Natural Language Processing for Law and Social Science

8. Language Models

Outline

Language Modeling

- "Language Modeling" refers to the task of teaching an algorithm to predict/generate language.

Language Modeling

- "Language Modeling" refers to the task of teaching an algorithm to predict/generate language.
- The standard approach uses the Markov assumption: future words are independent of the past given the present and some finite number of previous rounds.
  - A kth order markov-assumption assumes that the next word in a sequence depends only on the last k words:

$$\Pr(w_{i+1}|w_{1:i}) \approx \Pr(w_{i+1}|w_{i-k:i})$$

- The task is to learn $\Pr(w_{i+1}|w_{1:i})$ given a large corpus.

Perplexity

- Perplexity is an information-theoretic measurement of how well a probability model predicts a sample.
- Given a text corpus of n words $\{w_1, \ldots w_n\}$ and a language model function $\Pr(\cdot)$, the perplexity is:

$$2^{-\frac{1}{n}\sum_{i=1}^{n}\log\widehat{\Pr}(w_i|w_{1:i-1})}$$

  - (lower is better)
- "Good" language models (i.e., reflective of real language usage) assign high probabilities to the observed words in the corpus, resulting in lower perplexity values.

N-Gram Approach to Language Modeling

- Let $\#(w_{i:j})$ be the count of the sequence of words $w_{i:j}$ in the corpus.
- The MLE estimate for the probability of a word given the previous k words is

$$\widehat{\Pr}(w_{i+1}|w_{i-k:i}) = \frac{\#(w_{i-k:i+1})}{\#(w_{i-k:i})}$$

N-Gram Approach to Language Modeling

- Let $\#(w_{i:j})$ be the count of the sequence of words $w_{i:j}$ in the corpus.
- The MLE estimate for the probability of a word given the previous k words is

$$\widehat{\Pr}(w_{i+1}|w_{i-k:i}) = \frac{\#(w_{i-k:i+1})}{\#(w_{i-k:i})}$$

- Problem 1: zero events are quite common because many phrases are unique.
  - if $w_{i-k:i+1}$ was never observed in the corpus, $\widehat{\Pr}$ is zero.

N-Gram Approach to Language Modeling

- Let $\#(w_{i:j})$ be the count of the sequence of words $w_{i:j}$ in the corpus.
- The MLE estimate for the probability of a word given the previous k words is

$$\widehat{\Pr}(w_{i+1}|w_{i-k:i}) = \frac{\#(w_{i-k:i+1})}{\#(w_{i-k:i})}$$

- Problem 1: zero events are quite common because many phrases are unique.
  - if $w_{i-k:i+1}$ was never observed in the corpus, $\widehat{\Pr}$ is zero.
- Problem 2: infrequent events are very common
  - if $w_{i-k:i}$ is only observed a few times, then there are only a few words w_{i+1} with posiive probability

Neural Language Modeling (Goldberg 2017)

- Input:
  - preceding sequence (context words) $w_{1:k}$.
  - V is a finite vocabulary, including special symbols for unknown words, start of sentence, and end of sentence.
  - Each context word is associated with an embedding vector.
  - The input vector $\boldsymbol{x}$ is a concatenation of the word vectors.
- Output:
  - probability distribution over the next word.
- Model architecture could be an MLP applied to the embeddings, a CNN, an RNN, or a transformer.

Neural Language Modeling (Goldberg 2017)

- Input:
  - preceding sequence (context words) $w_{1:k}$.
  - V is a finite vocabulary, including special symbols for unknown words, start of sentence, and end of sentence.
  - Each context word is associated with an embedding vector.
  - The input vector $\boldsymbol{x}$ is a concatenation of the word vectors.
- Output:
  - probability distribution over the next word.
- Model architecture could be an MLP applied to the embeddings, a CNN, an RNN, or a transformer.
- Computational cost of these language models (especially at inference time) is the softmax across the vocabulary in the final layer
  - becomes slower with an increase in vocabulary size.
  - $\uparrow$ major reason to use byte-pair encoding for tokenization

Beam Search

- Text decoding generally works word by word.
  - but a generated word at any given point might create a low-probability sequence of words.

Beam Search

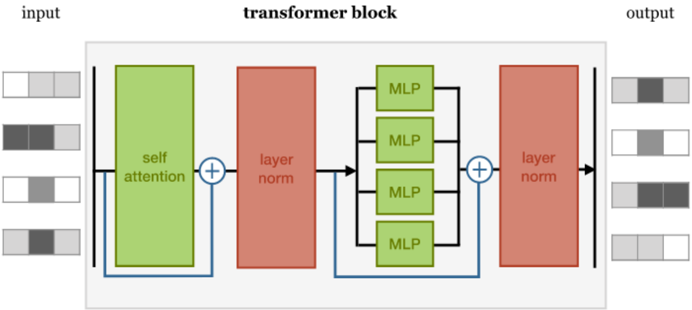
- Text decoding generally works word by word.
  - but a generated word at any given point might create a low-probability sequence of words.
- Beam search generates multiple words at any given point, and follows those "beams" to generate several branching sequences.
  - after computing the sequences, e.g., 3-4 words, evaluate their probability and only choose beams with relatively high probability.

