SIT330-770: Natural Language Processing

Week 8 – Large Language Models

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env

reda.bouadjenek@deakin.edu.au

1

SIT330-770: Natural Language Processing

Week 8. 1 - Introduction to Large Language Models

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env

2

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**

3

Large language models

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

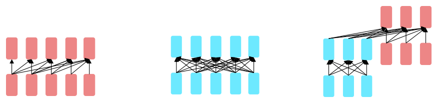
4

Large language models

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

5

Three architectures for large language models



Decoders GPT, Claude, Llama, Mixtral

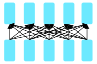
Encoders BERT family, HuBERT

Encoder-decoders Flan-T5, Whisper

6

Encoders

- Many varieties!
  - Popular: Masked Language Models (MLMs)
  - BERT family

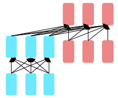


  - Trained by predicting words from surrounding words on both sides
  - Are usually **finetuned** (trained on supervised data) for classification tasks.

7

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)



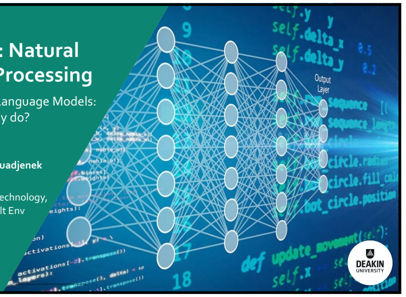
8

SIT330-770: Natural Language Processing

Week 8. 2 - Large Language Models: What tasks can they do?

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env



9

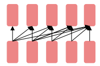
Big idea

Many tasks can be turned into tasks of predicting words!

10

This lecture: decoder-only models

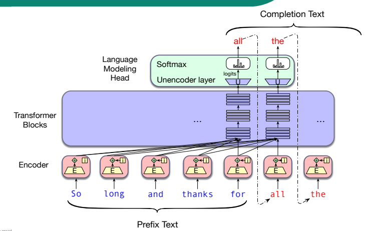
- Also called:
  - Causal LLMs
  - Autoregressive LLMs
  - Left-to-right LLMs



  - Predict words left to right

11

Conditional Generation: Generating text conditioned on previous text!



12

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

13

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:
     $P(w|\texttt{Q: Who wrote the book ``The Origin of Species"? A:})$
  3. And iterate:
     $P(w|\texttt{Q: Who wrote the book ``The Origin of Species"? A: Charles})$

14

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.

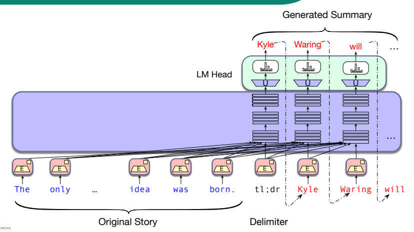
According to Boston.com, it all started a few weeks ago, when Waring and his wife were shoveling deep snow from their yard in Manchester-by-the-Sea, a coastal suburb north of Boston. He joked about shipping the stuff to friends and family in warmer states, and an idea was born. [...]

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.

15

LLMs for summarization (using tl;dr)



16

SIT330-770: Natural Language Processing

Week 8. 3 - Sampling for LLM Generation

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env

17

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

18

Random sampling

$$
\begin{aligned}
& i \leftarrow 1 \\
& w_i \sim p(w) \\
& \textbf{while } w_i \text{ != EOS} \\
& \quad i \leftarrow i+1 \\
& \quad w_i \sim p(w_i \mid w_{<i})
\end{aligned}
$$

19

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

20

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

21

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

22

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 $P(w_t \mid w_{<t})$, the top-p vocabulary V (p) is the smallest set of words such that

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

23

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.

24

Temperature sampling

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

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

- We do

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

25

Temperature sampling

$0 \le \tau \le 1$

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

- Why does this work?
  - When τ is close to 1 the distribution doesn't change much.
  - The lower τ is, the larger the scores being passed to the softmax
  - Softmax pushes high values toward 1 and low values toward 0.
  - Large inputs 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

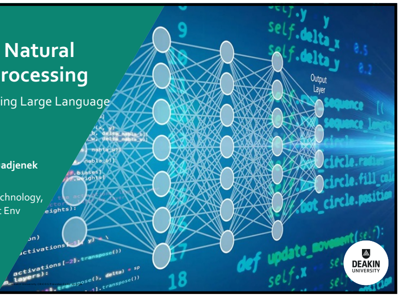
26

SIT330-770: Natural Language Processing

Week 8. 4 - Pretraining Large Language Models: Algorithm

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env



27

Pretraining

- The big idea that underlies all the amazing performance of language models
- First **pretrain** a transformer model on enormous amounts of text
- Then **apply** it to new tasks.

