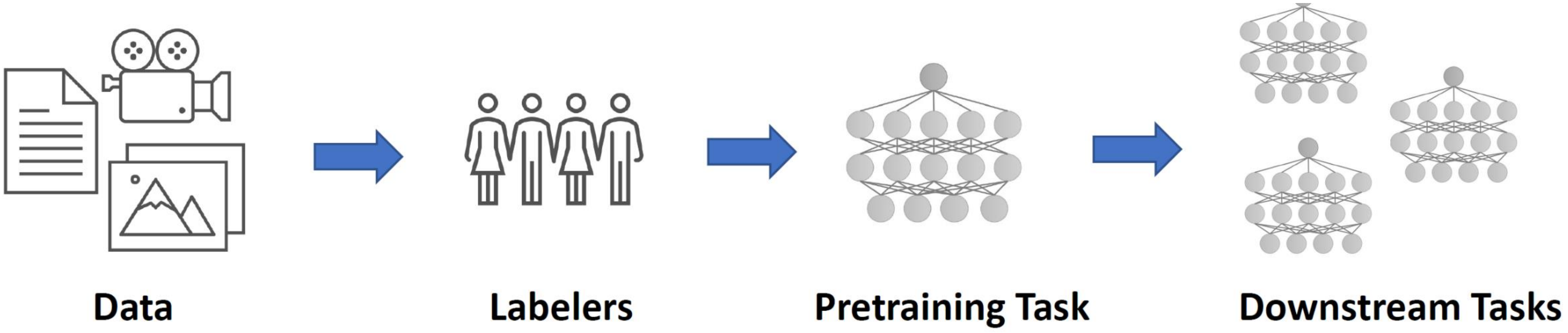
《多模态机器学习》

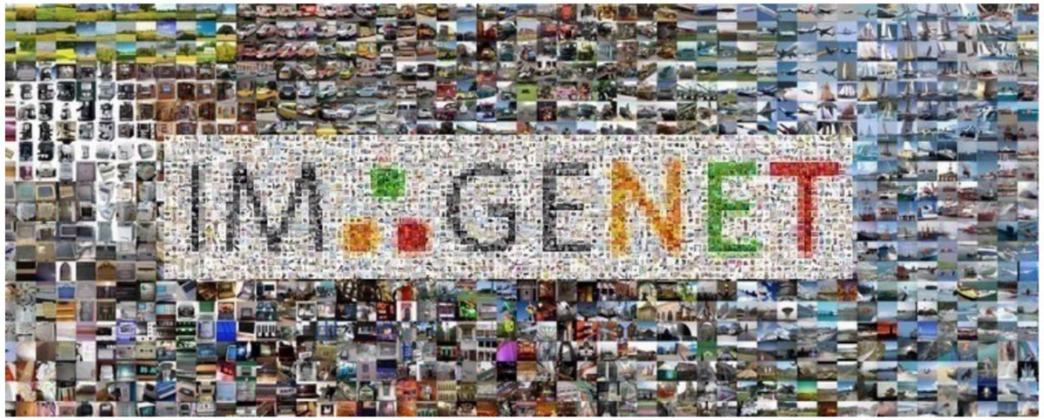
第八章 多模态自监督学习

黄文炳
中国人民大学高瓴人工智能学院
hwenbing@126.com

2024年秋季

Supervised pretraining on large labeled, datasets has led to successful transfer learning



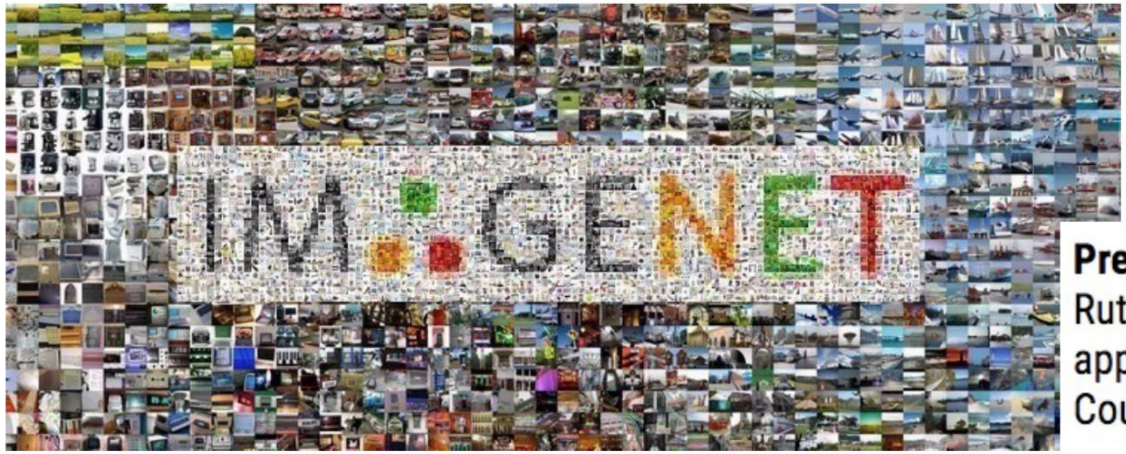


[Deng et al., 2009]

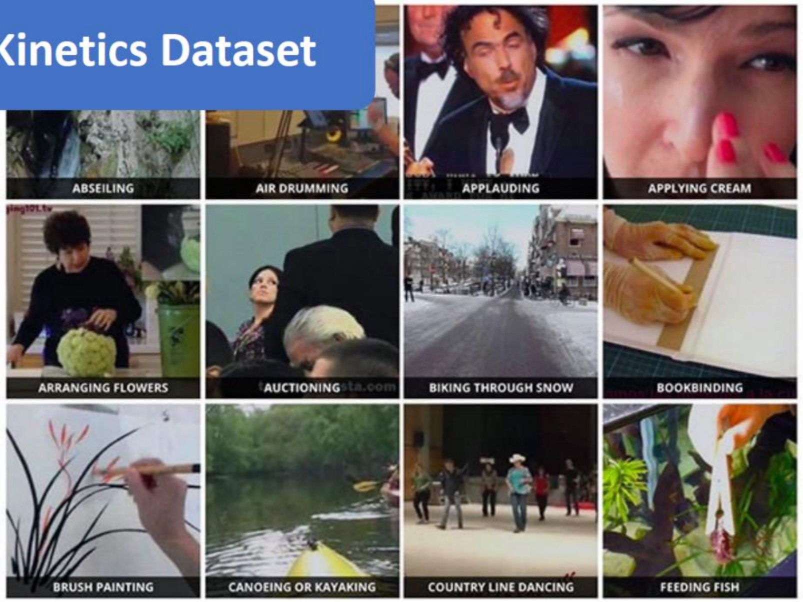
ImageNet

- Pretrain for fine-grained image classification over 1000 classes
- Use feature representations for downstream tasks, e.g. object detection, image segmentation, and action recognition

Supervised pretraining on large labeled, datasets has led to successful transfer learning



Kinetics Dataset



SNLI Dataset

Premise:
Ruth Bader Ginsburg being appointed to the US Supreme Court.

Hypothesis:
A grilled sandwich on a plate.

Label:
Contradiction [different scenes]

Across images, video, and text

[Deng et al., 2009] [Carreira et al., 2017] [Conneau et al., 2017]

But supervised pretraining comes at a cost…

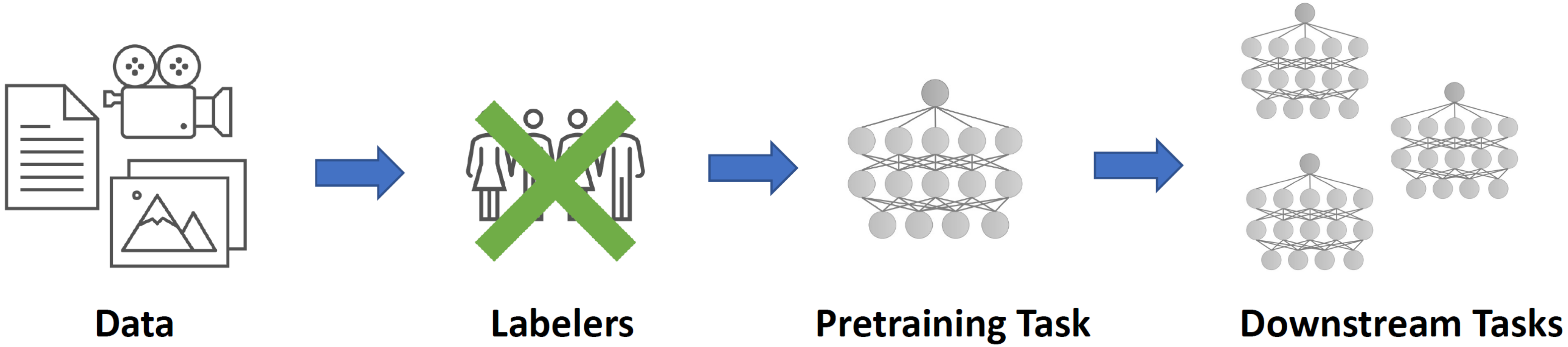
- **Time-consuming and expensive** to label datasets for new tasks
  - ImageNet: 3 years,
  49k Amazon MechanicalTurkers [1]

- **Domain expertise needed** for specialized tasks
  - Radiologists to label medical images
  - Native speakers or language specialists for labeling text in different languages

Can self-supervised learning help?

- Self-supervised learning (informal definition): supervise using labels ***generated from the data*** without any manual or weak label sources

- Idea: Hide or modify part of the input. Ask model to recover input or classify what changed.
  - Self-supervised task referred to as the pretext task



Pretext Task: Classify the Rotation

270° rotation

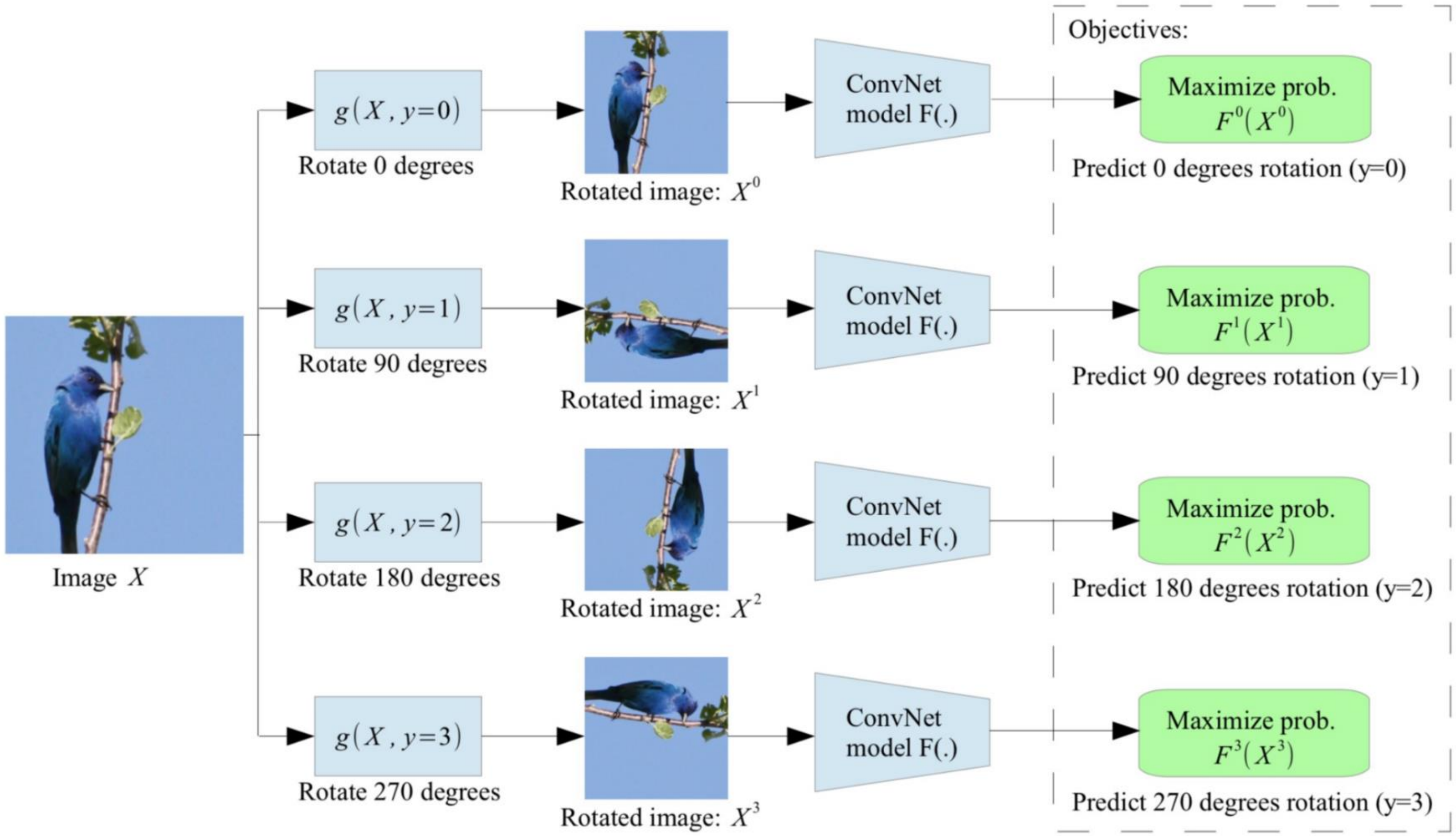
90° rotation

~~180°~~ 0° rotation

Catfish species that swims upside down…

Identifying the object helps solve rotation task!

Pretext Task: Classify the Rotation



Learning rotation improves results on object classification, object segmentation, and object detection tasks.

[Gidaris et al., ICLR 2018]

https://arxiv.org/abs/1803.07728

Benefits of Self-Supervised Learning

- ✓ Like supervised pretraining, can learn general-purpose feature representations for downstream tasks
- ✓ Reduces expense of hand-labeling large datasets
- ✓ Can leverage nearly unlimited (unlabeled) data available on the web

995 photos uploaded *every second*

6000 tweets sent *every second*

500 hours of video uploaded *every minute*

Today's Plan

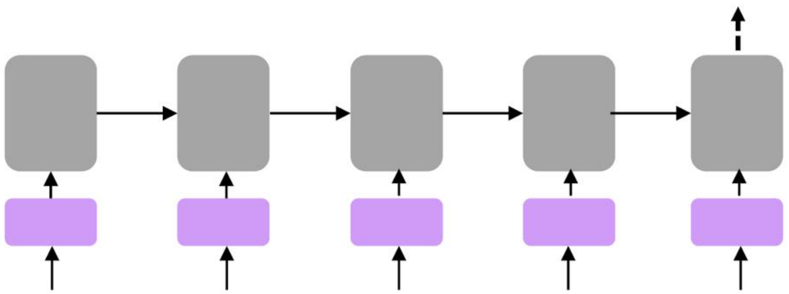
1. What is self-supervised learning?

