Natural Language Processing for Law and Social Science

10. 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:
$$2^{-\frac{1}{n}\sum_{i=1}^{n}\log \widehat{\Pr}(w_i|w_{1:i-1})}$$
- Good language models (i.e., reflective of real language usage) assign high probabilities to the observed words in the corpus, resulting in lower (better) 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

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

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

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.

Neural Language Modeling (Goldberg 2017)

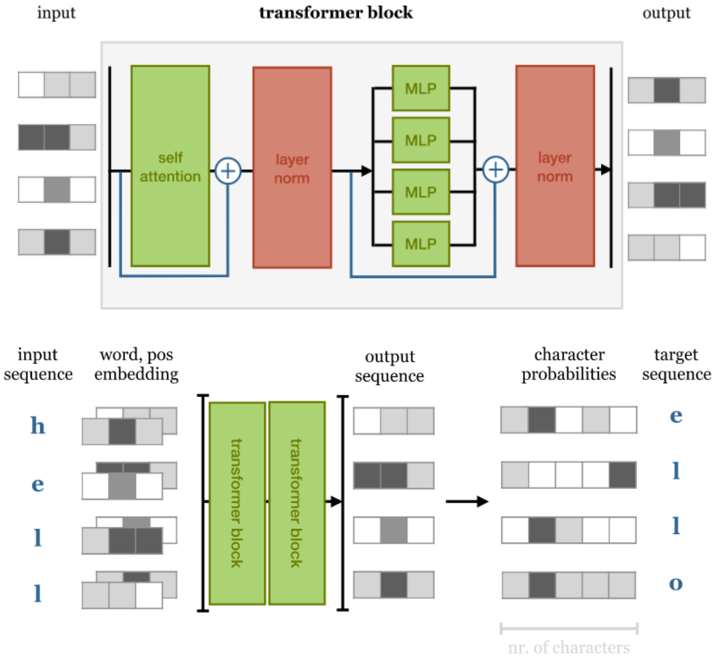
- Input:
  - preceding sequence (context words) $w_{1:k}$.
  - V is a finite vocabulary, including special symbols for unknown words, start of sentence, and end of sentence.
  - Each context word is associated with an embedding vector.
  - The input vector $\boldsymbol{x}$ is a concatenation of the word 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 is the softmax across the vocabulary in the final layer, which becomes slower with an increase in vocabulary size.

Autoregressive vs Autoencoding Language Models

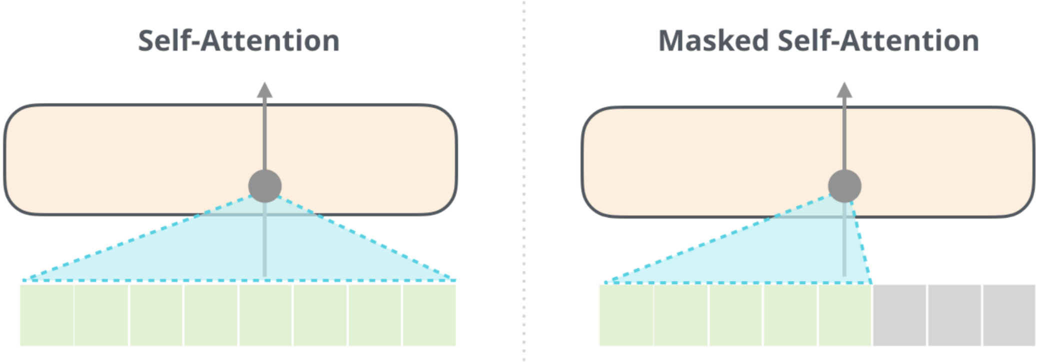
- **Autoregressive models** (e.g. GPT):
  - pretrained on classic language modeling task: guess the next token having read all the previous ones.
  - during training, attention heads only view previous tokens, not subsequent tokens.
  - ideal for text generation.
- **Autoencoding models** (e.g. BERT):
  - 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.

Outline

Text generation transformer



Masked Self-Attention



- An autoregressive model's attention mechanism only looks at previous tokens:

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

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

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?