28

Self-supervised training algorithm

- We just train them to predict the next word!
  1. Take a corpus of text
  2. At each time step t
     - I. ask the model to predict the next word
     - II. train the model using gradient descent to minimize the error in this prediction
  - "**Self-supervised**" because it just uses the next word as the label!

29

Intuition of language model training: loss

- Same loss function: **cross-entropy loss**
  - We want the model to assign a high probability to true word w
  - = want loss to be high if the model assigns too low a probability to w
- CE Loss: The negative log probability that the model assigns to the true next word w
  - If the model assigns too low a probability to w
  - We move the model weights in the direction that assigns a higher probability to w

30

Cross-entropy loss for language modeling

- **CE loss**: difference between the correct probability distribution and the predicted distribution

$$L_{CE} = -\sum_{w \in V} \mathbf{y}_t[w] \log \hat{\mathbf{y}}_t[w]$$

- The correct distribution $\mathbf{y}_t$ knows the next word, so is 1 for the actual next word and 0 for the others.
- So in this sum, all terms get multiplied by zero except one: the logp the model assigns to the correct next word, so:

$$L_{CE}(\hat{\mathbf{y}}_t, \mathbf{y}_t) = -\log \hat{\mathbf{y}}_t[w_{t+1}]$$

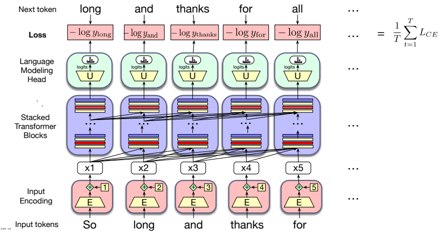
31

Teacher forcing

- At each token position t, model sees correct tokens $w_{1:t}$,
  - Computes loss (–log probability) for the next token w_{t+1}
- At next token position t+1 we ignore what model predicted for w_{t+1}
  - Instead we take the **correct** word w_{t+1}, add it to context, move on

32

Training a transformer language model



33

SIT330-770: Natural Language Processing

Week 8. 5 - Pretraining data for LLMs

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env

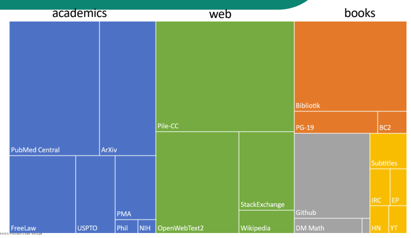
34

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

35

The Pile: a pretraining corpus



36

Filtering for quality and safety

- Quality is subjective
  - Many LLMs attempt to match Wikipedia, books, particular websites
  - Need to remove boilerplate, adult content
  - Deduplication at many levels (URLs, documents, even lines)
- Safety also subjective
  - Toxicity detection is important, although that has mixed results
  - Can mistakenly flag data written in dialects like African American English

37

What does a model learn from pretraining?

- There are canines everywhere! One dog in the front room, and two **dogs**
- It wasn't just big it was **enormous**
- The author of "A Room of One's Own" is Virginia **Woolf**
- The doctor told me that **he**
- The square root of 4 is **2**

38

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

39

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

40

SIT330-770: Natural Language Processing

Week 8. 6 - Finetuning

Dr. Mohamed Reda Bouadjenek

School of Information Technology,
Faculty of Sci Eng & Built Env

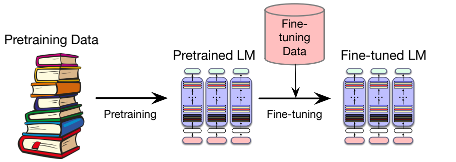
41

Finetuning for adaptation to new domains

- What happens if we need our LLM to work well on a domain it didn't see in pretraining?
- Perhaps some specific medical or legal domain?
- Or maybe a multilingual LM needs to see more data on some language that was rare in pretraining?