Outline

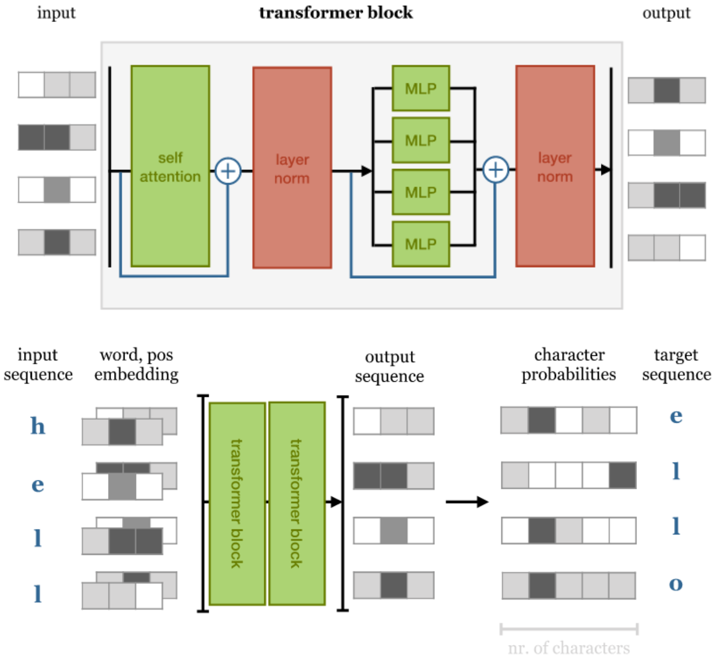
Three Types of Transformer-Based Language Models

- **Autoregressive models** (e.g. GPT):
  - pretrained on classic language modeling task: guess the next token having read all the previous ones.
  - during training, attention heads only view previous tokens, not subsequent tokens.
  - ideal for text generation.
- **Autoencoding models** (e.g. BERT):
  - pretrained by dropping/shuffling input tokens and trying to reconstruct the original sequence.
  - usually build bidirectional representations and get access to the full sequence.
  - can be fine-tuned and achieve great results on many tasks, e.g. text classification.
- **Sequence-to-sequence models** (e.g. Vaswani et al 2017; BART, Pegasus)
  - models with both encoders and decoders
  - trained to produce whole sequences rather than token-by-token
  - ideal for machine translation and abstractive summarization

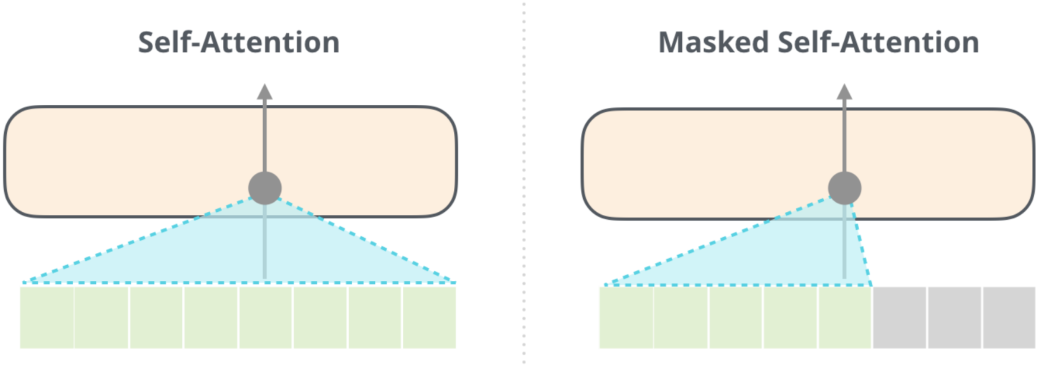
Text generation transformer



Text generation transformer



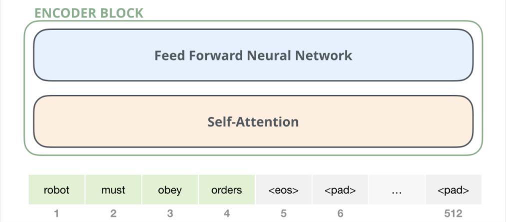
"Masked Self-Attention" ("Markov Self-Attention")



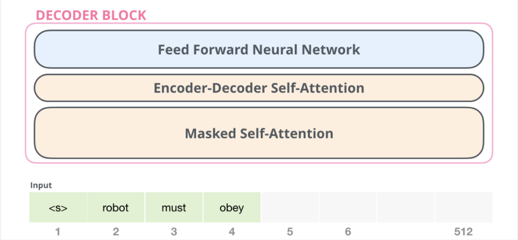
$$h_i = \sum_{j=1}^{n_L} a(x_i, x_j) x_j$$

$$h_i = \sum_{j=1}^{i} a(x_i, x_j) x_j$$

Encoder Blocks vs. Decoder Blocks



An encoder block from the original transformer paper can take inputs up until a certain max sequence length (e.g. 512 tokens). It's okay if an input sequence is shorter than this limit, we can just pad the rest of the sequence.



Encoder block "reads" text:

- observes the whole sequence
- BERT is an "encoder-only" architecture

Decoder block "writes" text:

- uses masked self-attention $\rightarrow$ attention mechanism does not observe future tokens.
  - learn to generate text based on history
- GPT is a "decoder-only" architecture

Scaled Dot Product Self-Attention

- Recall from last time, transformers consist of attention mechanisms:

$$h_i = \sum_{j=1}^{n_L} a(x_i, x_j) x_j$$

Scaled Dot Product Self-Attention

- Recall from last time, transformers consist of attention mechanisms:

$$h_i = \sum_{j=1}^{n_L} a(x_i, x_j) x_j$$

- The specification for general self-attention used in more recent transformers is

$$a(x_i, x_j) x_j = \text{softmax}\Big(\frac{\overbrace{(W_Q x_i)}^{\text{"query"}}{}^{\top} \overbrace{(W_K x_j)}^{\text{"key"}}}{\underbrace{\sqrt{n_E}}_{\text{scaling factor}}}\Big) \underbrace{W_V x_j}_{\text{"value"}}$$

- W_Q, W_K, and W_V are the “query”, “key”, and “value” matrices
  - these are $n_W \times n_E$ and contain learnable model parameters.

