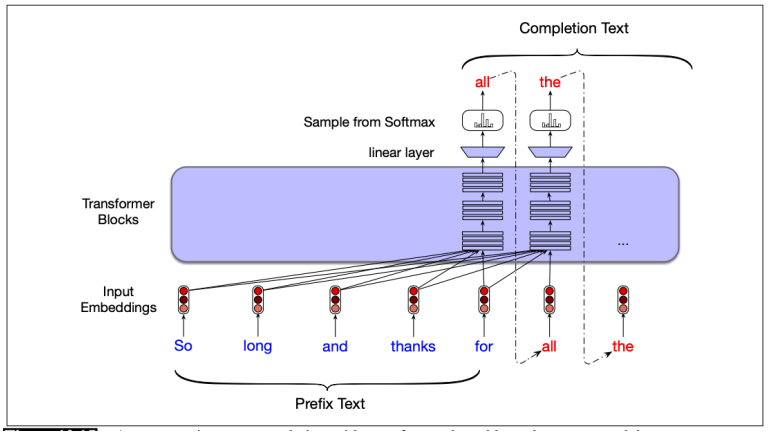


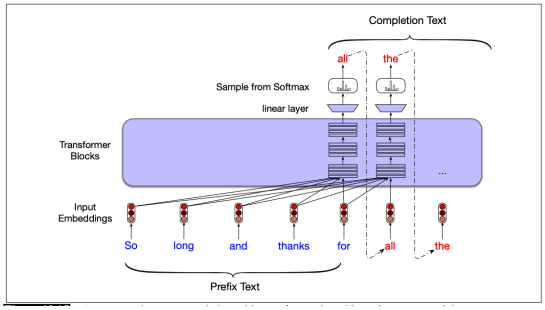
Image source: https://www.comet.com/site/blog/explainable-ai-for-transformers/

Text Completion

- A language model is given a text prefix and is asked to generate a possible completion.



Text Completion



- As the generation process proceeds, the model has direct access to the priming context as well as to all of its own subsequently generated outputs (at least as much as fits in the large context window).
- This ability to incorporate the entirety of the earlier context and generated outputs at each time step is the key to the power of large language models built from transformers.

Question Answering

- Why should we care about predicting upcoming words?
- Many practical NLP tasks can be cast as word prediction.
- Consider the task of answering simple questions.
- The system is given some question and must give a textual answer.
- We can cast the task of question answering as word prediction.
- Consider the question: Who wrote the book ‘‘The Origin of Species"?

Question Answering

- We may ask a language model to compute

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

- We then look at which words *w* with high probabilities, we might expect to see that *Charles* is very likely.
- If we choose *Charles* and continue and ask

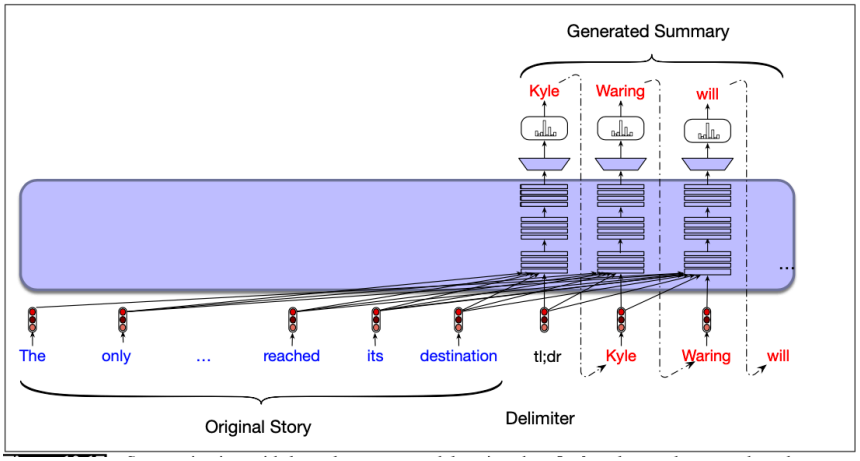
  $P(w|Q:$ *Who wrote the book "The Origin of Species"? A: Charles*$)$

- We might now see that *Darwin* is the most probable word, and select it.

Text Summarization

- In text summarization, we take a long text and produce a summary of it.
- We can cast summarization as language modeling.
- Give an LLM a text, followed by a token like **tl;dr;**
- This token is short for 'too long; did not read'
- We can perform conditional generation as follows:
  - Give the language model the text and token
  - Ask it to generate words, one by one
  - Take the entire response as a summary.

Text Summarization



Text Summarization

A human-produced summary.

Original Article

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. His business slogan: "Our nightmare is your dream!" At first, ShipSnowYo sold snow packed into empty 16.9-ounce water bottles for $19.99, but the snow usually melted before it reached its destination...