42

Finetuning



43

"Finetuning" means 4 different things

- We'll discuss 1 here, and 3 in later lectures
- In all four cases, finetuning means:
  - **taking a pretrained model and further adapting some or all of its parameters to some new data**

44

1. Finetuning as "continued pretraining" on new data

- Further train all the parameters of model on new data
  - using the same method (word prediction) and loss function (cross-entropy loss) as for pretraining.
  - as if the new data were at the tail end of the pretraining data
- Hence sometimes called **continued pretraining**

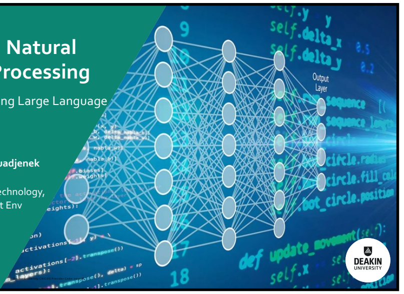
45

SIT330-770: Natural Language Processing

Week 8. 7 - Evaluating Large Language Models

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env



46

Perplexity

- Just as for n-gram grammars, we use perplexity to measure how well the LM predicts unseen text
- The perplexity of a model θ on an unseen test set is the **inverse probability that θ assigns to the test set, normalized by the test set length.**
- For a test set of *n* tokens $w_{1:n}$ the perplexity is :

$$\text{Perplexity}_\theta(w_{1:n}) = P_\theta(w_{1:n})^{-\frac{1}{n}} = \sqrt[n]{\frac{1}{P_\theta(w_{1:n})}}$$

47

Why perplexity instead of raw probability of the test set?

- Probability depends on size of test set
  - Probability gets smaller the longer the text
  - Better: a metric that is **per-word**, normalized by length
- **Perplexity** is the inverse probability of the test set, normalized by the number of words

(The inverse comes from the original definition of perplexity from cross-entropy rate in information theory)

Probability range is [0,1], perplexity range is [1,∞]

48

Perplexity

- The higher the probability of the word sequence, the lower the perplexity.
- Thus the lower the perplexity of a model on the data, the better the model.
- **Minimizing perplexity is the same as maximizing probability**

Also: perplexity is sensitive to length/tokenization so best used when comparing LMs that use the same tokenizer.

49

Many other factors that we evaluate, like:

- **Size**

  Big models take lots of GPUs and time to train, memory to store
- **Energy usage**

  Can measure kWh or kilograms of CO2 emitted
- **Fairness**

  Benchmarks measure gendered and racial stereotypes, or decreased performance for language from or about some groups.

50

SIT330-770: Natural Language Processing

Week 8. 8 - Dealing with Scale

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env

51

Scaling Laws

- LLM performance depends on
  - Model size: the number of parameters not counting embeddings
  - Dataset size: the amount of training data
  - Compute: Amount of compute (in FLOPS or etc
- Can improve a model by adding parameters (more layers, wider contexts), more data, or training for more iterations
- The performance of a large language model (the loss) scales as a power-law with each of these three

52

Scaling Laws

- Loss L as a function of # parameters N, dataset size D, compute budget C (if other two are held constant)

$$L(N) = \left(\frac{N_c}{N}\right)^{\alpha_N}$$

$$L(D) = \left(\frac{D_c}{D}\right)^{\alpha_D}$$

$$L(C) = \left(\frac{C_c}{C}\right)^{\alpha_C}$$

- Scaling laws can be used early in training to predict what the loss would be if we were to add more data or increase model size.

53

Number of non-embedding parameters N

$$N \approx 2\, d\, n_{\text{layer}}(2\, d_{\text{attn}} + d_{\text{ff}})$$
$$\approx 12\, n_{\text{layer}}\, d^2$$
$$(\text{assuming } d_{\text{attn}} = d_{\text{ff}}/4 = d)$$

- Thus GPT-3, with n = 96 layers and dimensionality d = 12288, has 12 × 96 × 122882 ≈ 175 billion parameters.