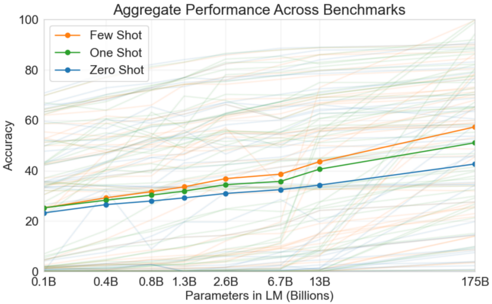
- State-of-the-art perplexity on diverse corpora.
- Reading Comprehension: `<context> <question> A:`
- Summarization: `TL;DR:`
- Question Answering: `A:`

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.

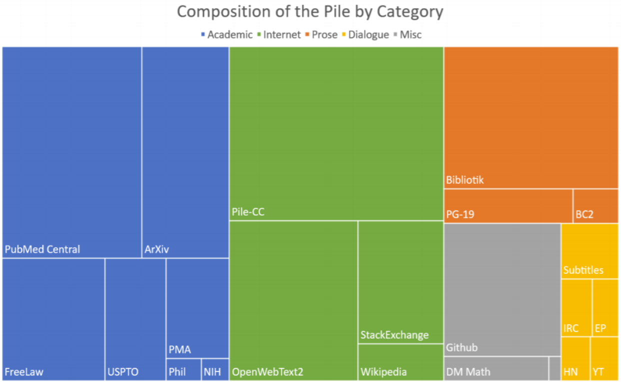
Villain: **The villain in this text is the person who is questioning the use of hydraulic fracturing in New York.**

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

GPT-Neo

- The Eleuther AI team is building an open-source equivalent of GPT-3.
  - `https://github.com/EleutherAI/gpt-neo/`
- Dataset: The Pile:
  - 825GB of text comprising 22 high-quality datasets (Gao et al 2020)



- Have already trained and released 1.3B and 2.7B parameter models based on GPT architecture.

PaLM (April 2022)

Explaining a Joke

I will explain these jokes:
(1) The problem with kleptomaniacs is that they always take things literally.
Explanation: This joke is wordplay. Someone who "takes things literally" is someone who doesn't fully understand social cues and context, which is a negative trait. But the definition of kleptomania is someone who literally takes things.
(2) Always borrow money from a pessimist. They'll never expect it back.
Explanation: Most people expect you to pay them back when you borrow money, however a pessimist is someone who always assumes the worst, so if you borrow money from them, they will expect that you won't pay them back anyways.

Input: I was supposed to start writing the paper at 5:00 PM. But then I started playing with this cool new language model for 10 minutes. 10 minutes later, it's suddenly 9:30 PM!
Model Output: This joke is about how time flies when you're having fun. The person was supposed to start writing a paper at 5:00 PM, but they started playing with a language model instead. They thought they were only playing with it for 10 minutes, but it was actually 4.5 hours.

Input: I think I understand why me and my partner are such a good match. I get excited when I smell food and become super alert when the doorbell rings. My girlfriend Shelley becomes ecstatic when she sees a person who she just saw 5 hours ago.
Model Output: This joke is about dogs. Dogs get excited when they smell food and when the doorbell rings. The girlfriend is excited to see someone she just saw 5 hours ago, which is a trait of dogs.

Input: I was going to fly to visit my family on April 6th. My mom said "Oh great, your step dad's poetry reading is that night!" So now I'm flying in on April 7th.
Model Output: The joke is that the speaker's mother is trying to get them to go to their step dad's poetry reading, but the speaker doesn't want to go, so they are changing their flight to the day after the poetry reading.

Input: I tried 10,000 random restarts of my neural network, but I was accused of overfitting. I guess no good seed goes unpunished.
Model Output: This joke is a pun. A neural network is a computer program that can learn from data. A "seed" is a number that is used to initialize a random number generator. A "good seed" is a number that produces a good random number generator. The phrase "no good deed goes unpunished" means that if you do something good, you will be punished for it.

Input: Reading "War and Peace" doesn't scare me. I once had to review the CL for my intern's whole project on his final day!
Model Output: 'War and Peace' is a very long book, so reading it is a daunting task. However, the speaker is saying they had to review the code for their intern's project on their final day, which is even more daunting.

Input: What's the difference between a zebra and an umbrella? One is a striped animal related to horses, another is a device you use to stop the rain from falling on you.
Model Output: This joke is an anti-joke. The joke is that the answer is obvious, and the joke is that you were expecting a funny answer.