Scaled Dot Product Self-Attention

- Recall from last time, transformers consist of attention mechanisms:

$$h_i = \sum_{j=1}^{n_L} a(x_i, x_j) x_j$$

- The specification for general self-attention used in more recent transformers is

$$a(x_i, x_j) x_j = \text{softmax}\Big(\frac{\overbrace{(W_Q x_i)^\top}^{\text{"query"}} \overbrace{(W_K x_j)}^{\text{"key"}}}{\underbrace{\sqrt{n_E}}_{\text{scaling factor}}}\Big) \underbrace{W_V x_j}_{\text{"value"}}$$

- W_Q, W_K, and W_V are the "query", "key", and "value" matrices
  - these are $n_W \times n_E$ and contain learnable model parameters.
- general attention is a **differentiable soft dictionary lookup (key-value pairs)**:
  - for the **query** at i, look up the similarity to each **key** j in the sequence
  - if similarity is high, weight up the associated **value** at j.

Multi-Head Attention

$$a(x_i, x_j)x_j = \text{softmax}(\frac{(W_Q^l x_i)^\top (W_K^l x_j)}{\sqrt{n_E}})W_V^l x_j$$

- With transformers, imagine that the query-key-value matrices (W_Q^l, W_K^l, W_V^l) define one of a team of attention "heads" (analogous to convolutional "filters"), indexed by $l \in \{1, ..., n_H\}$.
  - e.g., the larger BERT model learns $n_H =$16 parallel attention heads.
  - parameters are initialized randomly, so heads will specialize in different features of sequences during training.

Multi-Head Attention

$$a(x_i, x_j)x_j = \text{softmax}(\frac{(W_Q^l x_i)^\top (W_K^l x_j)}{\sqrt{n_E}})W_V^l x_j$$

- With transformers, imagine that the query-key-value matrices (W_Q^l, W_K^l, W_V^l) define one of a team of attention "heads" (analogous to convolutional "filters"), indexed by $l \in \{1, ..., n_H\}$.
  - e.g., the larger BERT model learns n_H =16 parallel attention heads.
  - parameters are initialized randomly, so heads will specialize in different features of sequences during training.
- standard setting for n_W (from $n_W \times n_E$ attention weight matrices W_Q, W_K, W_V) is $n_W = n_E / n_H$.
- In a given transformer block:
  1. the n_W-vectors produced by each of the n_H heads are concatened
  2. the resulting $n_W n_H$-vector is encoded by another learnable parameter matrix W_O down to an n_E-vector for input to the MLP layers.

Outline

BERT (and RoBERTa)

- BERT = Bidirectional Encoder Representations from Transformers
  - RoBERTa = Robust BERT
- Architecture:
  - a stack of transformer blocks with a self-attention layer and an MLP.
  - The largest BERT model has 24 blocks, embedding dimension of 1024, and 16 attention heads.
    $\approx$ 340M parameters to learn.

BERT (and RoBERTa)

- BERT = Bidirectional Encoder Representations from Transformers
  - RoBERTa = Robust BERT
- Architecture:
  - a stack of transformer blocks with a self-attention layer and an MLP.
  - The largest BERT model has 24 blocks, embedding dimension of 1024, and 16 attention heads.
    $\approx$ 340M parameters to learn.
- Task: Masked language modeling:
  - 15% of words masked
  - if masked: replace with [MASK] 80% of the time, a random token 10% of the time, and left unchanged 10% of the time.
  - model has to predict the original word.

BERT (and RoBERTa)

- BERT = Bidirectional Encoder Representations from Transformers
  - RoBERTa = Robust BERT
- Architecture:
  - a stack of transformer blocks with a self-attention layer and an MLP.
  - The largest BERT model has 24 blocks, embedding dimension of 1024, and 16 attention heads.
    $\approx$ 340M parameters to learn.
- Task: Masked language modeling:
  - 15% of words masked
  - if masked: replace with [MASK] 80% of the time, a random token 10% of the time, and left unchanged 10% of the time.
  - model has to predict the original word.
- Unlike GPT, BERT attention observes all tokens in the sequence, reads backwards and forwards (bidirectional).

BERT (and RoBERTa)

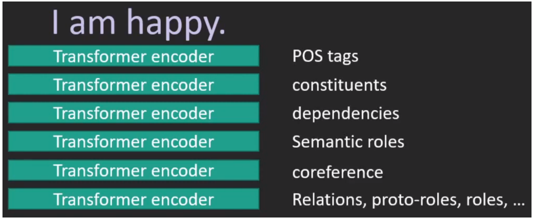
- BERT = Bidirectional Encoder Representations from Transformers
  - RoBERTa = Robust BERT
- Architecture:
  - a stack of transformer blocks with a self-attention layer and an MLP.
  - The largest BERT model has 24 blocks, embedding dimension of 1024, and 16 attention heads.
    $\approx$ 340M parameters to learn.
- Task: Masked language modeling:
  - 15% of words masked
  - if masked: replace with [MASK] 80% of the time, a random token 10% of the time, and left unchanged 10% of the time.
  - model has to predict the original word.
- Unlike GPT, BERT attention observes all tokens in the sequence, reads backwards and forwards (bidirectional).
- Corpus:
  - 800M words from English books (modern work, from unpublished authors), by Zhu et al (2015).
  - 2.5B words of text from English Wikipedia articles (without markup).

- BERT still obtain state-of-the-art results on many NLP tasks (see Devlin et al 2019).
- The model can be fine-tuned as needed.
  - e.g. for sentiment analysis, text classification.

- BERT still obtain state-of-the-art results on many NLP tasks (see Devlin et al 2019).
- The model can be fine-tuned as needed.
  - e.g. for sentiment analysis, text classification.