2. Examples of self-supervision in NLP
   - Word embeddings (e.g., word2vec)
   - Language models (e.g., GPT)
   - Masked language models (e.g., BERT)

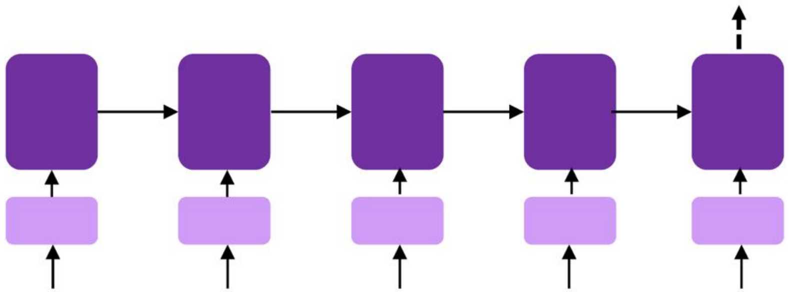
3. Open challenges
   - Demoting bias
   - Capturing factual knowledge
   - Learning symbolic reasoning

Examples of Self-Supervision in NLP

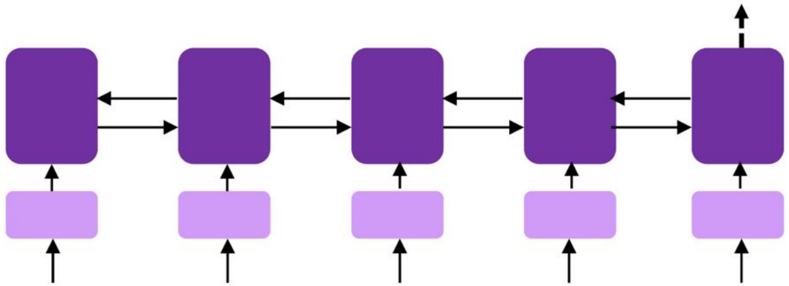
- **Word embeddings**
  - Pretrained word representations
  - Initializes *1st layer* of downstream models



- **Language models**
  - *Unidirectional*, pretrained language representations
  - Initializes *full* downstream model

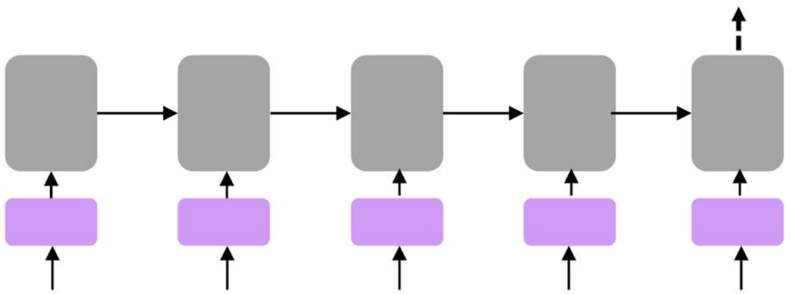


- **Masked language models**
  - *Bidirectional*, pretrained language representations
  - Initializes *full* downstream model

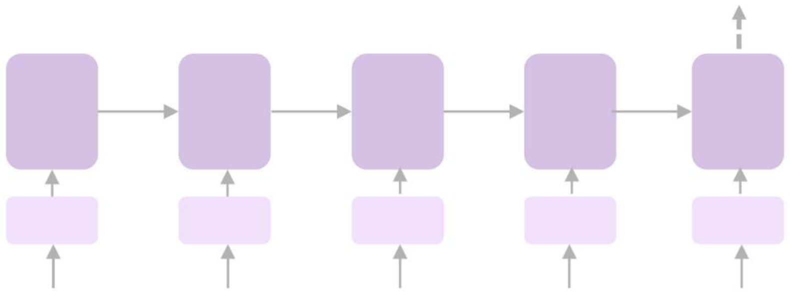


Examples of Self-Supervision in NLP

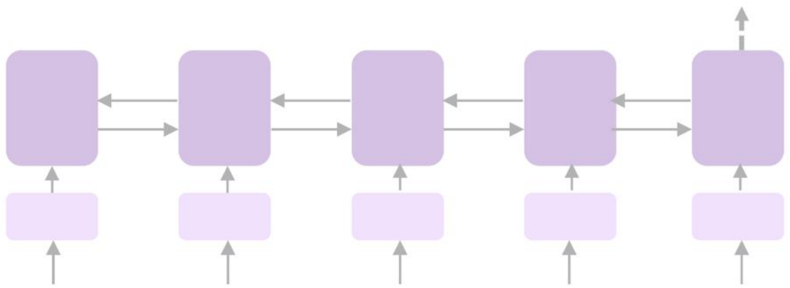
- **Word embeddings**
  - Pretrained word representations
  - Initializes *1st layer* of downstream models



- **Language models**
  - *Unidirectional*, pretrained language representations
  - Initializes *full* downstream model



- **Masked language models**
  - *Bidirectional*, pretrained language representations
  - Initializes *full* downstream model



Word Embeddings

- Goal: represent words as vectors for input into neural networks.

- One-hot vectors? (single 1, rest 0s)

  pizza = [0 0 0 0 0 1 0 … 0 0 0 0 0]

  pie = [0 0 0 0 0 0 0 … 0 0 0 1 0]

☹ Millions of words ⟶ high-dimensional, sparse vectors

☹ No notion of word similarity

- Instead: we want a **dense, low-dimensional** vector for each word such that words with similar meanings have similar vectors.

Distributional Semantics

- Idea: define a word by the words that frequently occur nearby in a corpus of text
  - "You shall know a word by the company it keeps" (J. R. Firth 1957: 11)

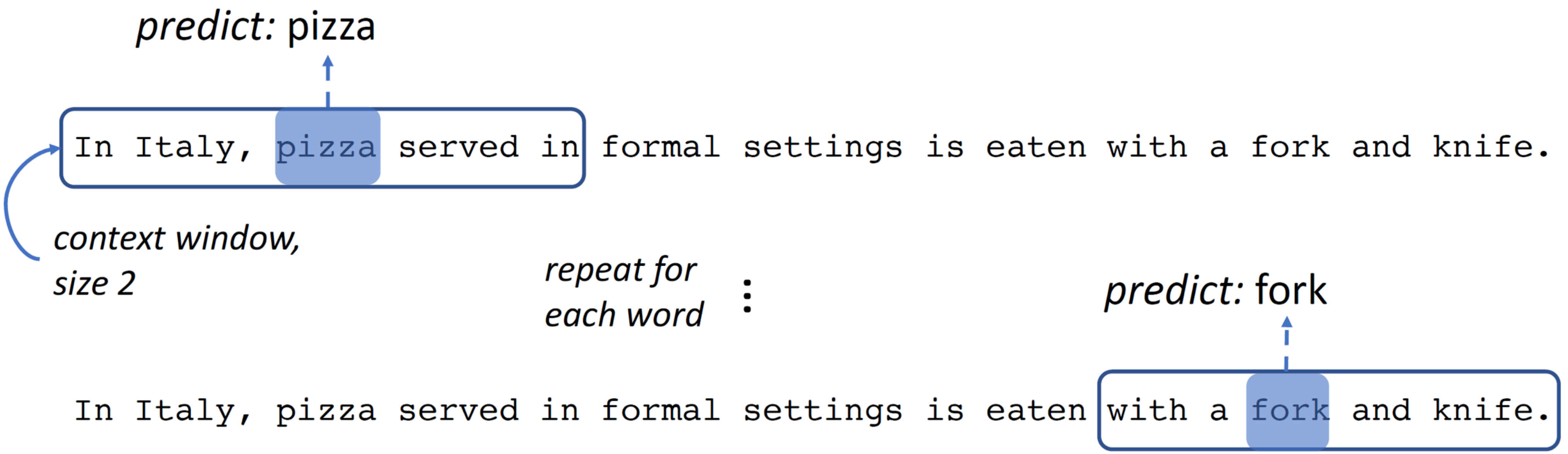
- Example: defining "pizza"
  - What words frequently occur in the context of pizza?

```
13% of the United States population eats pizza on any given day.
Mozzarella is commonly used on pizza, with the highest quality mozzarella from Naples.
In Italy, pizza served in formal settings is eaten with a fork and knife.
```

- Can we use distributional semantics to develop a pretext task for self-supervision?

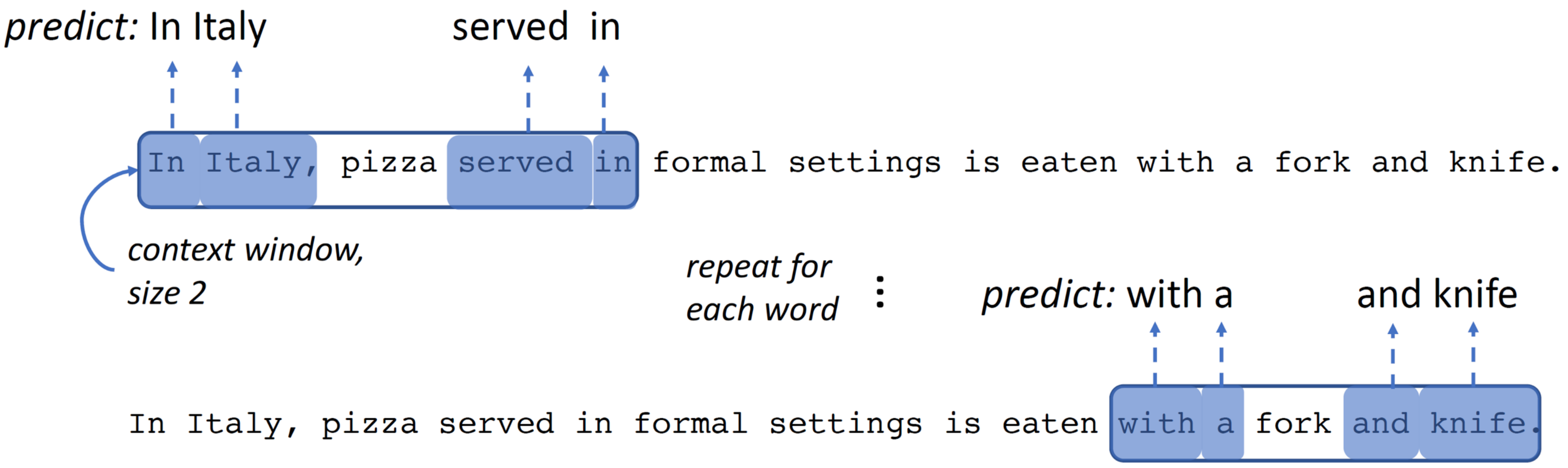
Pretext Task: Predict the Center Word

- Move context window across text data and use words in window to predict the center word.
  - No hand-labeled data is used!



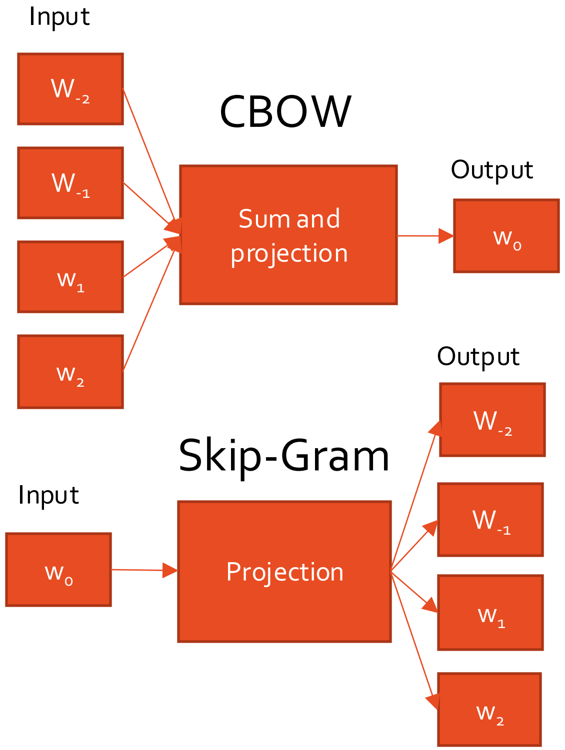
Pretext Task: Predict the Context Words

- Move context window across text data and use words in window to predict the *context* words, given the center word.
  - No hand-labeled data is used!



Case Study: word2vec

- Tool to produce word embeddings using self-supervision by Mikolov et al.
- Supports training word embeddings using 2 architectures:
  - Continuous bag-of-words (CBOW): predict the center word
  - Skip-gram: predict the context words
- Steps:
  1. Start with randomly initialized word embeddings.
  2. Move sliding window across *unlabeled* text data.
  3. Compute probabilities of center/context words, given the words in the window.
  4. Iteratively update word embeddings via stochastic gradient descent .



https://papers.nips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

Case Study: word2vec

- **Loss function (skip-gram):** For a corpus with T words, minimize the negative log likelihood of the context word w_{t+j} given the center word w_t.

Context word Center word

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m \le j \le m \\ j \ne 0}} \log P\left(w_{t+j} \mid w_t;\ \theta\right)$$

Context window size Model parameters

- Use two word embedding matrices (embedding dimension n, vocab size l):
  - Center word embeddings $V \in \mathbb{R}^{n\times l}$; context word embeddings $U \in \mathbb{R}^{l\times n}$

$$P\left(w_{t+j} \mid w_t;\ \theta\right) = P\left(u_{t+j} \mid v_t\right) = \frac{\exp(u_{t+j}^T v_t)}{\sum_{j=1}^{l}\exp(u_j^T v_t)}$$

Softmax

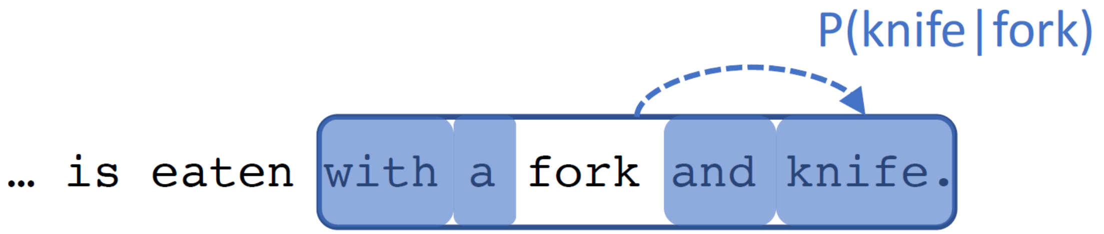
Word vectors

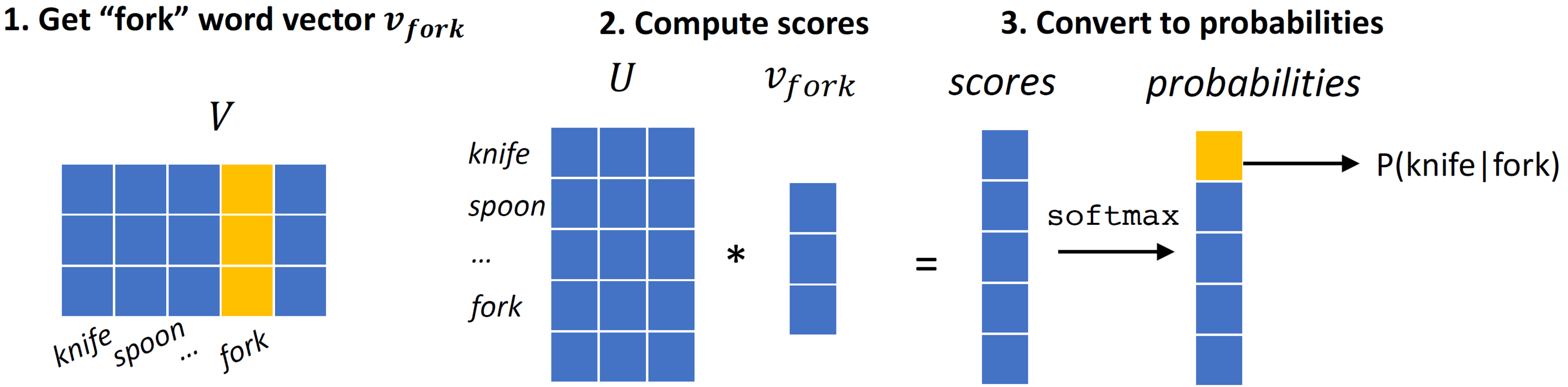
[Mikolov et al., 2013]

https://papers.nips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

Case Study: word2vec

- **Example:** using the skip-gram method (predict context words), compute the probability of ”knife” given the center word “fork”.





https://papers.nips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

Skip-gram with Negative Sampling

- Let's see where the complexity is:

$P(o \mid c)$

into $\quad$ banking

$v_c \quad u_o$

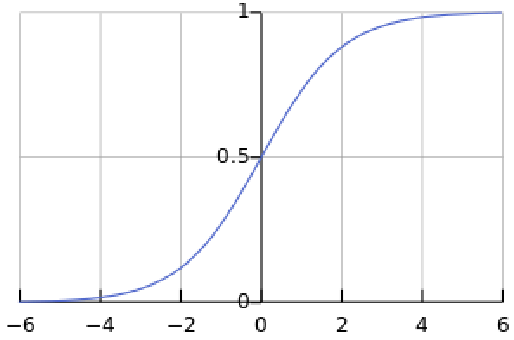
The expensive computation: O(|V|.d)

$$\log P(o|c) = \log \frac{\exp(u_o^T v_c)}{\sum_{x \in V} \exp(u_x^T v_c)} = \log \exp(u_o^T v_c) - \log \overbrace{\sum_{x \in V} \exp(u_x^T v_c)}$$

- **Idea:** rather than enumerating over all vocabulary, let's sample!

$$J_{NS}(\theta) = -\log \sigma(u_o^T v_c) - \sum_{k \in \{K\ samples\}} \log \sigma(-u_x^T v_c)$$

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



- Maximize the prob that outside word co-occurs w/ the center
- Minimize the prob of noise/random words far from the center (negatives)

Skip-gram with Negative Sampling

- Have to be careful with sampling negative examples
  - Challenge: uniform sampling will sample a lot of stop-words that are very popular.