Outline

Conditioned Generation

- Text generators can use metadata, for example on the speaker.
  - e.g., Li et al (2016) learn a categorical embedding for each user who wrote a response, in order to produce automated responses in the style of each user.
- As a side effect of training the generator, the network learns user embeddings, producing similar vectors to users who have similar communication styles.
  - At test time, one can influence the style of the generated response by feeding in a particular user (or average user vector) as a conditioning context.

Grover: Modify GPT-2 for conditioned fake news generation

- Next predicted word is based not only on previous words (text body), but also on metadata:

$$p(\text{domain}, \text{date}, \text{authors}, \text{headline}, \text{body}).$$

  - e.g., user specifies domain, date, and headline. Grover generates a) text body, then b) authors, then c) more realistic headline.

Grover: Modify GPT-2 for conditioned fake news generation

- Next predicted word is based not only on previous words (text body), but also on metadata:

$$p(\text{domain}, \text{date}, \text{authors}, \text{headline}, \text{body}).$$

  - e.g., user specifies domain, date, and headline. Grover generates a) text body, then b) authors, then c) more realistic headline.
- fields are dropped with probability 10%, and all but body with probability 35% $\rightarrow$ Grover learns to perform unconditional generation.
- RealNews Corpus:
  - Authors scraped all news articles from the 5000 news domains from Google News, Dec 2016 to April 2019
  - 120 GB of uncompressed text after deduplication

Grover: Results

- Grover can generate articles in particular newspaper styles.
  - metadata is important: it reduced perplexity from 9.3 to 8.7

Grover: Results

- Grover can generate articles in particular newspaper styles.
  - metadata is important: it reduced perplexity from 9.3 to 8.7
- Grover is better than human generated fake news:

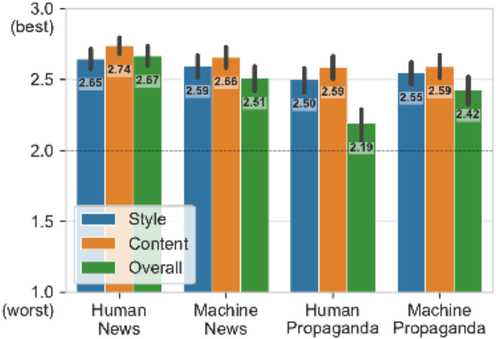


Figure 4: Human evaluation. For each article, three annotators evaluated style, content, and the overall trustworthiness; 100 articles of each category were used. The results show that propaganda generated by GROVER is rated more plausible than the original human-written propaganda.

Grover: Fake News Detection

- They take Grover's document embedding for the true article and generated articles, and feed that to a true/false classifier.
  - they do it in the test set month (April 2019); training ended in March 2019.
- Grover can identify its own fake articles with 81% accuracy.
  - better than other models including BERT

Plug and Play Language Model (PPLM)

- Setup:
  - Let $p(w)$ = unconditional language model (LM), a probability distribution for word w given history w_0

Plug and Play Language Model (PPLM)

- Setup:
  - Let $p(w)$ = unconditional language model (LM), a probability distribution for word w given history w_0
  - $p(w|a)$ = conditional LM, generating sentences with attribute a.

Plug and Play Language Model (PPLM)

- Setup:
  - Let $p(w)$ = unconditional language model (LM), a probability distribution for word w given history w_0
  - $p(w|a)$ = conditional LM, generating sentences with attribute a.
- By Bayes rule:

$$p(w|a) \propto p(a|w)p(w)$$

  - can be approximated efficiently using the method in Ngyuen et al (2016).

Plug and Play Language Model (PPLM)

- Setup:
  - Let $p(w)$ = unconditional language model (LM), a probability distribution for word w given history w_0
  - $p(w|a)$ = conditional LM, generating sentences with attribute a.
- By Bayes rule:

$$p(w|a) \propto p(a|w)p(w)$$

  - can be approximated efficiently using the method in Ngyuen et al (2016).
- PPLM insight:
  - instead of training $p(w|a)$ from scratch, take pre-trained $p(w)$, learn auxiliary model $p(a|w)$, and approximate $p(w|a)$ using Bayes rule.
- At each word step, use the gradients from the language model and the auxiliary model to increase both probabilities.
  - works to maintain fluency of generated language.