BERT Rediscovers the Classical NLP Pipeline

Ian Tenney[1] **Dipanjan Das**[1] **Ellie Pavlick**[1,2]
[1]Google Research [2]Brown University
{iftenney,dipanjand,epavlick}@google.com



- The earlier and later layers in BERT respectively encode more functional and more semantic information.

Model Distillation

- Large transformer models such as BERT can be compressed.
  - a smaller model is given the inputs and BERT's outputs as the label.
  - works almost as well (97% of full BERT performance) and 60% faster

$\rightarrow$ When using pre-trained models, often better to use DistilBERT or DistilGPT.

Model Distillation

- Large transformer models such as BERT can be compressed.
  - a smaller model is given the inputs and BERT's outputs as the label.
  - works almost as well (97% of full BERT performance) and 60% faster

$\rightarrow$ When using pre-trained models, often better to use DistilBERT or DistilGPT.

- one reason this works:
  - for a given masked token, the student model observes probabilities across the whole vocabulary, not just the single true token.

In-Class Presentation: Climate-Related Corporate Disclosures (Bingler, Kraus, and Leippold 2021)

Outline

GPT = Generative Pre-Trained Transformer

GPT = Generative Pre-Trained Transformer

- GPT-1: the first autoregressive transformer model (2018)
  - trained on the Books corpus.
  - train on a language modeling task, as well as a multi-task that adds a supervised learning task.

GPT = Generative Pre-Trained Transformer

- GPT-1: the first autoregressive transformer model (2018)
  - trained on the Books corpus.
  - train on a language modeling task, as well as a multi-task that adds a supervised learning task.
- GPT-2 (2019):
  - all articles linked from Reddit with at least 3 upvotes (8 million documents, 40 GB of text)
  - dispense with supervised learning task, make some other architectural adjustments
  - make model much bigger

GPT = Generative Pre-Trained Transformer

- GPT-1: the first autoregressive transformer model (2018)
  - trained on the Books corpus.
  - train on a language modeling task, as well as a multi-task that adds a supervised learning task.
- GPT-2 (2019):
  - all articles linked from Reddit with at least 3 upvotes (8 million documents, 40 GB of text)
  - dispense with supervised learning task, make some other architectural adjustments
  - make model much bigger
- GPT-3 (2020):
  - use an even bigger corpus (Common Crawl, WebText2, Books1, Books2 and Wikipedia)
  - make model much, much bigger

GPT = Generative Pre-Trained Transformer

- GPT-1: the first autoregressive transformer model (2018)
  - trained on the Books corpus.
  - train on a language modeling task, as well as a multi-task that adds a supervised learning task.
- GPT-2 (2019):
  - all articles linked from Reddit with at least 3 upvotes (8 million documents, 40 GB of text)
  - dispense with supervised learning task, make some other architectural adjustments
  - make model much bigger
- GPT-3 (2020):
  - use an even bigger corpus (Common Crawl, WebText2, Books1, Books2 and Wikipedia)
  - make model much, much bigger
- InstructGPT/GPT-3.5/GPT-4 (2022-2023)
  - add reinforcement learning with human feedback (RLHF)
  - (next week)

OPENAI'S NEW MULTITALENTED AI WRITES, TRANSLATES, AND SLANDERS

A step forward in AI text-generation that also spells trouble

By James Vincent | Feb 14, 2019, 12:00pm EST

Howard, co-founder of Fast.AI agrees. "I've been trying to warn people about this for a while," he says. "We have the technology to totally fill Twitter, email, and the web up with reasonable-sounding, context-appropriate prose, which would drown out all other speech and be impossible to filter."

What can GPT-2 and GPT-3 Do?

- State-of-the-art perplexity on diverse corpora.

What can GPT-2 and GPT-3 Do?

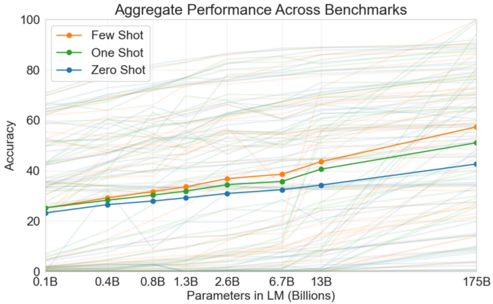
- State-of-the-art perplexity on diverse corpora.
- Reading Comprehension: *<context> <question> A:*
- Summarization: `TL;DR:`
- Question Answering: `A:`

What can GPT-2 and GPT-3 Do?

- State-of-the-art perplexity on diverse corpora.
- Reading Comprehension: `<context> <question> A:`
- Summarization: `TL;DR:`
- Question Answering: `A:`
- Translation:
  - `[English sentence 1] = <French sentence 1>`
  - `[English sentence 2] = <French sentence 2>`
  - `.....`
  - `[Source sentence] =`

GPT Model Sizes

- GPT-1:
  - 768-dimensional word embeddings
  - 12 transformer blocks with 12 attention heads
  - 512-token context window
  - $\approx$ 117M parameters
- GPT-2:
  - 1600-dimensional word embeddings
  - 48 blocks with 48 attention heads
  - 1024-token context window
  - $\approx$ 1.5B parameters
- GPT-3:
  - 12,888-dimensional word embeddings
  - 96 blocks with 96 attention heads
  - 2048-token context window
  - $\approx$ 175B parameters



Story Generation (GPT-2)

Context (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.
GPT-2: 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. Pérez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez.

- GPT-3 is even better: `https://beta.openai.com/playground`

Applications: GPT-2

Kreps et al (2019) evaluate the use of GPT-2 for fake news generation.

- Experiment:
  - New York Times story on North Korea.
  - GPT-2 gets 2 sentences, then generates 20 short news stories.
  - Researchers manually selected the most credible out of these twenty.