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.

Text Summarization

- Transformers succeed at this task because of their ability of self-attention to incorporate information from the large context windows.
- The model has access to the original article as well as to the newly generated text throughout the process.
- Which words shall we generate at each step?
- A simple way is to always generate the most likely word given the context.
- This is called *greedy decoding*.
- It will make a choice that is locally optimal.
- It may not be globally optimal.

Text Summarization

- A major problem with greedy decoding is that the words it chooses are predictable.
- The resulting text is generic and often quite repetitive.
- People prefer text which has been generated by more sophisticated methods that introduce a bit more diversity into the generations.

Self-supervised training algorithm for Transformers

- Transformers are trained on a corpus of text.
- At each time step *t*, we ask the model to predict the next word.
- We call such a model *self-supervised,* because the natural sequence of words is its own supervision.
- We simply train the model to minimize the error in predicting the true next word in the training sequence.

Self-supervised training algorithm for Transformers

- At each word position *t* of the input, the model takes as input the correct sequence of tokens $w_{1:t}$
- It uses them to compute a probability distribution over possible next words so as to compute the model's loss for the next token w_{t+1}
- Then we move to the next word.
- We ignore what the model predicted for the next word and instead use the correct sequence of tokens $w_{1:t+1}$ to estimate the probability of token w_{t+2}
- We always give the model the correct history sequence to predict the next word.
- This is called *teacher forcing*.

Self-supervised training algorithm for Transformers

- At each step, given all the preceding words, the final transformer layer produces an output distribution over the entire vocabulary.

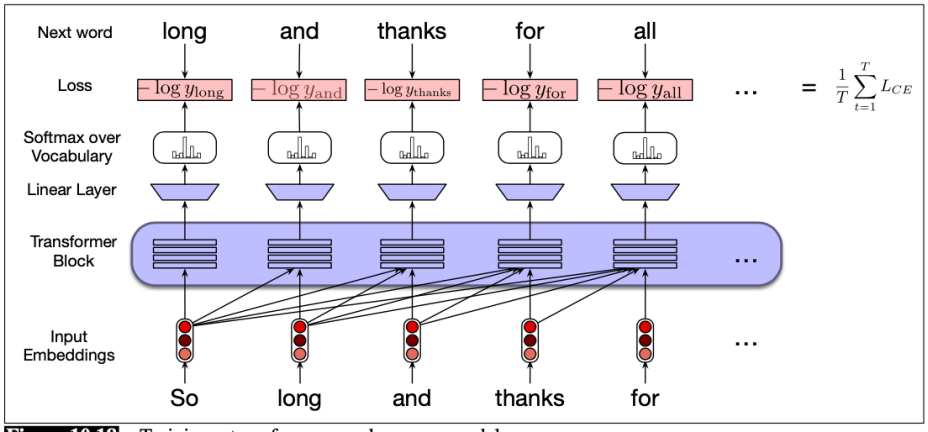


Figure 10.18 Training a transformer as a language model.

Self-supervised training algorithm for Transformers

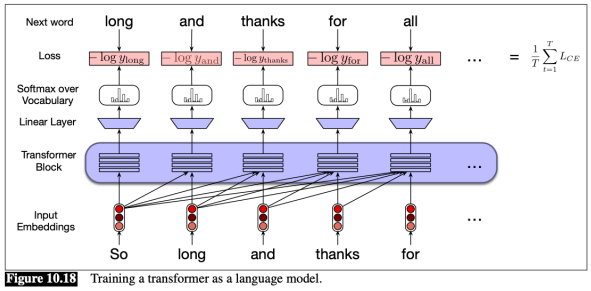


Figure 10.18 Training a transformer as a language model.

- During training, the probability assigned to the correct word is used to calculate the loss for each item in the sequence.
- The weights in the network are adjusted to minimize the average loss over the training sequence via gradient descent.

Training corpora for LLMs

- Large language models are mainly trained on text scraped from the web, augmented by more carefully curated data.
- Since those training corpora are so large, they are likely to contain many natural examples that can be helpful for NLP tasks:
  - question and answer pairs (for example from FAQ lists),
  - translations of sentences between various languages,
  - documents together with their summaries, and so on.