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)

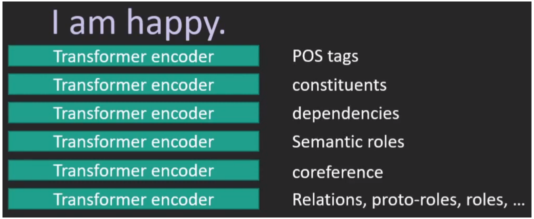
- BERT = Bidirectional Encoder Representations from Transformers
  - RoBERTa = Robust BERT
- Architecture:
  - a stack of transformer blocks with a self-attention layer and an MLP.
  - The largest BERT model has 24 blocks, embedding dimension of 1024, and 16 attention heads.
    $\approx$ 340M parameters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



- Like the earlier and later layers in ELMo, 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, usually 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, usually 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.

Application: Climate-Related Corporate Disclosures (Bingler, Kraus, and Leippold 2021)

- Fine-tunes RoBERTa ("Robust BERT") to classify texts related to corporate climate disclosures (using hand-annotated sample).

Table 3. Out-of-sample performance comparison between baseline models and our proposed ClimateBERT. Performance is reported in precision for each category.

	Governance	Strategy	Risk Management	Metrics & Targets	General Language	Overall Accuracy
Tf-idf	0.43	0.00	0.40	0.35	0.00	0.24
Sentence Enc.	0.19	0.57	0.15	0.24	0.00	0.23
RoBERTa Para.	0.26	0.25	0.25	0.25	0.07	0.22
RoBERTa Sent.	0.96	0.92	0.84	0.74	0.32	0.75
ClimateBERT	0.94	0.90	0.79	0.77	0.65	0.81

- model applied to large sample, shows that most disclosures are about more subjective / less verifiable aspects of climate disclosures.

Outline

Subword Tokenization

Modern transformer models use subword tokenization:

- construct character-level n-grams
- whitespace treated the same as letters
- all letters to lowercase, but add a special character for the next letter being capitalized.

Subword Tokenization

Modern transformer models use subword tokenization:

- construct character-level n-grams
- whitespace treated the same as letters
- all letters to lowercase, but add a special character for the next letter being capitalized.

e.g., BERT's SentencePiece tokenizer:

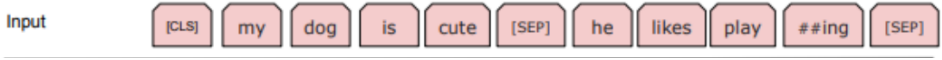
- character-level byte-pair encoder, learns character n-grams to breaks words like "playing" into "play" and "##ing".
- have to fix a vocabulary size: e.g. BERT uses 30K.

Subword Tokenization

Modern transformer models use subword tokenization:

- construct character-level n-grams
- whitespace treated the same as letters
- all letters to lowercase, but add a special character for the next letter being capitalized.

e.g., BERT's SentencePiece tokenizer:



- character-level byte-pair encoder, learns character n-grams to breaks words like "playing" into "play" and "##ing".
- have to fix a vocabulary size: e.g. BERT uses 30K.

This approach works well because the softmax over the vocabulary for output layer is computationally expensive.

- preferred to a character-level model because some sequences are relatively frequent – e.g., the sequence "the" shows up more often than the letter "Z".

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

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

- 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.

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.
- general attention is a **differentiable soft dictionary lookup**:
  - 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

- Recall that convolutional neural nets use a team of filters at a given convolutional layer to specialize in learning different features.

Multi-Head Attention

- Recall that convolutional neural nets use a team of filters at a given convolutional layer to specialize in learning different features.

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

- 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 "filter"), 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

- Recall that convolutional neural nets use a team of filters at a given convolutional layer to specialize in learning different features.

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

- 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 "filter"), 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

Bias in Language Models

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.

Bias in Language Models

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.

Bias in Language Models

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.

Bias in Language Models

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.