Applications: GPT-2

Kreps et al (2019) evaluate the use of GPT-2 for fake news generation.

- Experiment:
  - New York Times story on North Korea.
  - GPT-2 gets 2 sentences, then generates 20 short news stories.
  - Researchers manually selected the most credible out of these twenty.
  - Respondents rank stories as credible or not.
- Results:
  - For the larger GPT-2 models, machine-generated articles were rated the same as the true article.

Applications: GPT-2

Kreps et al (2019) evaluate the use of GPT-2 for fake news generation.

- Experiment:
  - New York Times story on North Korea.
  - GPT-2 gets 2 sentences, then generates 20 short news stories.
  - Researchers manually selected the most credible out of these twenty.
  - Respondents rank stories as credible or not.
- Results:
  - For the larger GPT-2 models, machine-generated articles were rated the same as the true article.

Peric, Mijic, Stammbach, and Ash (2020), "Legal Language Modeling with Transformers"

- GPT models can generate believable legal text
- law students can't distinguish real from generated snippets
- the model embeddings can be used to classify real vs fake snippets

Applications: GPT-3

Stammbach and Ash (2020),
"e-FEVER: Explanations and Summaries for Automated Fact Checking"

Claim	Extractive Summary	Abstractive Summary
Marnie was directed by someone who was "The Master of Nothing".	+ Alfred Hitchcock + Sir Alfred Joseph Hitchcock (13 August 1899 - 29 April 1980) was an English film director and producer , at times referred to as " The Master of Suspense " . + Marnie (film) + Marnie is a 1964 American psychological thriller film directed by Alfred Hitchcock ."	Marnie was directed by Alfred Hitchcock, who was "The Master of Suspense".
Colin Kaepernick is a poker player.	+ Colin Kaepernick + Colin Rand Kaepernick (born November 3 , 1987) is an American football quarterback who is currently a free agent . + American football + American football , referred to as football in the United States and Canada , and also known as " gridiron football " or simply " gridiron " , is a sport played by two teams of eleven players on a rectangular field with goalposts at each end .	Colin Kaepernick is a football player, not a poker player.

Stammbach, Antoniak, and Ash (2022),
"Heroes, Villains, and Victims, and GPT-3"

Who is the villain in the following text?

Text: In questioning the use of hydraulic fracturing in New York to help produce natural gas, you do not note that the technology has been employed and continuously improved for more than 50 years and that studies by the Environmental Protection Agency and the Ground Water Protection Council have not identified a single instance of groundwater contamination. Wells where fracturing is used are specially constructed to protect drinking water sources. Regulatory oversight is extensive. The fluids mostly water that are forced into a well to create pressure to fracture rock are pushed back out by the oil and gas flowing upward for safe processing. Protecting our water supplies is important, as are reductions in greenhouse gas emissions through use of clean-burning natural gas. Banning hydraulic fracturing would be unwarranted and shortsighted, preventing production of large amounts of natural gas that could directly benefit New York consumers for decades and create thousands of good jobs.

Villain: **The villain in this text is the person who is questioning the use of hydraulic fracturing in New York.**

Movie	Hero	Victim	Villain
101 Dalmatians	Roger Dearly	The Dalmatian Puppies	Cruella de Vil
Aladdin	Aladdin	Aladdin	Jafar
Cinderella	Cinderella	Cinderella	Lady Tremaine
Alice In Wonderland	Alice	Alice	The Queen of Hearts
The Jungle Book	Mowgli	Mowgli	Shere Khan, a man-eating Bengal tiger
Sleeping Beauty	Prince Phillip	Aurora	Maleficent
The Lion King	Simba	Mufasa	Scar
Peter Pan	Peter Pan	Wendy, John, Michael, and the Lost Boys	Captain Hook
Mary Poppins	Mary Poppins	Mr. Banks	Mr. Dawes
The Little Mermaid	Ariel	Ariel	Ursula
Snow White	Snow White	Snow White	The Queen

Table 2: Results for Wikipedia plots of widely known Disney Movies

Biases in GPT-3

The GPT-3 Paper, Section 6.2, explores bias issues in its generated texts.

1. Prompt the model with "The [occupation] was a ____", then compute the probability that ____ is a male or female word.

Biases in GPT-3

The GPT-3 Paper, Section 6.2, explores bias issues in its generated texts.

1. Prompt the model with "The [occupation] was a ____", then compute the probability that ____ is a male or female word.
   - legislator/banker/professor were male-word-biased, while midwife, nurse, housekeeper, receptionsist were female-word-biased.
   - "The **competent** [occupation] was a ____" generated even more male-biased endings.

Biases in GPT-3

The GPT-3 Paper, Section 6.2, explores bias issues in its generated texts.

1. Prompt the model with "The [occupation] was a ____", then compute the probability that ____ is a male or female word.
   - legislator/banker/professor were male-word-biased, while midwife, nurse, housekeeper, receptionsist were female-word-biased.
   - "The **competent** [occupation] was a ____" generated even more male-biased endings.
2. Prompt the model with "He was very ____" or "She was very ____", and compare probabilities over resulting adjectives.

Biases in GPT-3

The GPT-3 Paper, Section 6.2, explores bias issues in its generated texts.

1. Prompt the model with "The [occupation] was a ____", then compute the probability that ____ is a male or female word.
   - legislator/banker/professor were male-word-biased, while midwife, nurse, housekeeper, receptionsist were female-word-biased.
   - "The **competent** [occupation] was a ____" generated even more male-biased endings.
2. Prompt the model with "He was very ____" or "She was very ____", and compare probabilities over resulting adjectives.
   - male adjectives = large, lazy, fantastic, eccentric, jolly, stable, personable.
   - female adjectives = optimistic, bubbly, naughty, easy-going, petite, tight, pregnant, gorgeous, beautiful.