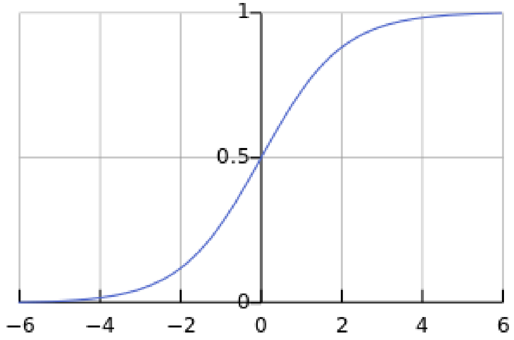
- Mikolov et al. proposed to sample: $p(w_i) = {f(w_i)^{3/4}} \Big/ {\sum_j f(w_j)^{3/4}}$
  - Assigns more prob to less frequent words. No theory backing, but works!

- **Idea:** rather than enumerating over all vocabulary, let's sample!

$$J_{NS}(\theta) = -\log \sigma(u_o^T v_c) - \sum_{k \in \{K\ samples\}} \log \sigma(-u_x^T v_c)$$

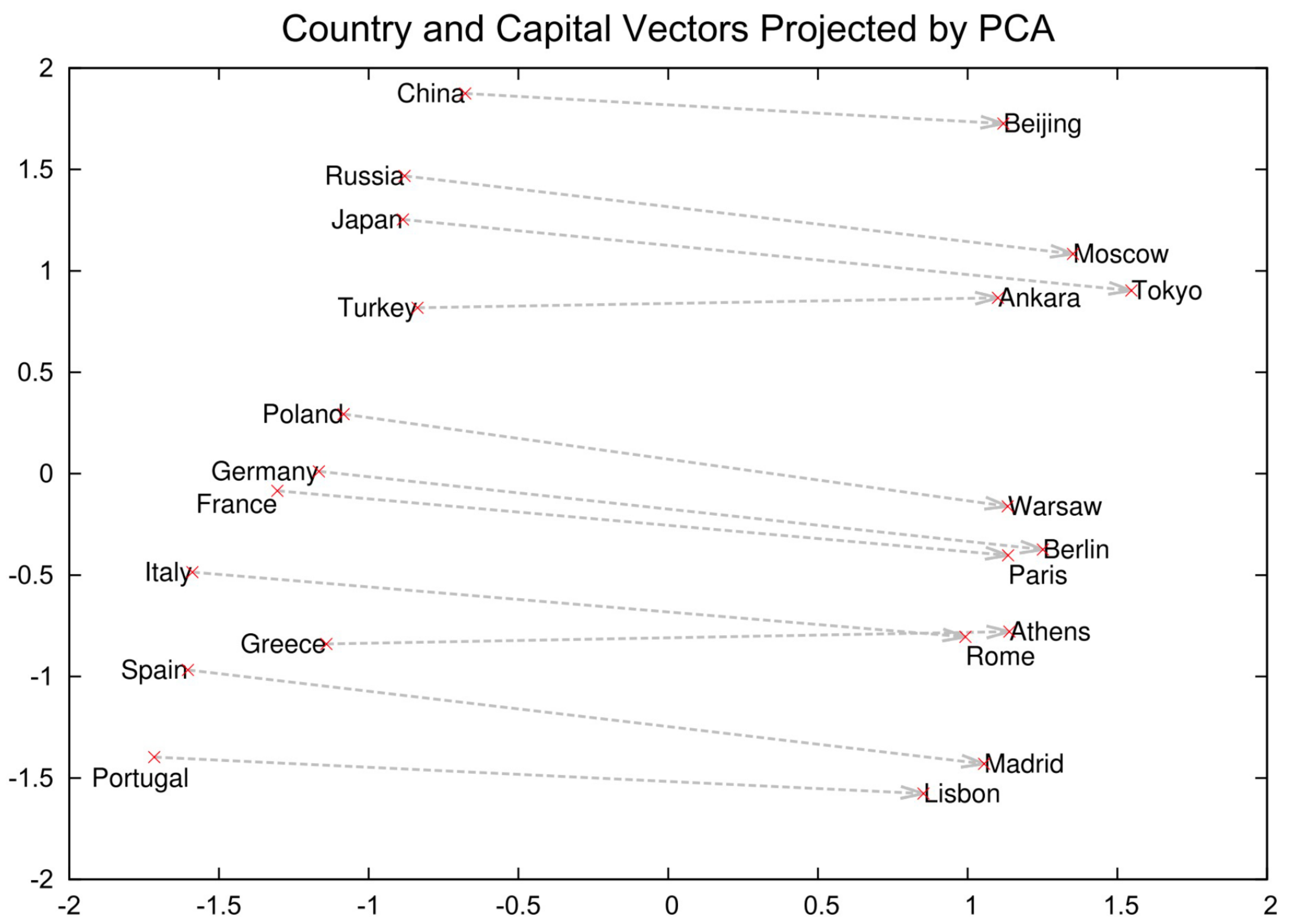
  - Maximize the prob that outside word co-occurs w/ the center
  - Minimize the prob of noise/random words far from the center (negatives)

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Case Study: word2vec

- Mikolov et al. released word2vec embeddings pretrained on **100 billion word** Google News dataset.
- Embeddings exhibited meaningful properties despite being trained with **no hand-labeled data**.

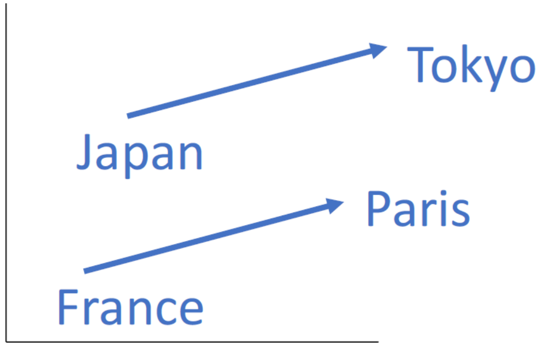


https://papers.nips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

Case Study: word2vec

- Vector arithmetic can be used to evaluate word embeddings on analogies

  - France is to Paris as Japan is to ?



Cosine similarity

$$w^* = argmax_w \frac{\boldsymbol{v_w y}}{\|\boldsymbol{v_w}\|\|\boldsymbol{y}\|},$$

where $\boldsymbol{y} = \boldsymbol{v_{Paris}} - \boldsymbol{v_{France}} + \boldsymbol{v_{Japan}}$

w^* = Tokyo

Expected answer

- Analogies have become a common **intrinsic task** to evaluate the properties learned by word embeddings

https://papers.nips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

[Efficient Estimation of Word Representations in Vector Space. Mikolov et al. 2013]

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

[Efficient Estimation of Word Representations in Vector Space. Mikolov et al. 2013]

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

[Efficient Estimation of Word Representations in Vector Space. Mikolov et al. 2013]

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

[Efficient Estimation of Word Representations in Vector Space. Mikolov et al. 2013]

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

[Efficient Estimation of Word Representations in Vector Space. Mikolov et al. 2013]

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

[Efficient Estimation of Word Representations in Vector Space. Mikolov et al. 2013]

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

[Efficient Estimation of Word Representations in Vector Space. Mikolov et al. 2013]

Mismatch Between Cosine and Dot Product

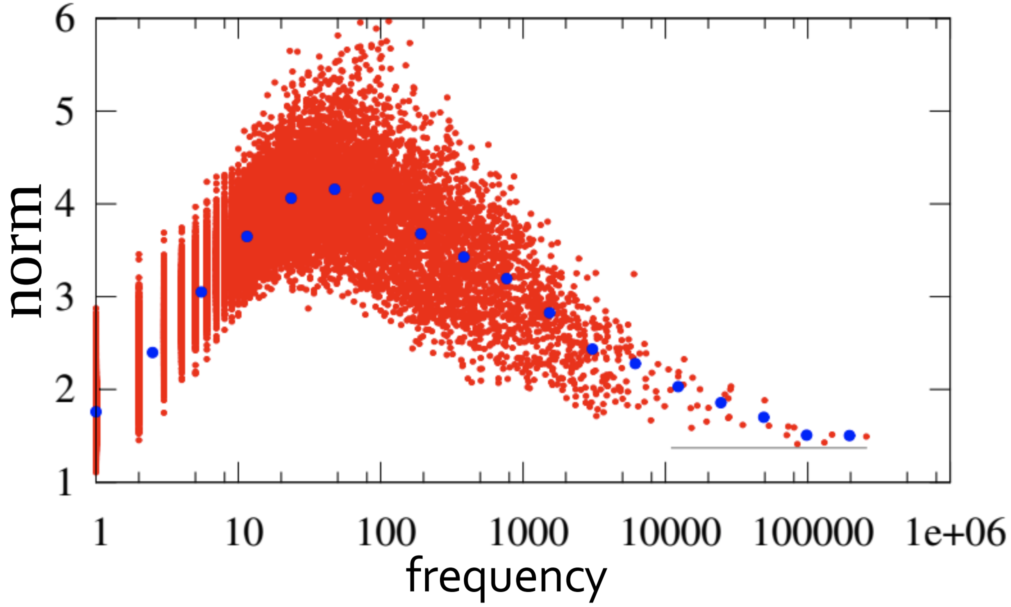
- **Observation:** there a mismatch between Word2Vec objective and cosine distance!

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{x \in V} \exp(u_x^T v_c)}$$

$$\text{distance}(\mathrm{x}, \mathrm{y}) = \cos(v_{\mathrm{x}}, v_{\mathrm{y}}) = \frac{v_{\mathrm{x}}^T v_{\mathrm{y}}}{\|v_{\mathrm{y}}\| \|v_{\mathrm{x}}\|}$$

1. Why use cosine distance instead of dot product?
    - Term frequencies affect the embedding norms.
    - Without normalization, frequent terms would seem more similar.

2. Why not change W2V objective to use cos?
    - ¯_(ツ)_/¯
    - It's possible that the resulting vectors would conflate semantic similarity and frequency.



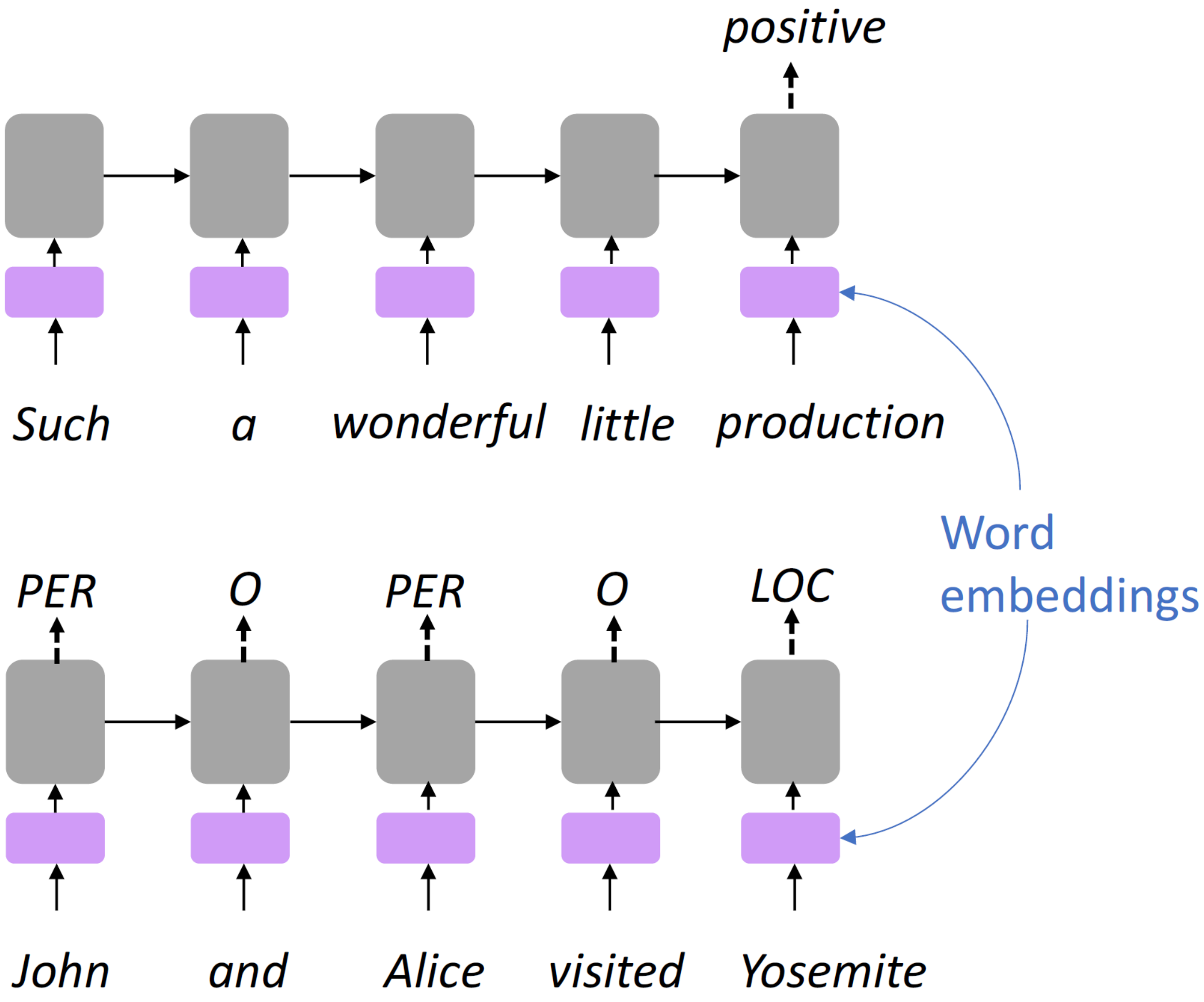
[Measuring Word Significance using Distributed Representations of Words]

Case Study: word2vec

- Pretrained word2vec embeddings can be used to initialize the first layer of downstream models

- Improved performance on many downstream NLP tasks, including sentence classification, machine translation, and sequence tagging
  - Most useful when downstream data is limited

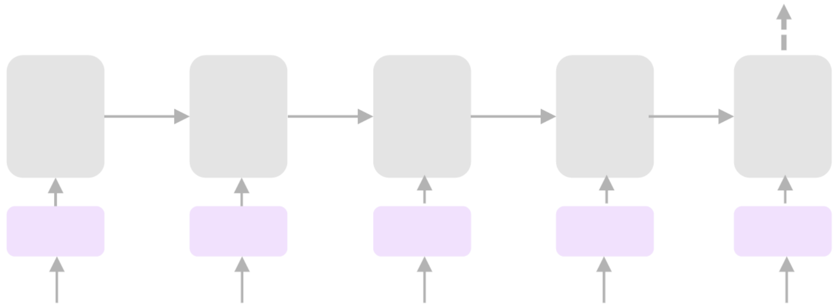
- Still being used in applications in industry today!