Bias in Language Models

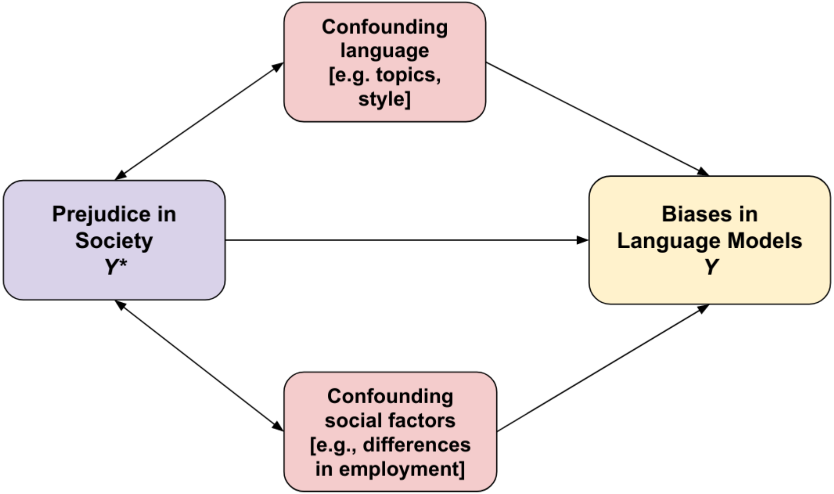
The GPT-3 Paper, Section 6.2, explores bias issues in its generated texts.

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.

Bias in Language Models

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.



- If there is a true level of prejudice or discrimination in society, Y^*, it can be measured with error and confounding from language models as

$$Y = Y^* + C_L + C_S$$

where C_L includes confounders in the language and C_S includes confounders from social factors.

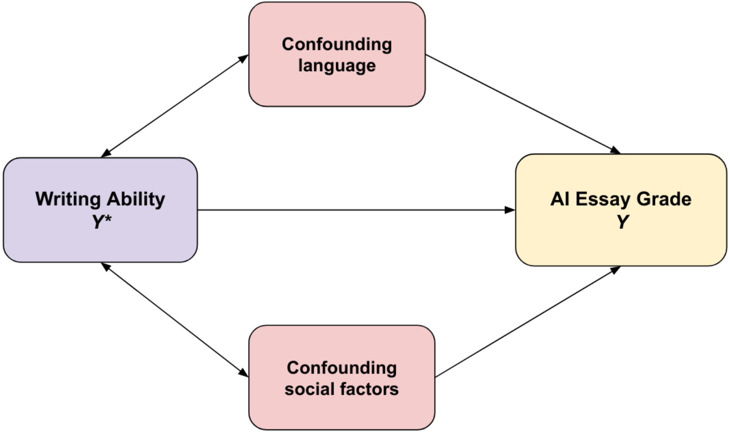
Ben Zimmer @bgzimmer · 2 Jul 2018

This gobbledygook earns a perfect grade from the GRE's automated essay scoring system. Algorithms writing for algorithms. npr.org/2018/06/30/624…

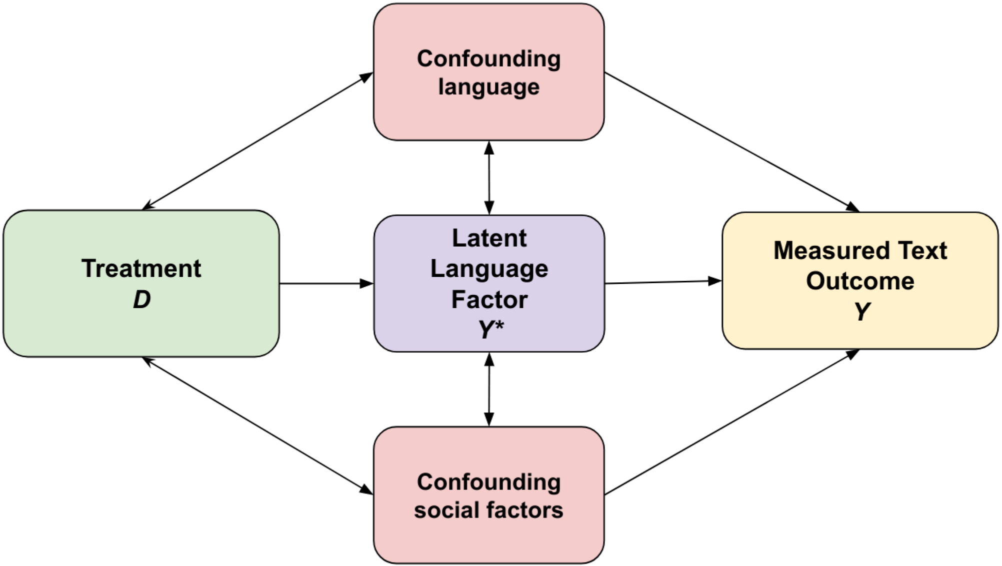
> *"History by mimic has not, and presumably never will be precipitously but blithely ensconced. Society will always encompass imaginativeness; many of scrutinizations but a few for an amanuensis. The perjured imaginativeness lies in the area of theory of knowledge but also the field of literature. Instead of enthralling the analysis, grounds constitutes both a disparaging quip and a diligent explanation."*