54

KV Cache

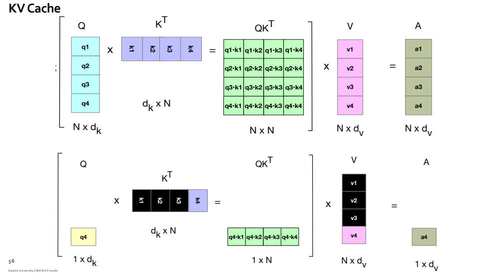
- In training, we can compute attention very efficiently in parallel:

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{QK}^\intercal}{\sqrt{d_k}}\right)\mathbf{V}$$

- But not at inference! We generate the next tokens **one at a time!**
- For a new token x, need to multiply by W^Q, W^K, and W^V to get query, key, values
- But don't want to **recompute** the key and value vectors for all the prior tokens $x_{<i}$
- Instead, store key and value vectors in memory in the KV cache, and then we can just grab them from the cache

55

KV Cache



56

Parameter-Efficient Finetuning

- Adapting to a new domain by continued pretraining (finetuning) is a problem with huge LLMs.
  - Enormous numbers of parameters to train
  - Each pass of batch gradient descent has to backpropagate through many many huge layers.
  - Expensive in processing power, in memory, and in time.
- Instead, **parameter-efficient fine tuning** (PEFT)
  - Efficiently select a subset of parameters to update when finetuning.
  - E.g., freeze some of the parameters (don't change them),
  - And only update some a few parameters.

57

LoRA (Low-Rank Adaptation)

- Transformers have many dense matrix multiply layers
  - Like $\mathbf{W^Q}$, $\mathbf{W^K}$, $\mathbf{W^V}$, $\mathbf{W^O}$ layers in attention
- Instead of updating these layers during finetuning,
  - Freeze these layers
  - Update a low-rank approximation with fewer parameters.

58

LoRA

- Consider a **matrix W** (shape $[N \times d]$) that needs to be updated during finetuning via gradient descent.
  - Normally updates are ΔW (shape $[N \times d]$)
- In LoRA, we freeze W and update instead a low-rank decomposition of W:
  - A of shape $[N \times r]$,
  - B of shape $[r \times d]$, r is very small (like 1 or 2)
  - That is, during finetuning we update A and B instead of W.
  - Replace W + ΔW with W + BA.
- Forward pass: instead of

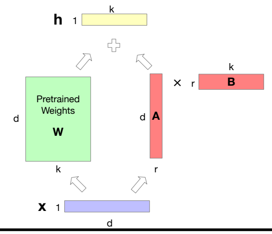
$$\mathbf{h = xW}$$

- We do

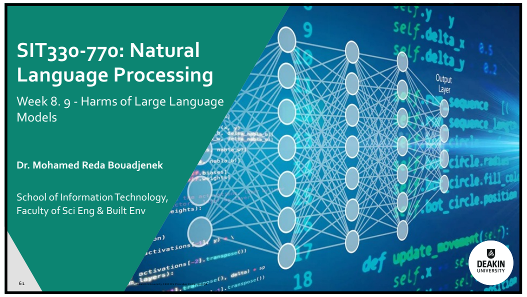
$$\mathbf{h = xW + xAB}$$

59

LoRA



60

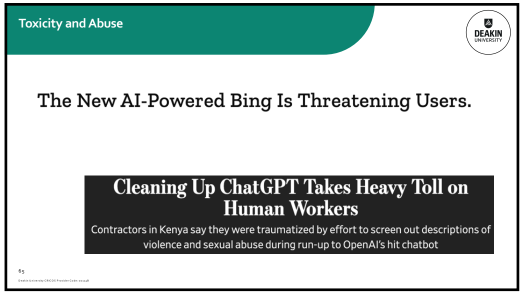


61

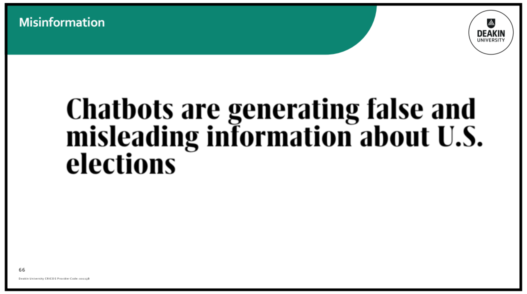
62

63

64



65



66