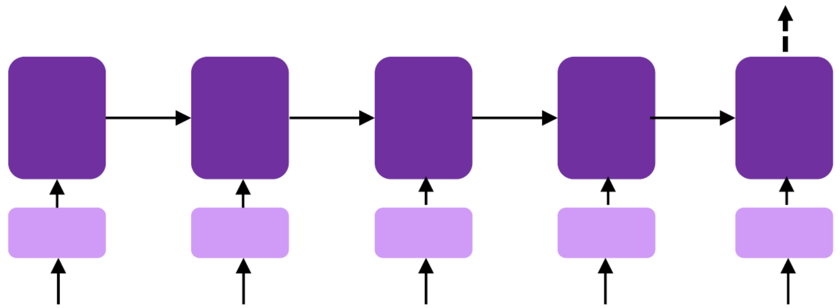
https://arxiv.org/pdf/1408.5882 https://www.aclweb.org/anthology/N18-2084.pdf https://arxiv.org/pdf/1603.01360

Examples of Self-Supervision in NLP

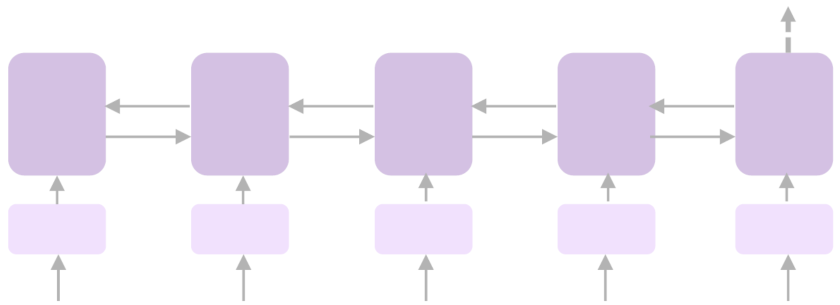
- **Word embeddings**
  - Pretrained word representations
  - Initializes *1st layer* of downstream models



- **Language models**
  - *Unidirectional*, pretrained language representations
  - Initializes *full* downstream model



- **Masked language models**
  - *Bidirectional*, pretrained language representations
  - Initializes *full* downstream model

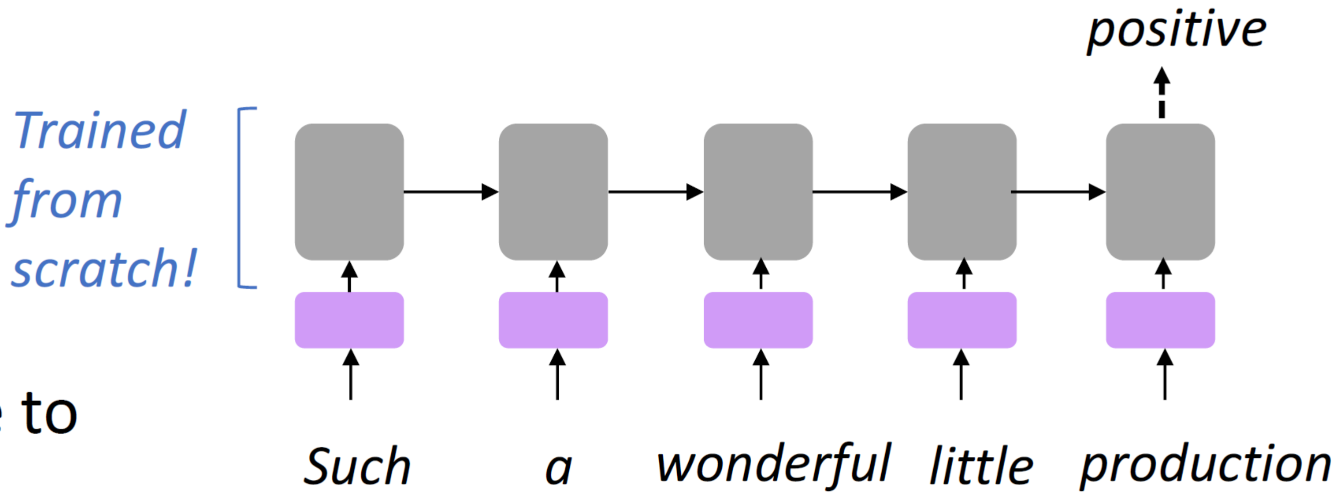


Why weren't word embeddings enough?

- Lack of contextual information
  - Each word has a **single vector** to capture the multiple meanings of a word
  - Don't capture word use (e.g. syntax)

The **ship** is used to **ship** packages.

- Most of the downstream model still needs training



- What self-supervised tasks can we use to pretrain full models for contextual understanding?

The

The cat

The cat sat

The cat sat on

The cat sat on __?__

The cat sat on the mat.

The cat sat on the mat.

$$\mathbf{P}(\underbrace{\text{mat}}_{\text{next word}} \mid \underbrace{\text{The cat sat on the}}_{\text{context or prefix}})$$

Probability of Upcoming Word

$$\mathbf{P}(\underbrace{X_t}_{\text{next word}} \mid \underbrace{X_1, \ldots, X_{t-1}}_{\text{context or prefix}})$$

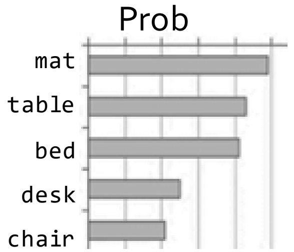
LMs as a Marginal Distribution

- Directly we train models on “marginals”:

$$\mathbf{P}(\overbrace{X_t}^{\text{next word}} | \overbrace{X_1, \ldots, X_{t-1}}^{\text{context}})$$

“The cat sat on the [MASK]”

Some model



LMs as Implicit Joint Distribution of Language

- Though implicitly we are learning the full distribution over the language:
  - Remember the chain rule: $\mathbf{P}(X_1, \dots, X_t) = \mathrm{P}(X_1) \prod_{i=1}^{t} \mathrm{P}(X_i \mid X_1, X_2 \dots, X_i)$

- Language Modeling $\triangleq$ learning prob distribution over language sequence.

Doing Things with Language Model

- What is the probability of …. "I like Johns Hopkins University"

  "like Hopkins I University Johns"

- LMs assign a probability to every sentence (or any string of words).

P("I like Johns Hopkins University EOS") $=10^{-5}$

P("like Hopkins I University Johns EOS") $=10^{-15}$

Doing Things with Language Model (2)

- We can rank sentences.

$$\mathbf{P}(\underbrace{X_t}_{\text{next word}} \mid \underbrace{X_1, \ldots, X_{t-1}}_{\text{context}})$$

- While LMs show “typicality”, this may be a proxy indicator to other properties:
  - Grammaticality, fluency, factuality, etc.

P(*“I like Johns Hopkins University. EOS”*) > **P**(*“I like John Hopkins University EOS”*)

P(*“I like Johns Hopkins University. EOS”*) > **P**(*“University. I Johns EOS Hopkins like”*)

P(*“JHU is located in Baltimore. EOS”*) > **P**(*“JHU is located in Virginia. EOS”*)

Doing Things with Language Model (3)

- Can also generate strings

$$\mathbf{P}(\underbrace{X_t}_{\text{next word}} | \underbrace{X_1, \ldots, X_{t-1}}_{\text{context}})$$

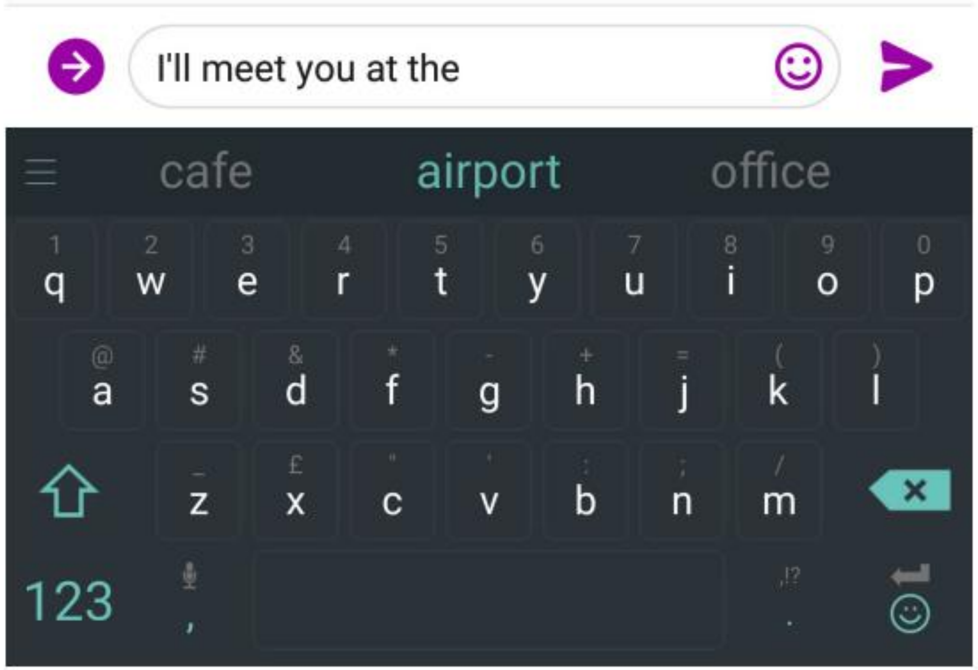
- Let's say we start *"Johns Hopkins is "*
- Using this prompt as initial condition, recursively sample from an LM:

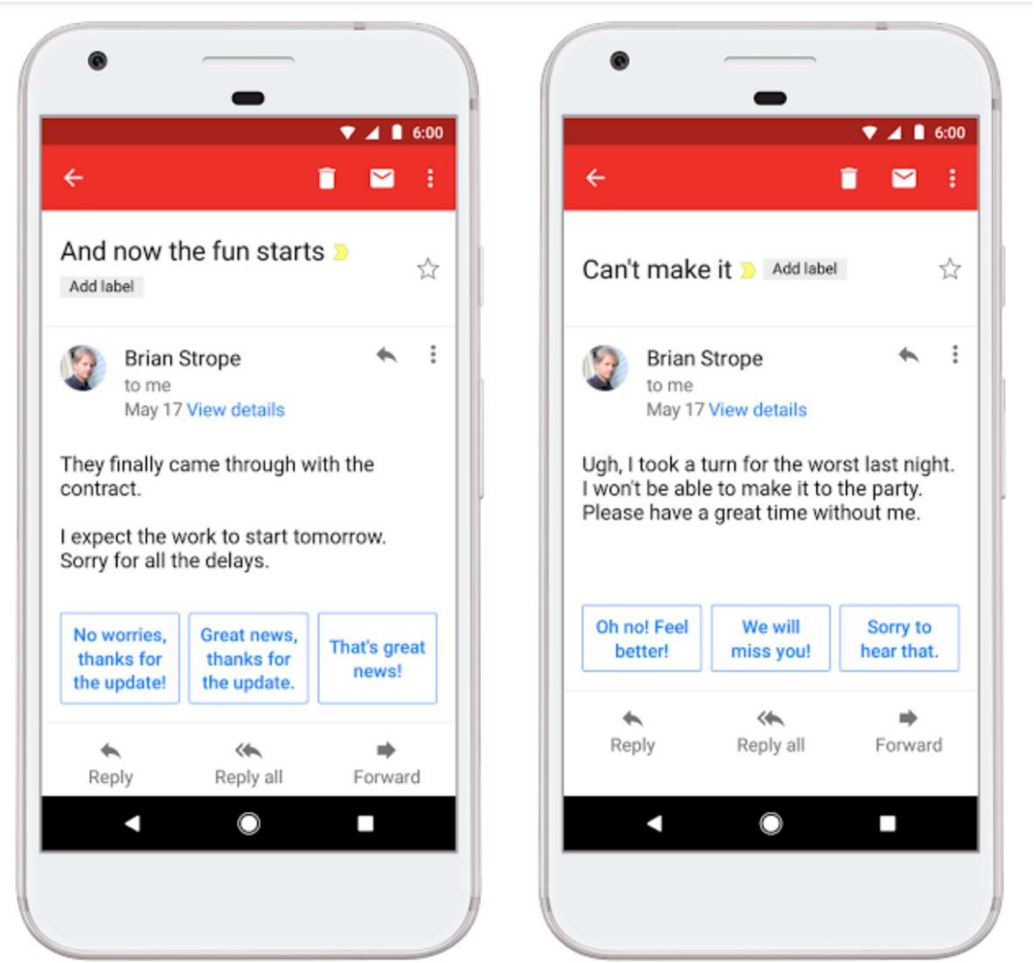
1. Sample from **P**(X| *"Johns Hopkins is "*) → "located"
2. Sample from **P**(X| *"Johns Hopkins is located"*) → "at"
3. Sample from **P**(X| *"Johns Hopkins is located at"*) → "the"
4. Sample from **P**(X| *"Johns Hopkins is located at the"*) → "state"
5. Sample from **P**(X| *"Johns Hopkins is located at the state"*) → "of"
6. Sample from **P**(X| *"Johns Hopkins is located at the state of"*) → "Maryland"
7. Sample from **P**(X| *"Johns Hopkins is located at the state of Maryland"*) → "EOS"

Why Should We Care About Language Modeling?