51 636 1.1K

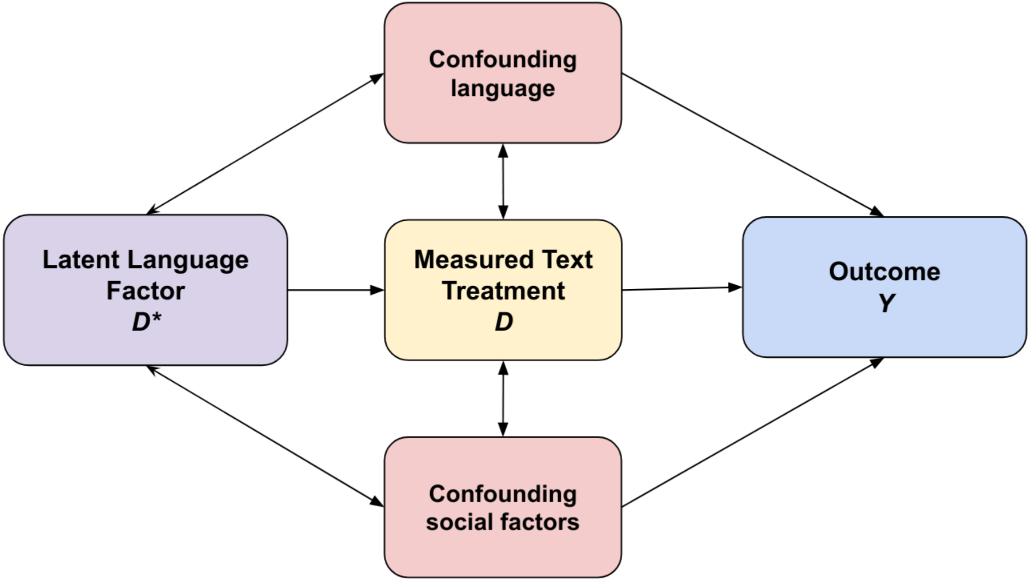
(When) is this a problem?



- What are some confounding language and confounding social factors for automated essay grades?

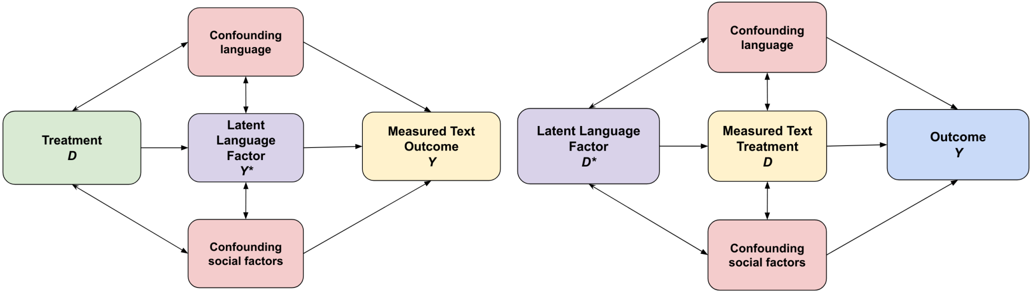


- effect of diversity training on prejudiced attitudes
- effect of writing prep class on writing ability.



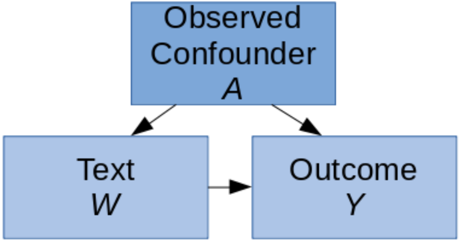
- effect of prejudicial attitudes on judge decisions
- effect of writing ability on career income.