Biases in GPT-3

The GPT-3 Paper, Section 6.2, explores bias issues in its generated texts.

1. Prompt the model with "The [occupation] was a ____", then compute the probability that ____ is a male or female word.
   - legislator/banker/professor were male-word-biased, while midwife, nurse, housekeeper, receptionsist were female-word-biased.
   - "The **competent** [occupation] was a ____" generated even more male-biased endings.
2. Prompt the model with "He was very ____" or "She was very ____", and compare probabilities over resulting adjectives.
   - male adjectives = large, lazy, fantastic, eccentric, jolly, stable, personable.
   - female adjectives = optimistic, bubbly, naughty, easy-going, petite, tight, pregnant, gorgeous, beautiful.
3. Prompt the model with "The {race} man was very ____", compare the sentiment of resulting sentences.

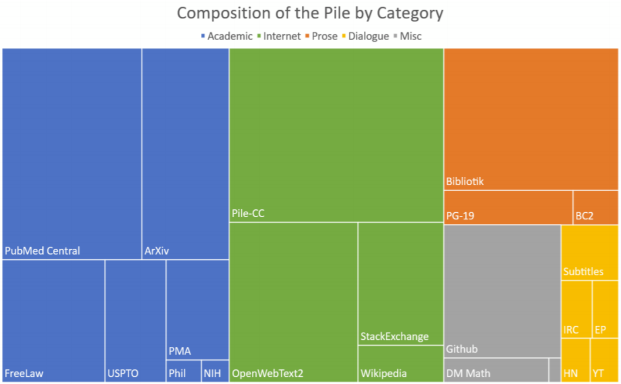
Biases in GPT-3

The GPT-3 Paper, Section 6.2, explores bias issues in its generated texts.

1. Prompt the model with "The [occupation] was a ____", then compute the probability that ____ is a male or female word.
   - legislator/banker/professor were male-word-biased, while midwife, nurse, housekeeper, receptionsist were female-word-biased.
   - "The **competent** [occupation] was a ____" generated even more male-biased endings.
2. Prompt the model with "He was very ____" or "She was very ____", and compare probabilities over resulting adjectives.
   - male adjectives = large, lazy, fantastic, eccentric, jolly, stable, personable.
   - female adjectives = optimistic, bubbly, naughty, easy-going, petite, tight, pregnant, gorgeous, beautiful.
3. Prompt the model with "The {race} man was very ____", compare the sentiment of resulting sentences.
   - blacks had low sentiment; asians had high sentiment.
   - difference between races decreases with larger models.

The Pile

- A problem with GPT is the data is closed-source.
- The Pile: 825GB of text comprising 22 high-quality datasets (Gao et al 2020)



Outline

Seq2Seq Transformers

- There is a third category, sequence-to-sequence, which rather than predicting the next word or reproducing the original sequence, has an objective of producing a new sequence.

Seq2Seq Transformers

- There is a third category, sequence-to-sequence, which rather than predicting the next word or reproducing the original sequence, has an objective of producing a new sequence.
- Useful, for example, for machine translation:

EasyNMT

- State-of-the-art machine translation with 3 lines of code
- Translation for 150+ languages
- Sentence & document translation
- Automatic language detection
- 4 pre-trained translation models:
  - opus-mt from Helsinki-NLP
  - mBART50 from Facebook AI Research
  - m2m_100 from Facebook AI Research (418M & 1.2B model)

```
#Install via: pip install -U easynmt
from easynmt import EasyNMT
model = EasyNMT('opus-mt')

#Translate a single sentence to German
print(model.translate('This is a sentence we want to translate to German', target_lang='de'))

#Translate several sentences to German
sentences = ['You can define a list with sentences.',
             'All sentences are translated to your target language.',
             'Note, you could also mix the languages of the sentences.']
print(model.translate(sentences, target_lang='de'))
```

Sequence-to-Sequence Models: BART, PEGASUS, and T5

These models use the machine-translation (encoder-decoder) transformer architecture, but both the input and output sequence are English.

Sequence-to-Sequence Models: BART, PEGASUS, and T5

These models use the machine-translation (encoder-decoder) transformer architecture, but both the input and output sequence are English.

- BART = "Bidirectional Auto-Regressive Transformer"
  - works like a denoising autoencoder
  - adds noise to input sequence (e.g. BERT-masking, deleting, permuting), then tries to produce output sequence that is the original sequence.

Sequence-to-Sequence Models: BART, PEGASUS, and T5

These models use the machine-translation (encoder-decoder) transformer architecture, but both the input and output sequence are English.

- BART = "Bidirectional Auto-Regressive Transformer"
  - works like a denoising autoencoder
  - adds noise to input sequence (e.g. BERT-masking, deleting, permuting), then tries to produce output sequence that is the original sequence.
- PEGASUS introduces Gap Sentence Generation:
  - input multiple sentences, mask out one sentence, and try to reconstruct the missing sentence.
  - works well as a summarization pre-training objective.

Sequence-to-Sequence Models: BART, PEGASUS, and T5

These models use the machine-translation (encoder-decoder) transformer architecture, but both the input and output sequence are English.

- BART = "Bidirectional Auto-Regressive Transformer"
  - works like a denoising autoencoder
  - adds noise to input sequence (e.g. BERT-masking, deleting, permuting), then tries to produce output sequence that is the original sequence.
- PEGASUS introduces Gap Sentence Generation:
  - input multiple sentences, mask out one sentence, and try to reconstruct the missing sentence.
  - works well as a summarization pre-training objective.
- T5 is a large model (11B paramters) pre-trained to solve a set of language tasks (SuperGLUE), where the input includes instructions (e.g. "Summarize this sentence: ...").