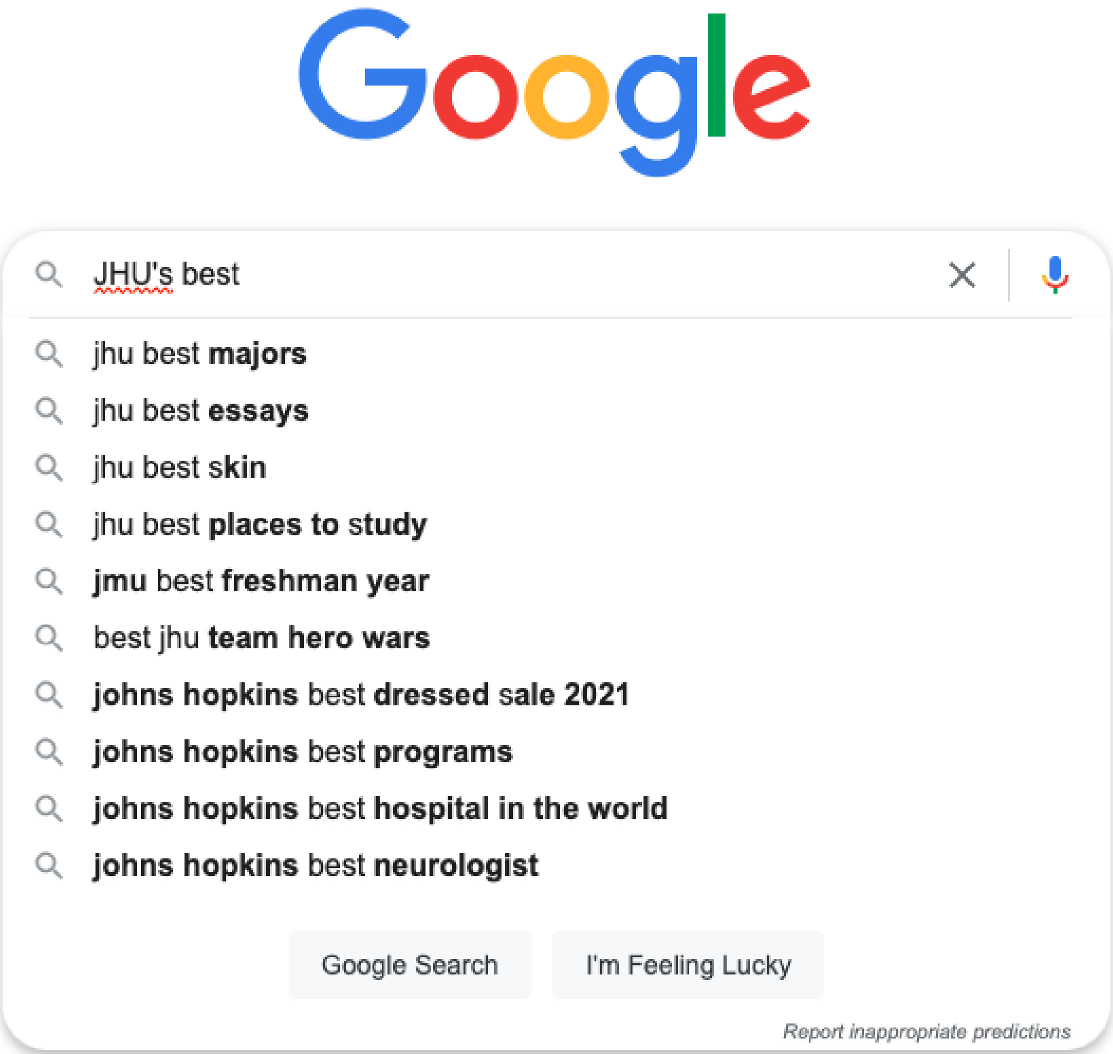
- Language Modeling is an effective proxy for language understanding.
  - Effective ability to predict forthcoming words rely on understanding of context/prefix
- Language Modeling is a ~~subcomponent~~ superset of many NLP tasks, especially those involving text generation:
  - Summarization
  - Machine translation
  - Spelling correction
  - Dialogue etc.

You use Language Models every day!

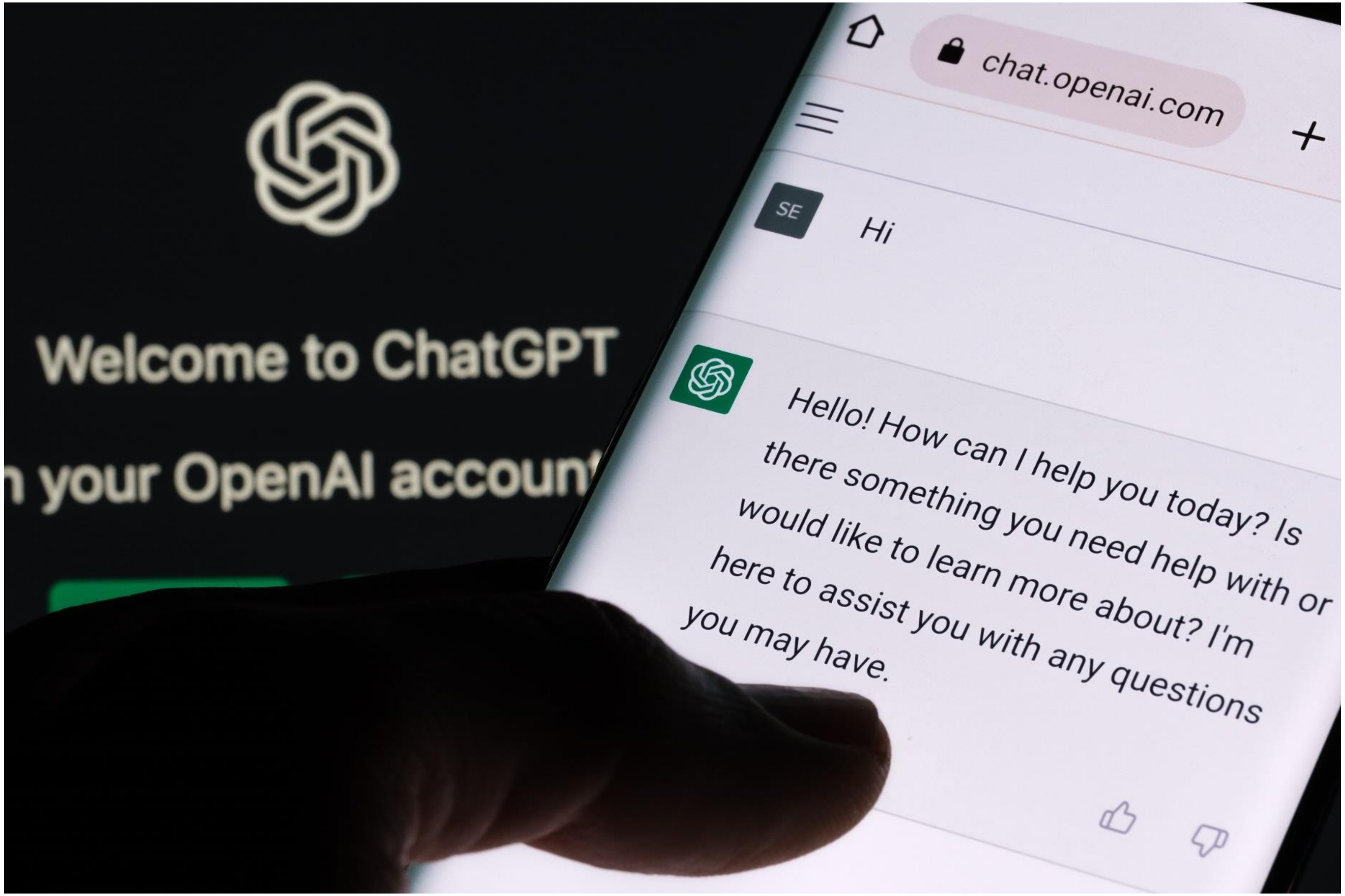




You use Language Models every day!



You use Language Models every day!



It Can be Misused Too …

Is this a real science article?

- A lot more about harms later in the class.

Rooter: A Methodology for the Typical Unification of Access Points and Redundancy

Jeremy Stribling, Daniel Aguayo and Maxwell Krohn

ABSTRACT

Many physicists would agree that, had it not been for congestion control, the evaluation of web browsers might never have occurred. In fact, few hackers worldwide would disagree with the essential unification of voice-over-IP and public-private key pair. In order to solve this riddle, we confirm that SMPs can be made stochastic, cacheable, and interposable.

I. INTRODUCTION

Many scholars would agree that, had it not been for active networks, the simulation of Lamport clocks might never have occurred. The notion that end-users synchronize with the investigation of Markov models is rarely outdated. A theoretical grand challenge in theory is the important unification of virtual machines and real-time theory. To what extent can web browsers be constructed to achieve this purpose?

Certainly, the usual methods for the emulation of Smalltalk that paved the way for the investigation of rasterization do not apply in this area. In the opinions of many, despite the fact that conventional wisdom states that this grand challenge is continuously answered by the study of access points, we believe that a different solution is necessary. It should be noted that Rooter runs in $\Omega(\log \log n)$ time. Certainly, the shortcoming of this type of solution, however, is that compilers and superpages are mostly incompatible. Despite the fact that similar methodologies visualize XML, we surmount this issue without synthesizing distributed archetypes.

The rest of this paper is organized as follows. For starters, we motivate the need for fiber-optic cables. We place our work in context with the prior work in this area. To address this obstacle, we disprove that even though the much-tauted autonomous algorithm for the construction of digital-to-analog converters by Jones [10] is NP-complete, object-oriented languages can be made signed, decentralized, and signed. Along these same lines, to accomplish this mission, we concentrate our efforts on showing that the famous ubiquitous algorithm for the exploration of robots by Sato et al. runs in $\Omega((n + \log n))$ time [22]. In the end, we conclude.

II. ARCHITECTURE

Our research is principled. Consider the early methodology by Martin and Smith; our model is similar, but will actually overcome this grand challenge. Despite the fact that such a claim at first glance seems unexpected, it is buffetted by previous work in the field. Any significant development of secure theory will clearly require that the acclaimed real-time algorithm for the refinement of write-ahead logging by Edward Feigenbaum et al. [15] is impossible; our application is no different. This may or may not actually hold in reality. We consider an application consisting of n access points. Next, the model for our heuristic consists of four independent components: simulated annealing, active networks, flexible modalities, and the study of reinforcement learning.

We consider an algorithm consisting of n semaphores. Any unproven synthesis of introspective methodologies will

https://pdos.csail.mit.edu/archive/scigen/

Language Models: A History

- Shannon (1950): The predictive difficulty (entropy) of English.

Prediction and Entropy of Printed English

By C. E. SHANNON

(Manuscript Received Sept. 15, 1950)

A new method of estimating the entropy and redundancy of a language is described. This method exploits the knowledge of the language statistics possessed by those who speak the language, and depends on experimental results in prediction of the next letter when the preceding text is known. Results of experiments in prediction are given, and some properties of an ideal predictor are developed.

[Prediction and Entropy of Printed English, Shanon 1950]

$$\mathbf{P}(X_t \mid X_1, \ldots, X_{t-1})$$


Andrey Markov

Shannon (1950) build an approximate language model with word co-occurrences.

Markov assumptions: every node in a Bayesian network is conditionally independent of its nondescendants, given its parents.

1st order approximation:

$$\mathbf{P}(\text{mat} \mid \text{the cat sat on the}) \approx \mathbf{P}(\text{mat} \mid \overbrace{\text{the}}^{1\ \text{element}})$$

[Prediction and Entropy of Printed English, Shanon 1950]

$$\mathbf{P}(X_t | X_1, \ldots, X_{t-1})$$

Andrey Markov

Shannon (1950) build an approximate language model with word co-occurrences.

Markov assumptions: every node in a Bayesian network is conditionally independent of its nondescendants, given its parents.

2nd order approximation:

$$\mathbf{P}(\text{mat} \mid \text{the cat sat on the}) \approx \mathbf{P}(\text{mat} \mid \underbrace{\text{on the}}^{2 \text{ elements}})$$

[Prediction and Entropy of Printed English, Shanon 1950]

$$\mathbf{P}(X_t | X_1, \dots, X_{t-1})$$

Andrey Markov

Shannon (1950) build an approximate language model with word co-occurrences.

Markov assumptions: every node in a Bayesian network is conditionally independent of its nondescendants, given its parents.

3rd order approximation:

$$\mathbf{P}(\text{mat} \mid \text{the cat sat on the}) \approx \mathbf{P}(\text{mat} \mid \overbrace{\text{sat on the}}^{3 \text{ elements}})$$

[Prediction and Entropy of Printed English, Shanon 1950]

$$\mathbf{P}(X_t | X_1, \ldots, X_{t-1})$$

Andrey Markov

Shannon (1950) build an approximate language model with word co-occurrences.

Then, we can use counts of approximate conditional probability.
Using the 3rd order approximation, we can:

$$\mathbf{P}(\text{mat} \mid \text{the cat sat on the}) \approx \mathbf{P}(\text{mat} \mid \text{sat on the}) = \frac{\text{count(“sat on the mat”)}}{\text{count(“on the mat”)}}$$

[Prediction and Entropy of Printed English, Shanon 1950]

N-gram Language Models

- **Terminology:** *n*-gram is a chunk of *n* consecutive words:
  - unigrams: “cat”, “mat”, “sat”, …
  - bigrams: “the cat”, “cat sat”, “sat on”, …
  - trigrams: “the cat sat”, “cat sat on”, “sat on the”, …
  - four-grams: “the cat sat on”, “cat sat on the”, “sat on the mat”, …

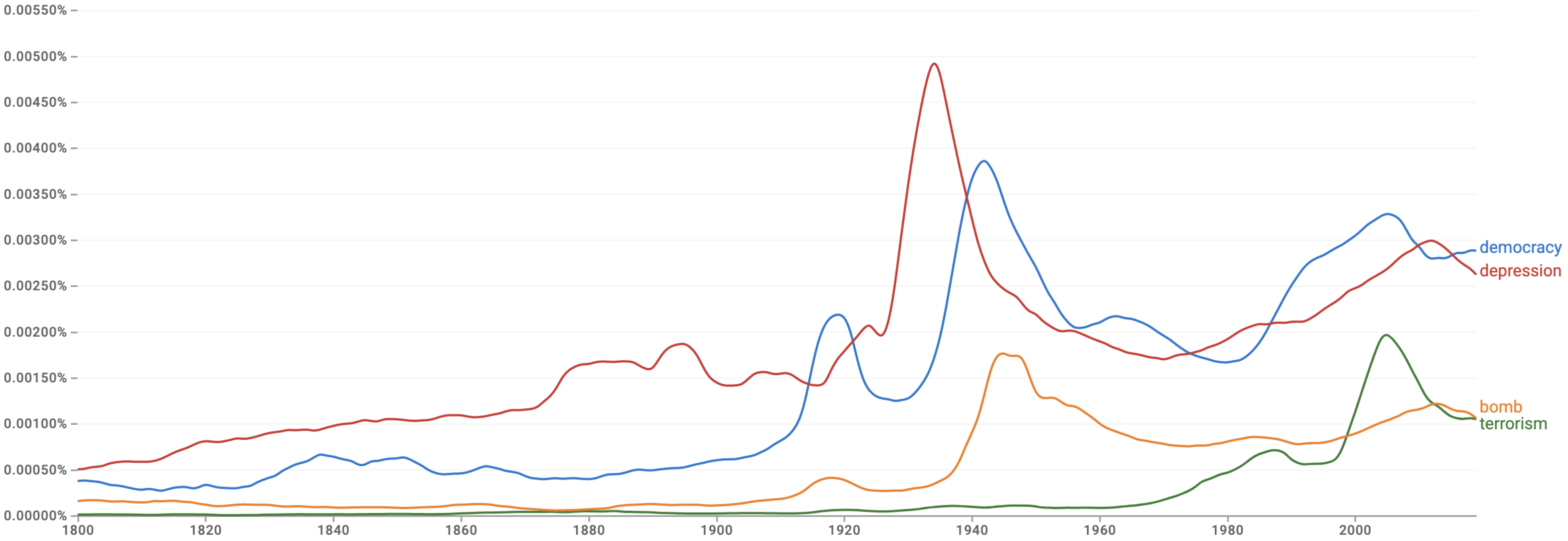
- *n*-gram language model:

$$\mathrm{P}(X_t \mid X_1, \ldots, X_{t-1}) \approx \mathrm{P}(X_t \mid \overbrace{X_{t-n+1}, \ldots, X_{t-1}}^{n-1 \text{ elements}})$$

[Prediction and Entropy of Printed English, Shanon 1950]