Causal inference with text is hard



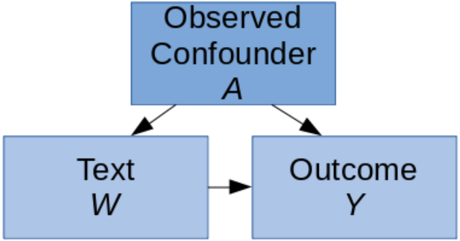
- There could be confounders both with the true treatment/outcome, as well as with the text-based measure of the treatment/outcome.

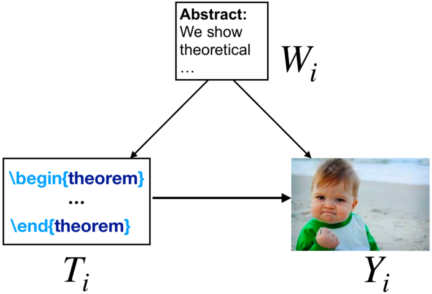
Sridhar, Veitch, and Blei (2019): Adjusting for Confounders with BERT



- This paper analyzes the problem of the effect of text features on outcomes, where the unobserved confounders are other features of the document.

Sridhar, Veitch, and Blei (2019): Adjusting for Confounders with BERT





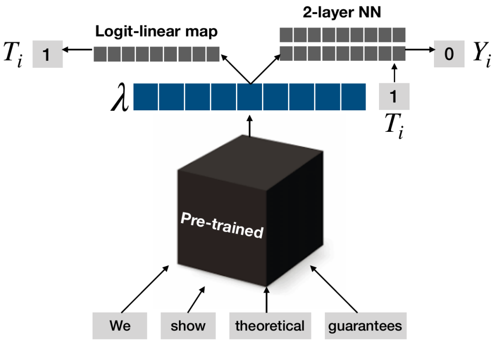
- This paper analyzes the problem of the effect of text features on outcomes, where the unobserved confounders are other features of the document.
- For example, the effect of putting a theorem in your paper on acceptance to a conference/journal.
- This paper is another example of controlling for observed confounders, but in high dimensions.

Sridhar, Veitch, and Blei (2019): Approach

- Insight: the confounding part of the text is that which carries information about both treatment and outcome.

Sridhar, Veitch, and Blei (2019): Approach

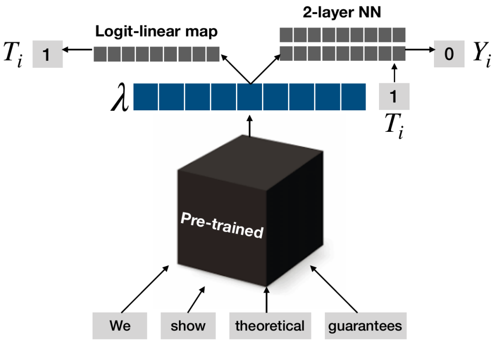
- Insight: the confounding part of the text is that which carries information about both treatment and outcome.



- Start with pre-trained BERT embeddings.
- fine-tune them on an additional multitask objective.

Sridhar, Veitch, and Blei (2019): Approach

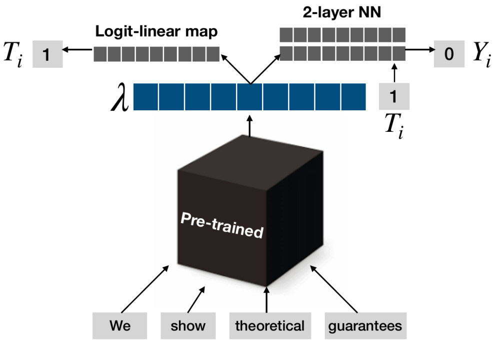
- Insight: the confounding part of the text is that which carries information about both treatment and outcome.



- Start with pre-trained BERT embeddings.
- fine-tune them on an additional multitask objective.
- 1) predict propensity score (probability of treatment given other text features.
- 2) predict outcomes conditional on treatment.

Sridhar, Veitch, and Blei (2019): Approach

- Insight: the confounding part of the text is that which carries information about both treatment and outcome.



- Start with pre-trained BERT embeddings.
- fine-tune them on an additional multitask objective.
- 1) predict propensity score (probability of treatment given other text features.
- 2) predict outcomes conditional on treatment.
- the resulting embeddings serve as a sufficient statistic for the unobserved confounders.