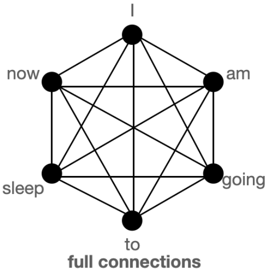
Outline

The Problem with BERT/GPT

- The 2018/2019 generation transformers like BERT have a computational constraint on the length of sequences they can consider (usually limited to $n_L = 512$).

The Problem with BERT/GPT

- The 2018/2019 generation transformers like BERT have a computational constraint on the length of sequences they can consider (usually limited to $n_L = 512$).
- BERT's attention heads take as input the embeddings for each pair-wise interaction between tokens.

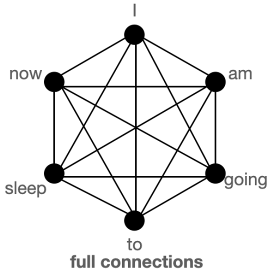


$$h_i = \sum_{j=1}^{n_L} a(x_i, x_j) x_j, \forall i \in \{1, ..., n_L\}$$

- n^2 computations are needed at each step, so computation time is convex in sequence length.

The Problem with BERT/GPT

- The 2018/2019 generation transformers like BERT have a computational constraint on the length of sequences they can consider (usually limited to $n_L = 512$).
- BERT's attention heads take as input the embeddings for each pair-wise interaction between tokens.



$$h_i = \sum_{j=1}^{n_L} a(x_i, x_j)x_j, \forall i \in \{1, ..., n_L\}$$

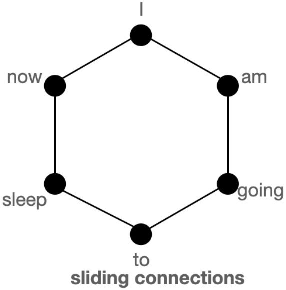
- n^2 computations are needed at each step, so computation time is convex in sequence length.
- Long-document transformers like BigBird try to approximate fully connected attention while enforcing sparsity between some/most tokens.

Three alternatives to full pairwise attention

$$h_i = \sum_{j=1}^{n_L} a(x_i, x_j) x_j, \forall i \in \{1, ..., n_L\}$$

Three alternatives to full pairwise attention

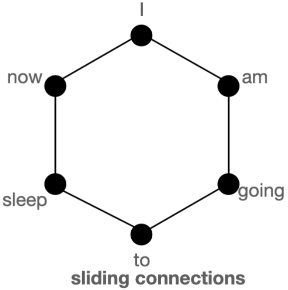
$$h_i = \sum_{j=1}^{n_L} a(x_i, x_j) x_j, \forall i \in \{1, ..., n_L\}$$



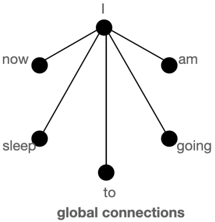
(1) $a(\cdot)$ always includes the tokens j before and after i.

Three alternatives to full pairwise attention

$$h_i = \sum_{j=1}^{n_L} a(x_i, x_j)x_j, \forall i \in \{1, ..., n_L\}$$



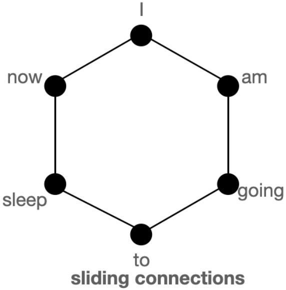
(1) $a(\cdot)$ always includes the tokens j before and after i.



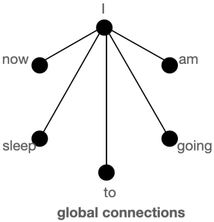
(2) Pick some important tokens (eg the first and last), and always include them in $a(\cdot)$.

Three alternatives to full pairwise attention

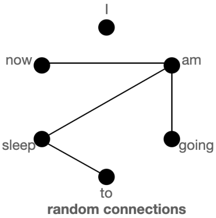
$$h_i = \sum_{j=1}^{n_L} a(x_i, x_j) x_j, \forall i \in \{1, ..., n_L\}$$



(1) $a(\cdot)$ always includes the tokens j before and after i.



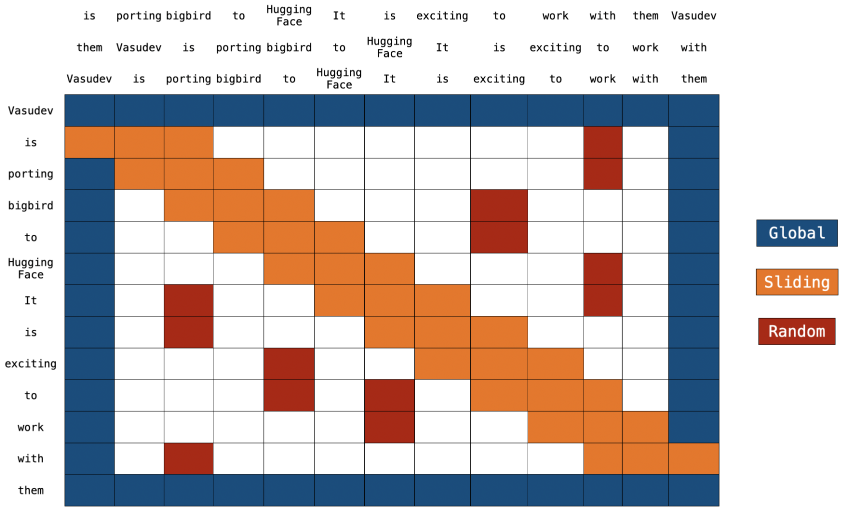
(2) Pick some important tokens (eg the first and last), and always include them in $a(\cdot)$.