Pre-Computed N-Grams

Google Books Ngram Viewer

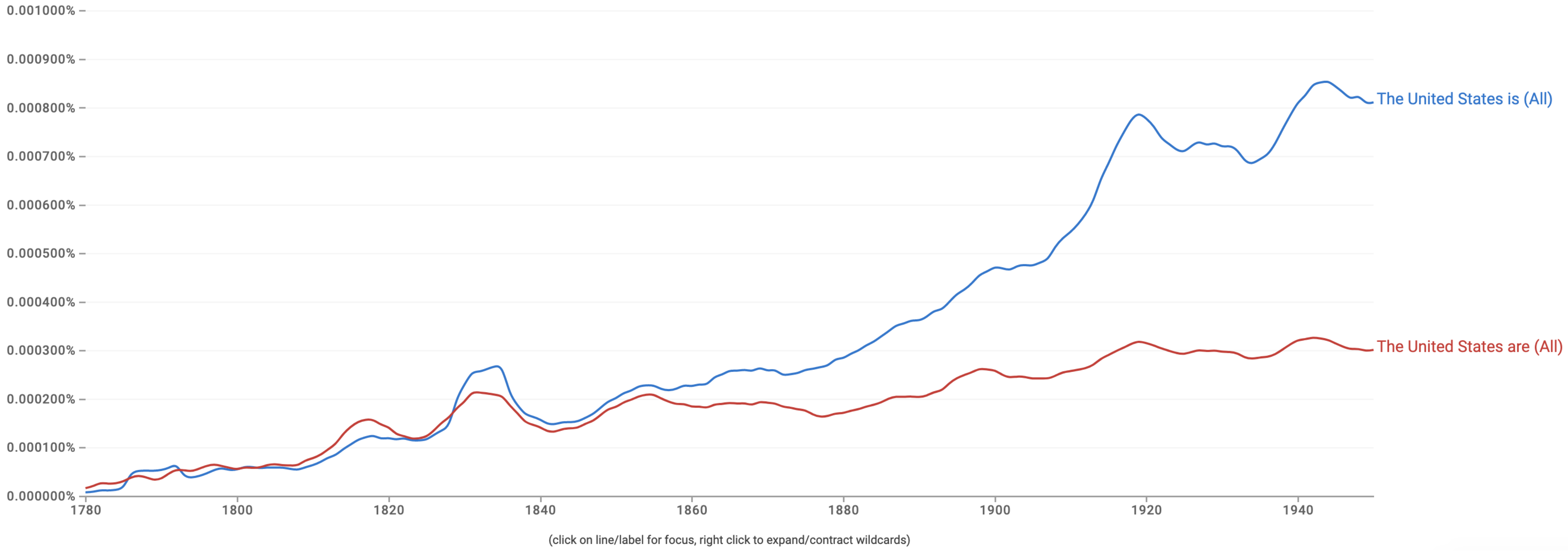


Google n-gram viewer https://books.google.com/ngrams/

Data: http://storage.googleapis.com/books/ngrams/books/datasetsv2.html

Pre-Computed N-Grams

Google Books Ngram Viewer



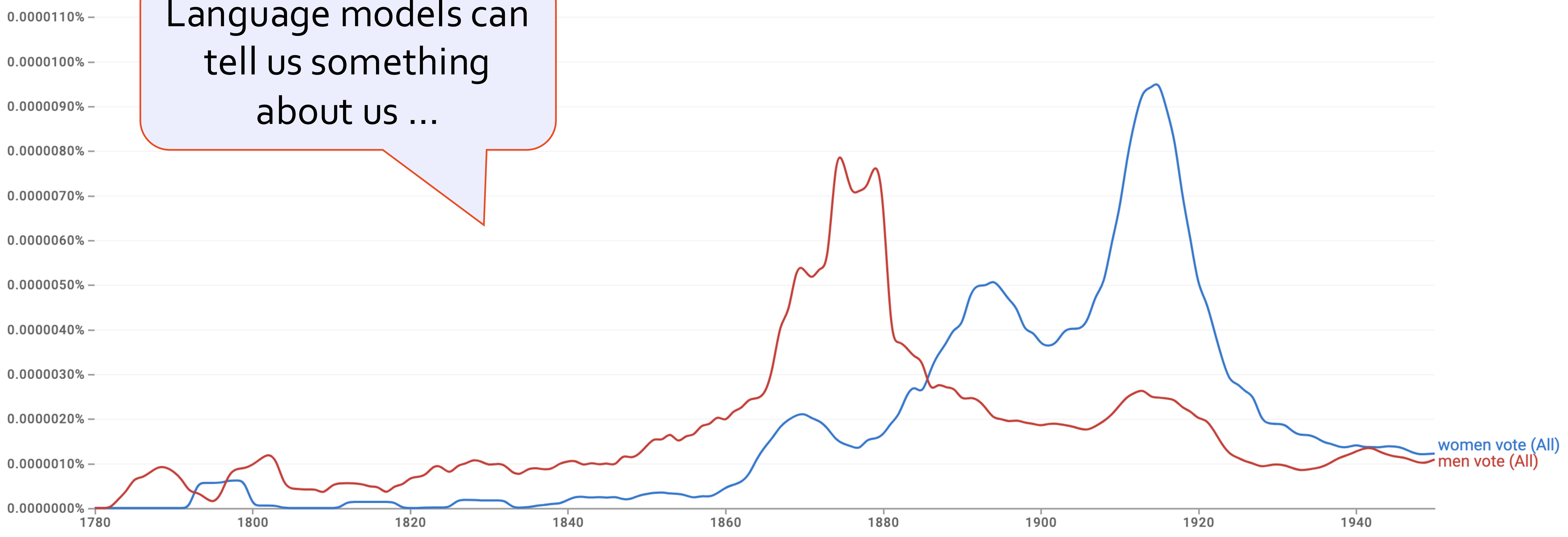
Google n-gram viewer https://books.google.com/ngrams/

Data: http://storage.googleapis.com/books/ngrams/books/datasetsv2.html

Pre-Computed N-Grams

Google Books Ngram Viewer

Language models can tell us something about us …



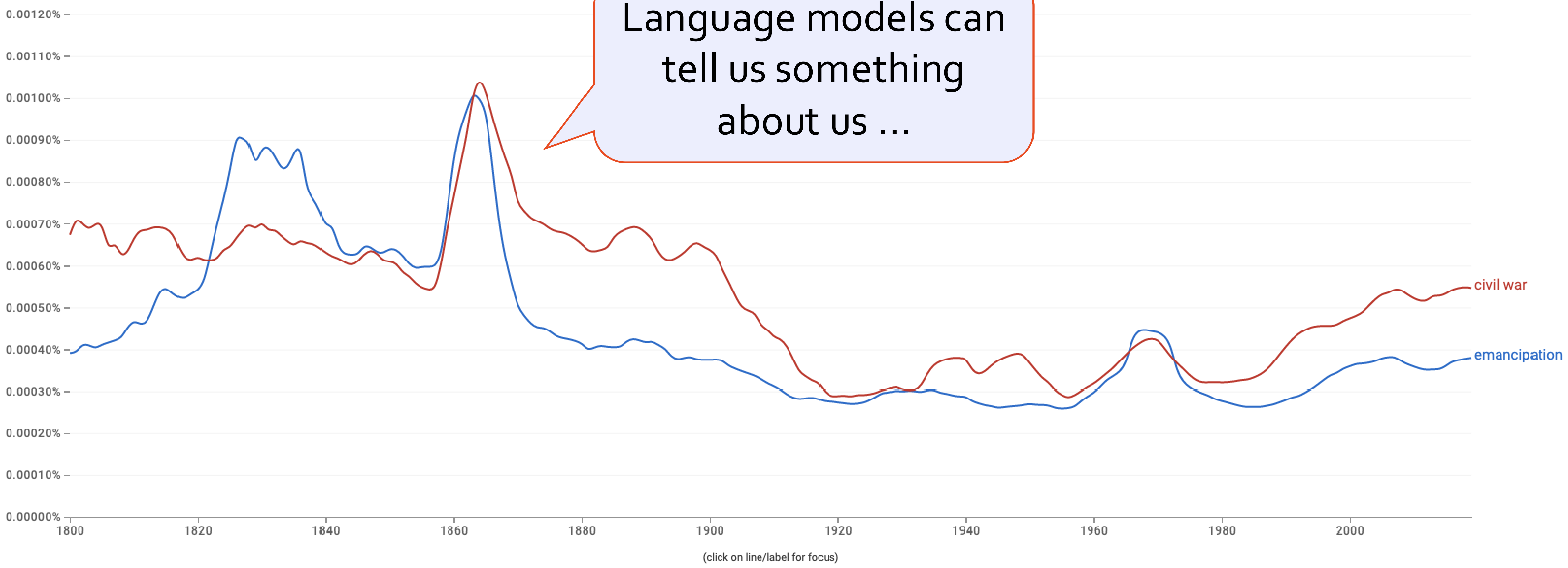
Google n-gram viewer https://books.google.com/ngrams/

Data: http://storage.googleapis.com/books/ngrams/books/datasetsv2.html

Pre-Computed N-Grams

Google Books Ngram Viewer

Language models can tell us something about us ...



Google n-gram viewer https://books.google.com/ngrams/

Data: http://storage.googleapis.com/books/ngrams/books/datasetsv2.html

Generation from N-Gram Models

- You can build a simple **tri**gram Language Model over a 1.7 million words corpus in a few seconds on your laptop*

```
today the ____
```

get probability distribution

```
company    0.153
bank       0.153
price      0.077
italian    0.039
emirate    0.039
...
```

Sparsity problem: not much granularity in the probability distribution

Otherwise, seems reasonable!

* Try for yourself: https://nlpforhackers.io/language-models/

[adopted from Chris Manning]

Generation from N-Gram Models

- Now we can sample from this mode:

`today the ____`

get probability distribution

company	0.153
bank	0.153
price	0.077
italian	0.039
emirate	0.039
...	

Sparsity problem: not much granularity in the probability distribution

Otherwise, seems reasonable!

* Try for yourself: https://nlpforhackers.io/language-models/

[adopted from Chris Manning]

Generation from N-Gram Models

- Now we can sample from this mode:

Condition on this

$\texttt{today } \underbrace{\texttt{the price}}_{} \texttt{ \_\_\_\_}$

get probability distribution

```
of    0.308
for   0.050
it    0.046
to    0.046
is    0.031
...
```

Sparsity problem: not much granularity in the probability distribution

Otherwise, seems reasonable!

* Try for yourself: https://nlpforhackers.io/language-models/

[adopted from Chris Manning]

Generation from N-Gram Models

- Now we can sample from this mode:

Condition on this

$\texttt{today the } \overbrace{\texttt{price of}}^{\text{Condition on this}} \texttt{ \_}$

get probability distribution

```
the   0.072
18    0.043
oil   0.043
its   0.036
gold  0.018
...
```

Sparsity problem: not much granularity in the probability distribution

Otherwise, seems reasonable!

* Try for yourself: https://nlpforhackers.io/language-models/

[adopted from Chris Manning]

N-Gram Models in Practice

- Now we can sample from this mode:

```
today the price of gold per ton , while production of shoe
lasts and shoe industry , the bank intervened just after it
considered and rejected an imf demand to rebuild depleted
european stocks , sept 30 end primary 76 cts a share .
```

Surprisingly grammatical!

But quite incoherent! To improve coherence, one may consider increasing larger than 3-grams, but that would worsen the sparsity problem!

* Try for yourself: https://nlpforhackers.io/language-models/ [adopted from Chris Manning]

Why is language modeling a good pretext task?

✓ Captures aspects of language useful for downstream tasks, including long-term dependencies, syntactic structure, and sentiment

✓ Lots of available data (especially in high-resource languages, e.g. English)

✓ Already a key component of many downstream tasks (e.g. machine translation)

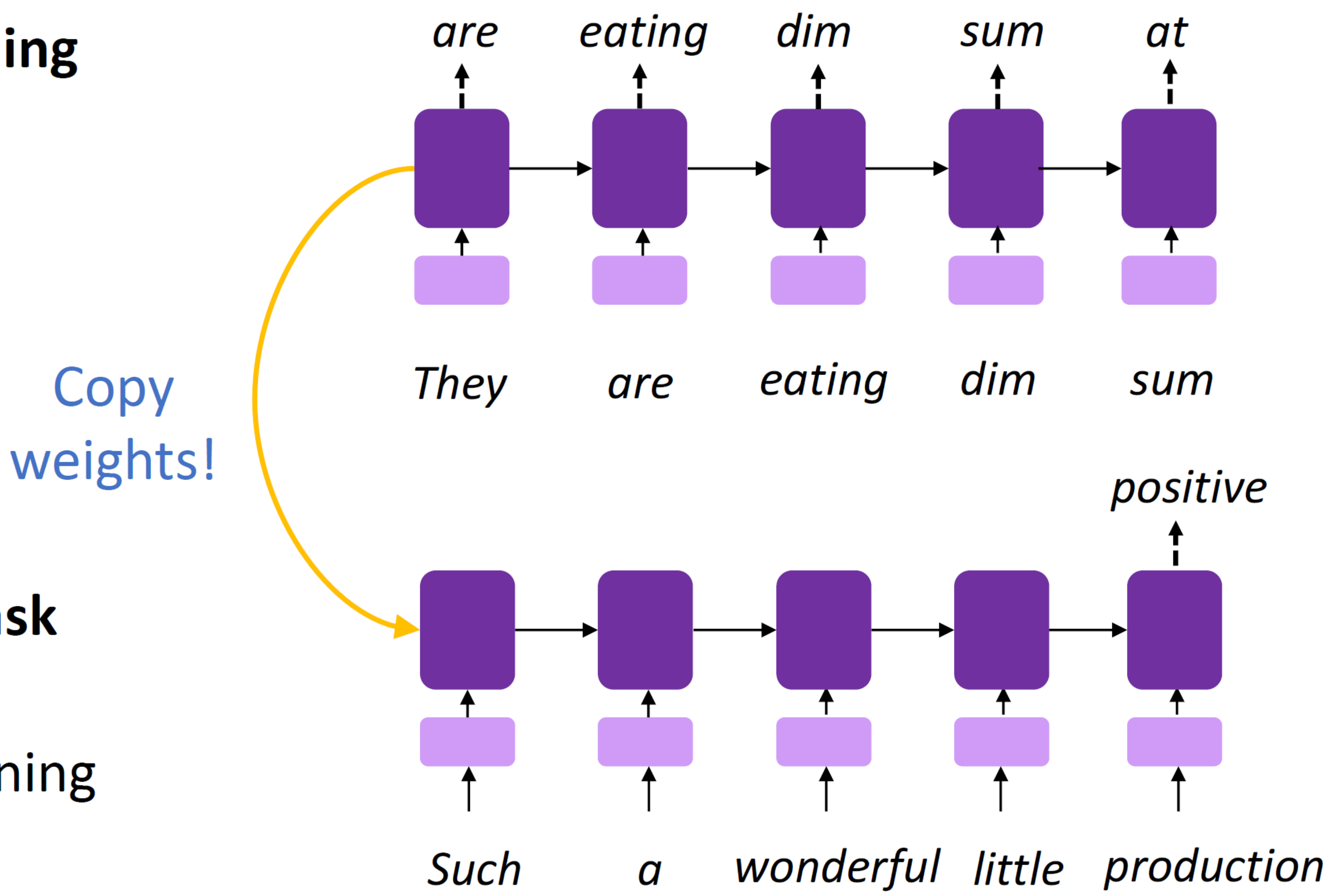
Using language modeling for pretraining

1. Pretrain on language modeling (pretext task)

- Self-supervised learning
- Large, unlabeled datasets

2. Finetune on downstream task (e.g. sentiment analysis)

- Supervised learning for finetuning
- Small, hand-labeled datasets



Case Study: Generative Pretrained Transformer (GPT)

- Introduced by Radford et al. in 2018 as a "universal" pretrained language representation
  - Pretrained with language modeling