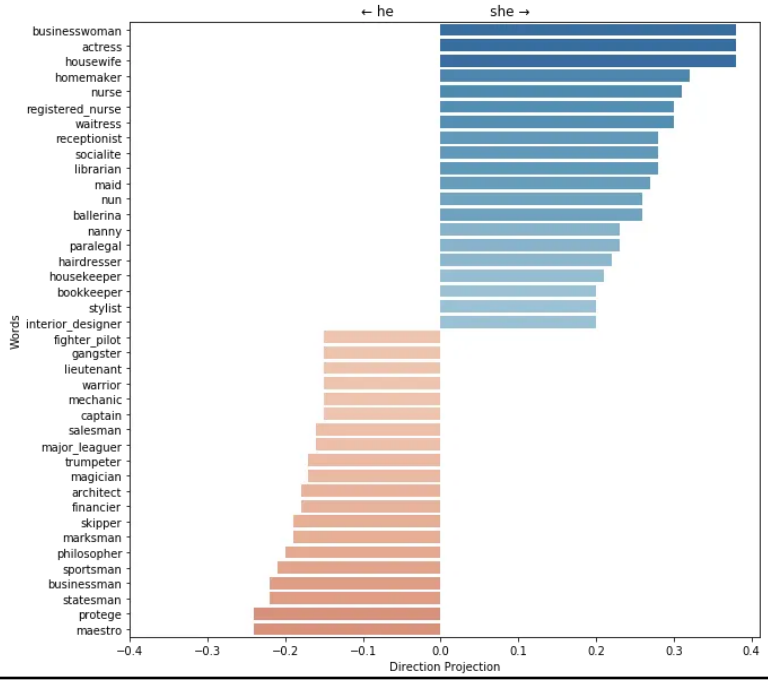
Training corpora for LLMs

- Web text is usually taken from corpora of automatically-crawled web pages like the *common crawl*.
- It is a series of snapshots of the entire web produced by the non-profit Common Crawl that each have billions of webpages.
- Various cleanups of common crawl data exist.
- One is *Colossal Clean Crawled Corpus* (C4)
- It is a corpus of 156 billion tokens of English that is filtered in various ways.
- Filtering includes:
  - Removing duplicated data,
  - removing non-natural language like code,
  - sentences with offensive words from a blocklist.

Training corpora for LLMs

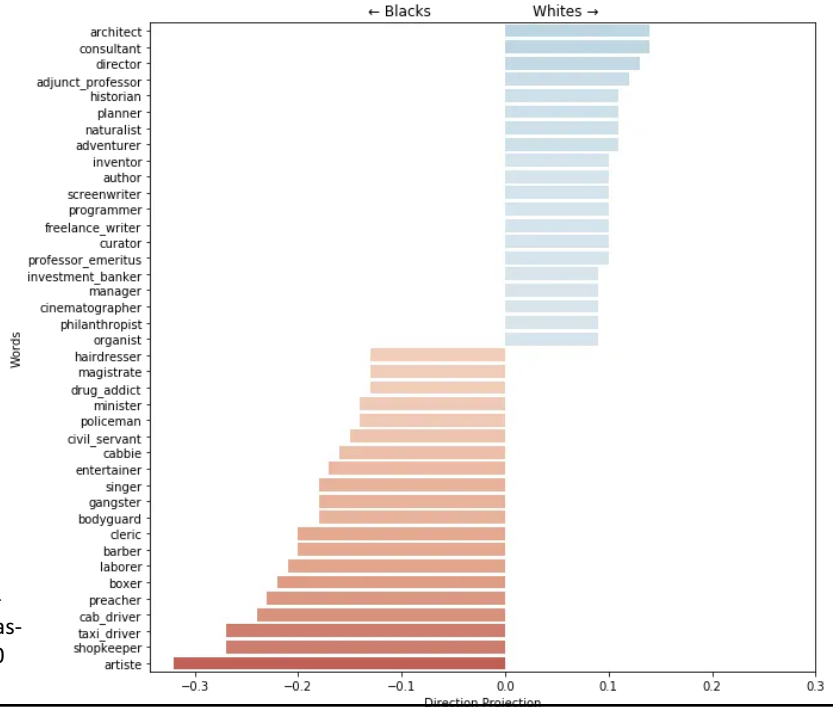
- What is in the training data?
- An analysis suggests that in large part it's:
  - patent text documents,
  - Wikipedia, and
  - news sites
- Wikipedia plays a role in lots of language model training, as do corpora of books.
- The GPT3 models are trained mostly on the web (429 billion tokens), some text from books (67 billion tokens) and Wikipedia (3 billion tokens).

Bias in Embeddings



Source:
https://medium.com/institute-for-applied-computational-science/bias-in-nlp-embeddings-b1dabb8bbe20

Bias in Embeddings



Source: https://medium.com/institute-for-applied-computational-science/bias-in-nlp-embeddings-b1dabb8bbe20

Scaling Laws

- The performance of large language models has shown to be mainly determined by 3 factors:
  - model size (the number of parameters),
  - dataset size (the amount of training data), and
  - the number of iterations used for training.
- We can improve a model by adding parameters (adding more layers or having wider contexts or both), by training on more data, or by training for more iterations.
- The relationships between these factors and performance are known as *scaling laws*.