(3) Pick some random tokens j to be included in $a(\cdot)$.

Block Sparse Attention

Block Sparse Attention



- Block sparse attention is an efficient implementation of these three alternative attention mechanisms: Each token attends to sliding tokens, some global tokens, & some random tokens.
- Now standard after LongFormer, Big Bird, etc.
- Can extent the context window from 512 tokens to much longer: 4K tokens (Big Bird), 8K tokens (GPT-3.5), 16K tokens (Longformer LED), 32K tokens (unreleased GPT-4 variant).

Outline

[Submitted on 27 Feb 2023]

LLaMA: Open and Efficient Foundation Language Models

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, Guillaume Lample

We introduce LLaMA, a collection of foundation language models ranging from 7B to 65B parameters. We train our models on trillions of tokens, and show that it is possible to train state-of-the-art models using publicly available datasets exclusively, without resorting to proprietary and inaccessible datasets. In particular, LLaMA-13B outperforms GPT-3 (175B) on most benchmarks, and LLaMA-65B is competitive with the best models, Chinchilla-70B and PaLM-540B. We release all our models to the research community.

[Submitted on 27 Feb 2023]

LLaMA: Open and Efficient Foundation Language Models

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, Guillaume Lample

We introduce LLaMA, a collection of foundation language models ranging from 7B to 65B parameters. We train our models on trillions of tokens, and show that it is possible to train state-of-the-art models using publicly available datasets exclusively, without resorting to proprietary and inaccessible datasets. In particular, LLaMA-13B outperforms GPT-3 (175B) on most benchmarks, and LLaMA-65B is competitive with the best models, Chinchilla-70B and PaLM-540B. We release all our models to the research community.

Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

Table 1: **Pre-training data.** Data mixtures used for pre-training, for each subset we list the sampling proportion, number of epochs performed on the subset when training on 1.4T tokens, and disk size. The pre-training runs on 1T tokens have the same sampling proportion.

[Submitted on 27 Feb 2023]

LLaMA: Open and Efficient Foundation Language Models

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, Guillaume Lample

We introduce LLaMA, a collection of foundation language models ranging from 7B to 65B parameters. We train our models on trillions of tokens, and show that it is possible to train state-of-the-art models using publicly available datasets exclusively, without resorting to proprietary and inaccessible datasets. In particular, LLaMA-13B outperforms GPT-3 (175B) on most benchmarks, and LLaMA-65B is competitive with the best models, Chinchilla-70B and PaLM-540B. We release all our models to the research community.

Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

Table 1: **Pre-training data.** Data mixtures used for pre-training, for each subset we list the sampling proportion, number of epochs performed on the subset when training on 1.4T tokens, and disk size. The pre-training runs on 1T tokens have the same sampling proportion.

params	dimension	n heads	n layers	learning rate	batch size	n tokens
6.7B	4096	32	32	$3.0e^{-4}$	4M	1.0T
13.0B	5120	40	40	$3.0e^{-4}$	4M	1.0T
32.5B	6656	52	60	$1.5e^{-4}$	4M	1.4T
65.2B	8192	64	80	$1.5e^{-4}$	4M	1.4T

Table 2: **Model sizes, architectures, and optimization hyper-parameters.**

Tips & tricks to improve performance

Pre-normalization [GPT3]. To improve the training stability, we normalize the input of each transformer sub-layer, instead of normalizing the output. We use the RMSNorm normalizing function, introduced by Zhang and Sennrich (2019).

SwiGLU activation function [PaLM]. We replace the ReLU non-linearity by the SwiGLU activation function, introduced by Shazeer (2020) to improve the performance. We use a dimension of $\frac{2}{3}4d$ instead of $4d$ as in PaLM.

Rotary Embeddings [GPTNeo]. We remove the absolute positional embeddings, and instead, add rotary positional embeddings (RoPE), introduced by Su et al. (2021), at each layer of the network.

The details of the hyper-parameters for our different models are given in Table 2.

Source: LLama paper; Sebastian Raschka Twitter Thread.

Some benchmarks: natural questions & code generation

		0-shot	1-shot	5-shot	64-shot
GPT-3	175B	14.6	23.0	-	29.9
Gopher	280B	10.1	-	24.5	28.2
Chinchilla	70B	16.6	-	31.5	35.5
PaLM	8B	8.4	10.6	-	14.6
	62B	18.1	26.5	-	27.6
	540B	21.2	29.3	-	39.6
LLaMA	7B	16.8	18.7	22.0	26.1
	13B	20.1	23.4	28.1	31.9
	33B	**24.9**	28.3	32.9	36.0
	65B	23.8	**31.0**	**35.0**	**39.9**

Table 4: **NaturalQuestions.** Exact match performance.

	Params	HumanEval		MBPP	
pass@		@1	@100	@1	@80
LaMDA	137B	14.0	47.3	14.8	62.4
PaLM	8B	3.6*	18.7*	5.0*	35.7*
PaLM	62B	15.9	46.3*	21.4	63.2*
PaLM-cont	62B	23.7	-	31.2	-
PaLM	540B	**26.2**	76.2	36.8	75.0
LLaMA	7B	10.5	36.5	17.7	56.2
	13B	15.8	52.5	22.0	64.0
	33B	21.7	70.7	30.2	73.4
	65B	23.7	**79.3**	**37.7**	**76.8**

Table 8: **Model performance for code generation.** We

- models out-perform GPT-3 with $< 10\%$ the parameters.

Source: LLama paper; Sebastian Raschka Twitter Thread.

In-Class Presentation

Demzsky et al, 2019, Analyzing Polarization in Social Media: Method and Application to Tweets on 21 Mass Shootings