- Uses the Transformer model [Vaswani et al., 2017]
  - Better **handles long-term dependencies** than alternatives (i.e. recurrent neural networks like LSTMs) and **more efficient on current hardware**

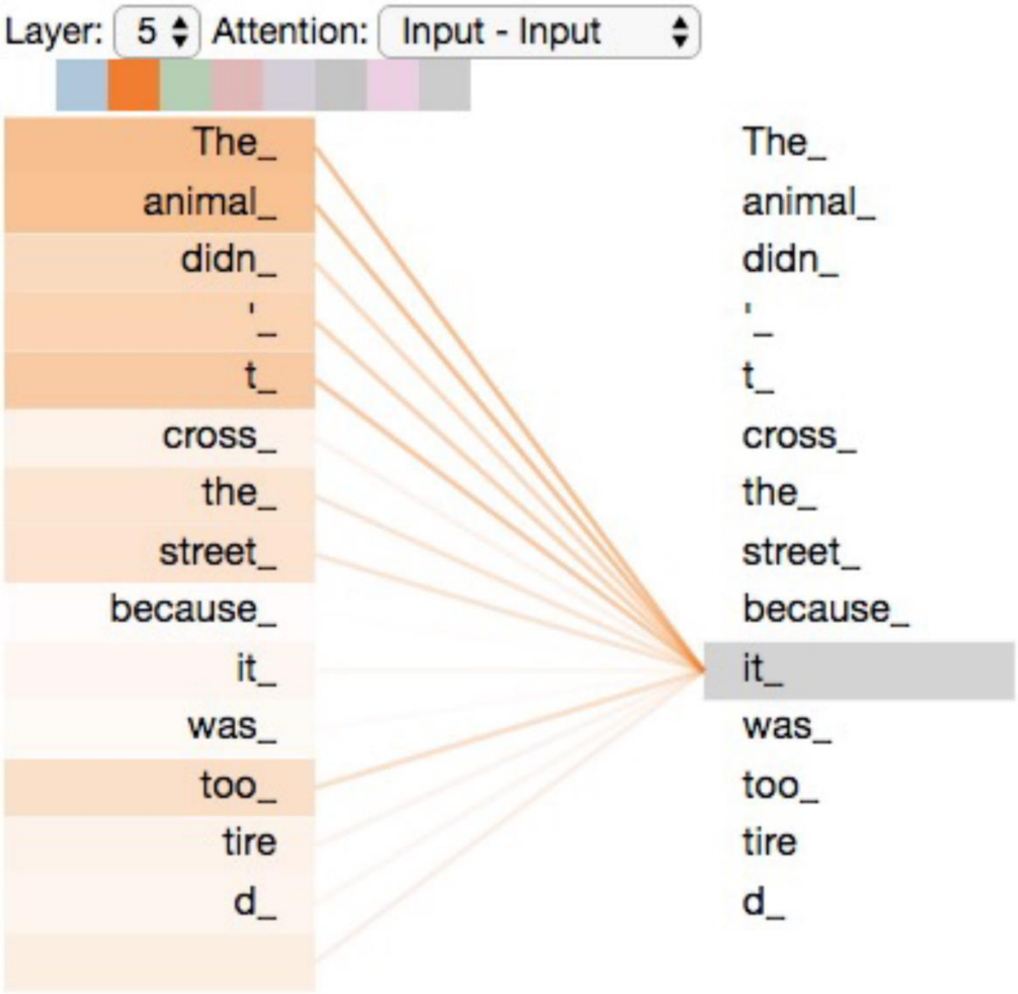
- Has since had follow-on work with GPT-2 and GPT-3 resulting in even larger pretrained models

https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/languageunsupervised/language_understanding_paper.pdf

Quick Aside: Basics of Transformers

- Model architecture that has recently replaced recurrent neural networks (e.g. LSTMS) as the building block in many NLP pipelines

- Uses **self-attention** to pay attention to relevant words in the sequence (“Attention is all you need”)
  - Can attend to words that are far away

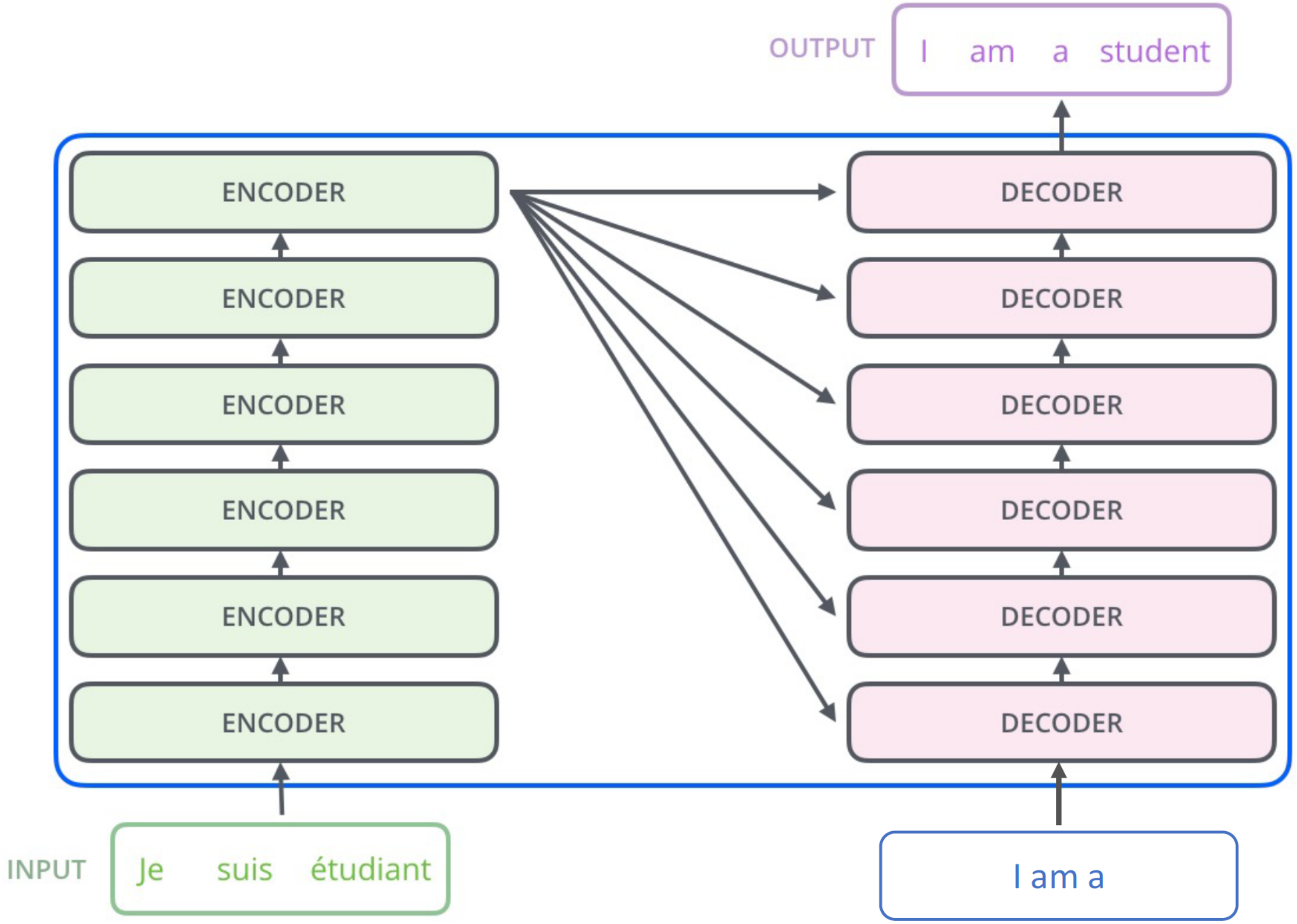


[Alammar et al., Illustrated Transformer]

https://jalammar.github.io/illustrated-transformer/

Quick Aside: Basics of Transformers

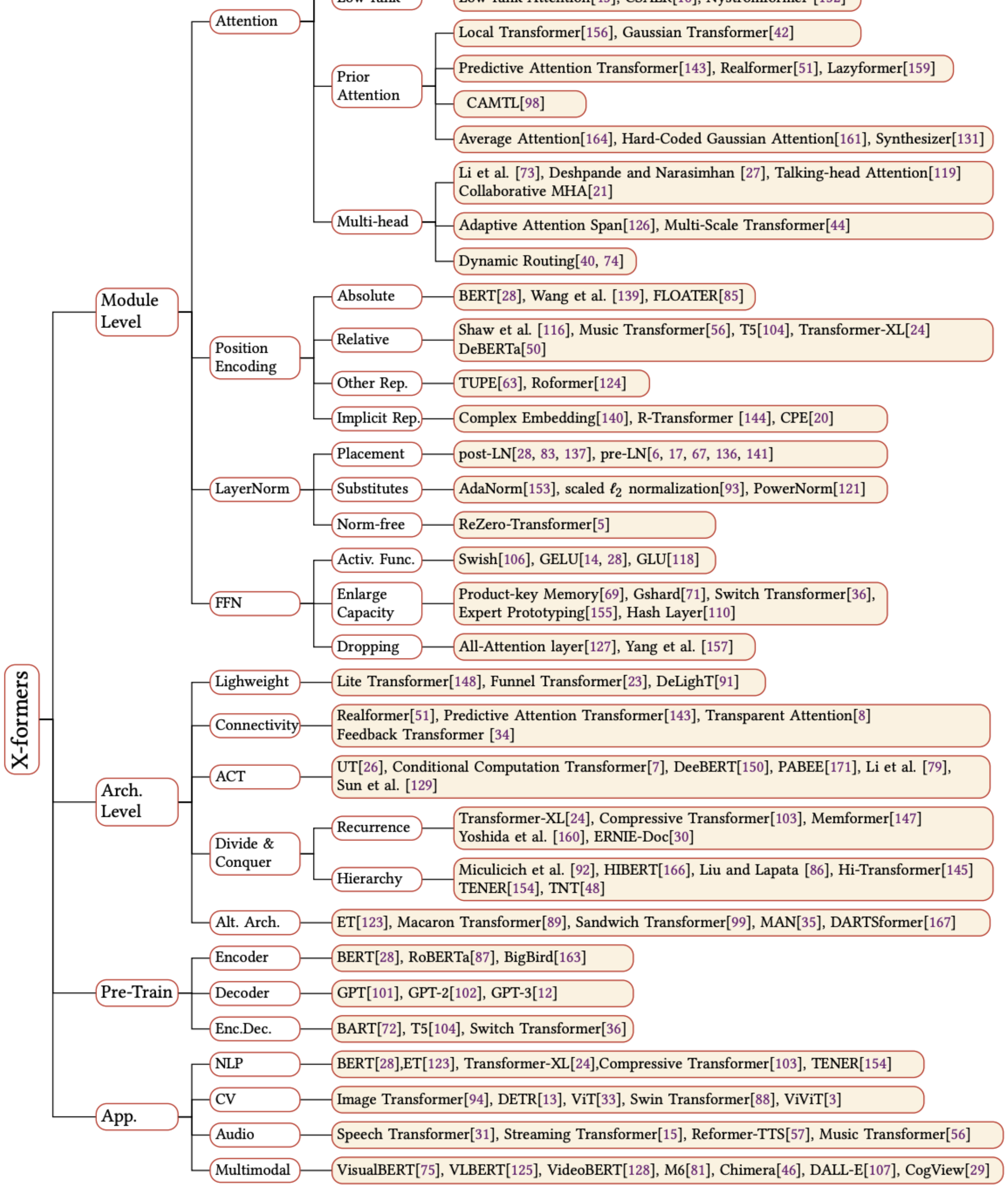
- Composed of two modules:
  - **Encoder** to learn representations of the input
  - **Decoder** to generate output conditioned on the encoder output and the previous decoder output (auto-regressive)
- Each block contains a self-attention and feedforward layer



[Alammar et al., Illustrated Transformer]

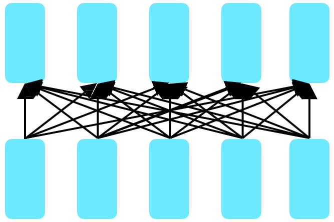
https://jalammar.github.io/illustrated-transformer/

After Transformer …


X-formers
Module Level
Attention
Prior Attention
Local Transformer[156], Gaussian Transformer[42]
Predictive Attention Transformer[143], Realformer[51], Lazyformer[159]
CAMTL[98]
Average Attention[164], Hard-Coded Gaussian Attention[161], Synthesizer[131]
Multi-head
Li et al. [73], Deshpande and Narasimhan [27], Talking-head Attention[119] Collaborative MHA[21]
Adaptive Attention Span[126], Multi-Scale Transformer[44]
Dynamic Routing[40, 74]
Position Encoding
Absolute
BERT[28], Wang et al. [139], FLOATER[85]
Relative
Shaw et al. [116], Music Transformer[56], T5[104], Transformer-XL[24] DeBERTa[50]
Other Rep.
TUPE[63], Roformer[124]
Implicit Rep.
Complex Embedding[140], R-Transformer [144], CPE[20]
LayerNorm
Placement
post-LN[28, 83, 137], pre-LN[6, 17, 67, 136, 141]
Substitutes
AdaNorm[153], scaled ℓ_2 normalization[93], PowerNorm[121]
Norm-free
ReZero-Transformer[5]
FFN
Activ. Func.
Swish[106], GELU[14, 28], GLU[118]
Enlarge Capacity
Product-key Memory[69], Gshard[71], Switch Transformer[36], Expert Prototyping[155], Hash Layer[110]
Dropping
All-Attention layer[127], Yang et al. [157]
Arch. Level
Lighweight
Lite Transformer[148], Funnel Transformer[23], DeLighT[91]
Connectivity
Realformer[51], Predictive Attention Transformer[143], Transparent Attention[8] Feedback Transformer [34]
ACT
UT[26], Conditional Computation Transformer[7], DeeBERT[150], PABEE[171], Li et al. [79], Sun et al. [129]
Divide & Conquer
Recurrence
Transformer-XL[24], Compressive Transformer[103], Memformer[147] Yoshida et al. [160], ERNIE-Doc[30]
Hierarchy
Miculicich et al. [92], HIBERT[166], Liu and Lapata [86], Hi-Transformer[145] TENER[154], TNT[48]
Alt. Arch.
ET[123], Macaron Transformer[89], Sandwich Transformer[99], MAN[35], DARTSformer[167]
Pre-Train
Encoder
BERT[28], RoBERTa[87], BigBird[163]
Decoder
GPT[101], GPT-2[102], GPT-3[12]
Enc.Dec.
BART[72], T5[104], Switch Transformer[36]
App.
NLP
BERT[28],ET[123], Transformer-XL[24],Compressive Transformer[103], TENER[154]
CV
Image Transformer[94], DETR[13], ViT[33], Swin Transformer[88], ViViT[3]
Audio
Speech Transformer[31], Streaming Transformer[15], Reformer-TTS[57], Music Transformer[56]
Multimodal
VisualBERT[75], VLBERT[125], VideoBERT[128], M6[81], Chimera[46], DALL-E[107], CogView[29]

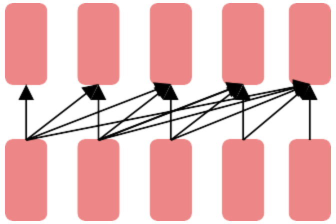
Impact of Transformers

- A building block for a variety of LMs



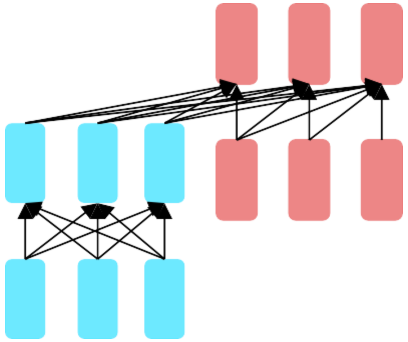
Encoders

- ❖ Examples: BERT, RoBERTa, SciBERT.
- ❖ Captures bidirectional context.
- ❖ Wait, how do we pretrain them?



Decoders

- ❖ Examples: GPT-2, GPT-3, LaMDA
- ❖ Other name: causal or auto-regressive language model
- ❖ Nice to generate from; can't condition on future words



Encoder-Decoders

- ❖ Examples: Transformer, T5, Meena
- ❖ What's the best way to pretrain them?

Case Study: Generative Pretrained Transformer (GPT)

- Pretrain the **Transformer decoder model** on the language modeling task:

Word in a sequence

$$L_{LM}(U) = \sum_{i=1}^{n} \log P(u_i \mid u_{i-k}, \dots, u_{i-1};\ \theta)$$

Text corpus

Context window

$$h_{i-k}, \dots, h_{i-1} = \text{decoder}(u_{i-k}, \dots, u_{i-1})$$

$$P(u_i \mid u_{i-k}, \dots, u_{i-1}) = \text{softmax}(h_{i-1} W_e^T)$$

Previous word hidden representation

Linear layer

https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/languageunsupervised/language_understanding_paper.pdf

Case Study: Generative Pretrained Transformer (GPT)

- Finetune the pretrained Transformer model with a randomly initialized linear layer for **supervised downstream tasks**:

Labeled dataset

Input sequence x, label y

$$L_{downstream}(C) = \sum_{(x,y)} \log P(y \mid x_1, \ldots, x_m)$$

$$h_1, \ldots, h_m = \text{decoder}(u_1, \ldots, u_m)$$

$$P(y \mid x_1, \ldots, x_m) = \text{softmax}(h_m W_y)$$

Last word's hidden representation

New linear layer, replaces W_e from pretraining

- Linear layer makes up most of the **new** parameters needed for downstream tasks, rest are initialized from pretraining!

https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/languageunsupervised/language_understanding_paper.pdf

Case Study: Generative Pretrained Transformer (GPT)

- Pretrained on the BooksCorpus (7000 unique books)
- Achieved state-of-the-art on **downstream** question answering tasks (as well as natural language inference, semantic similarity, and text classification tasks)

select the correct end to the story

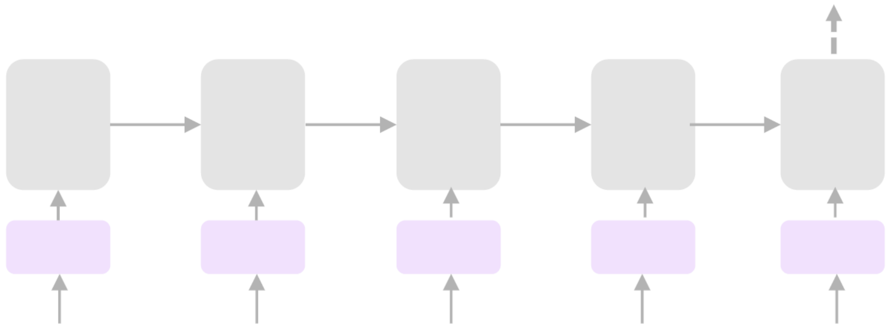
middle and high school exam reading comprehension questions

Method	Story Cloze	RACE-m	RACE-h	RACE
val-LS-skip [55]	76.5	-	-	-
Hidden Coherence Model [7]	77.6	-	-	-
Dynamic Fusion Net [67] (9x)	-	55.6	49.4	51.2
BiAttention MRU [59] (9x)	-	60.2	50.3	53.3
Finetuned Transformer LM (ours)	**86.5**	**62.9**	**57.4**	**59.0**

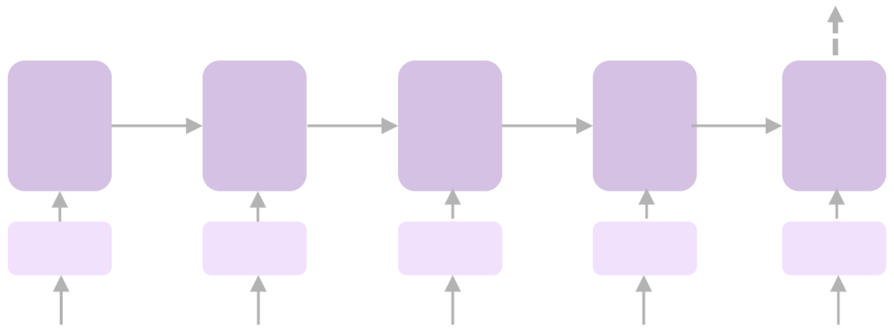
https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/languageunsupervised/language_understanding_paper.pdf

Examples of Self-Supervision in NLP

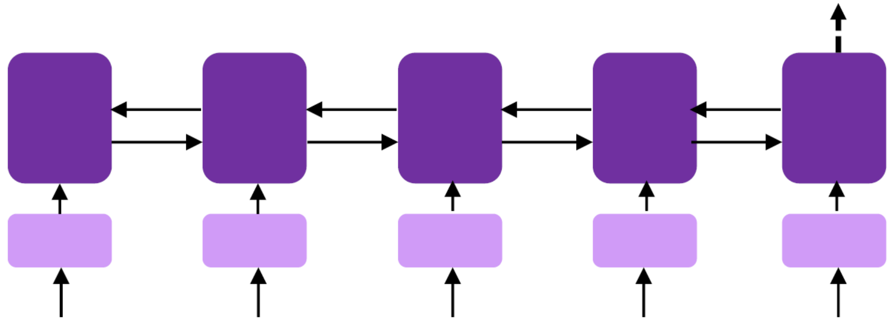
- **Word embeddings**
  - Pretrained word representations
  - Initializes *1st layer* of downstream models



- **Language models**
  - *Unidirectional*, pretrained language representations
  - Initializes *full* downstream model



- **Masked language models**
  - *Bidirectional*, pretrained language representations
  - Initializes *full* downstream model



Using context from the future

- Consider predicting the next word for the following example:

He is going to the ________.

movies *park*
store *theater*
library *treehouse*
school *pool*

- What if you have more (bidirectional) context?

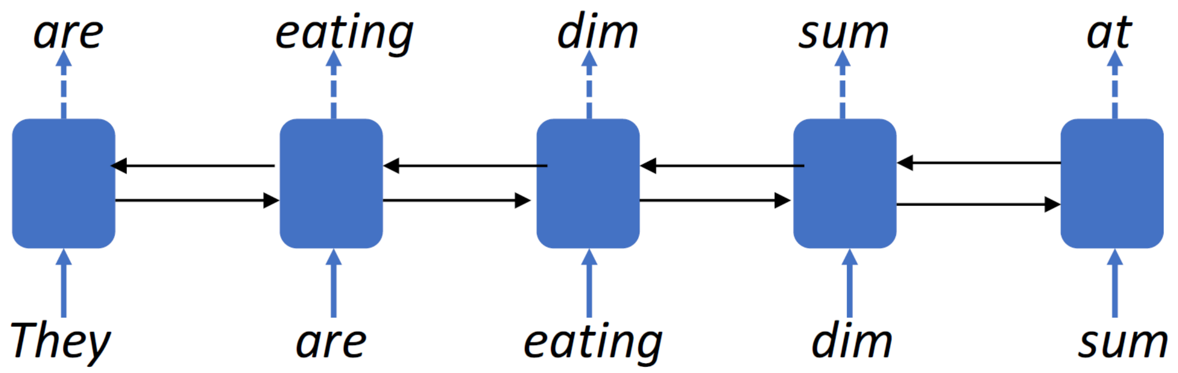
He is going to the ________ to buy some milk.

store
market
Safeway

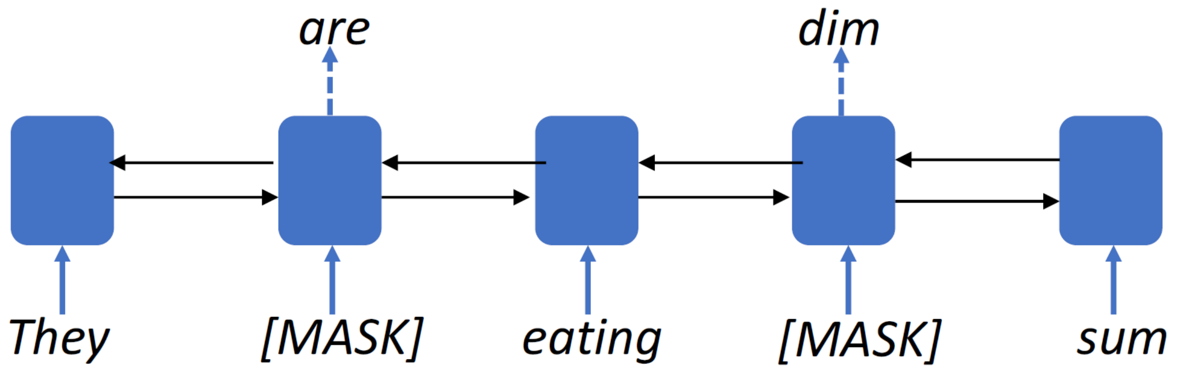
- Information **from the future** can be helpful for language understanding!

Masked language models (MLMs)

- With bidirectional context, if we aren't careful, model can "cheat" and see next word

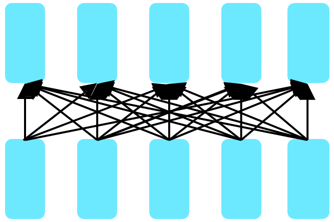


- What if we mask out some words and ask the model to predict them?



This is called ***masked language modeling***.

BERT



Encoders

BERT

Bidirectional Encoder Representations from Transformers

BERT

Bidirectional **E**ncoder **R**epresentations from **T**ransformers

Like Bidirectional LSTMs (ELMo), let's look in **both** directions

BERT

Bidirectional **E**ncoder **R**epresentations from **T**ransformers

Let's only use Transformer Encoders, no Decoders

BERT

Bidirectional Encoder Representations from Transformers

It's a language model that builds rich representations
via self-supervised learning (pre-training)

BERT (2018)

- Transformer based network to learn representations of language

- Improvements
  - Bi-directional LSTM -> Self-attention
  - Massive data
  - Masked-LM objective

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

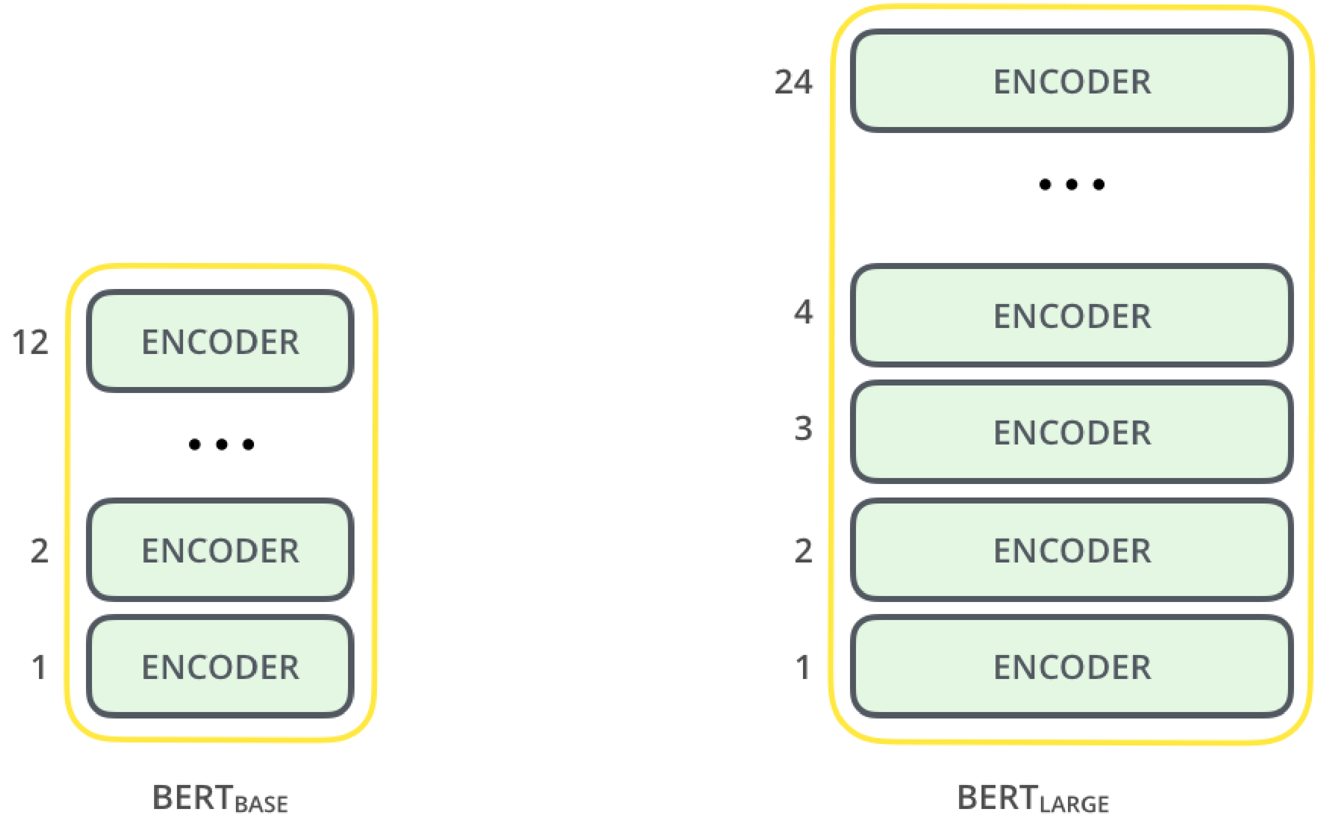
BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute im-

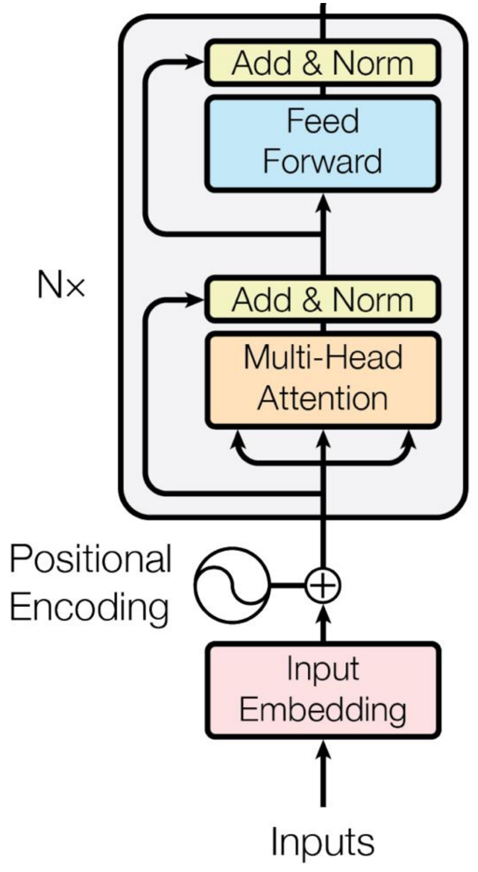
There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pre-trained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations.

We argue that current techniques restrict the power of the pre-trained representations, especially for the fine-tuning approaches. The major limitation is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training. For example, in OpenAI GPT, the authors use a left-to-

BERT: Architecture

- Stacks of Transformer encoders”

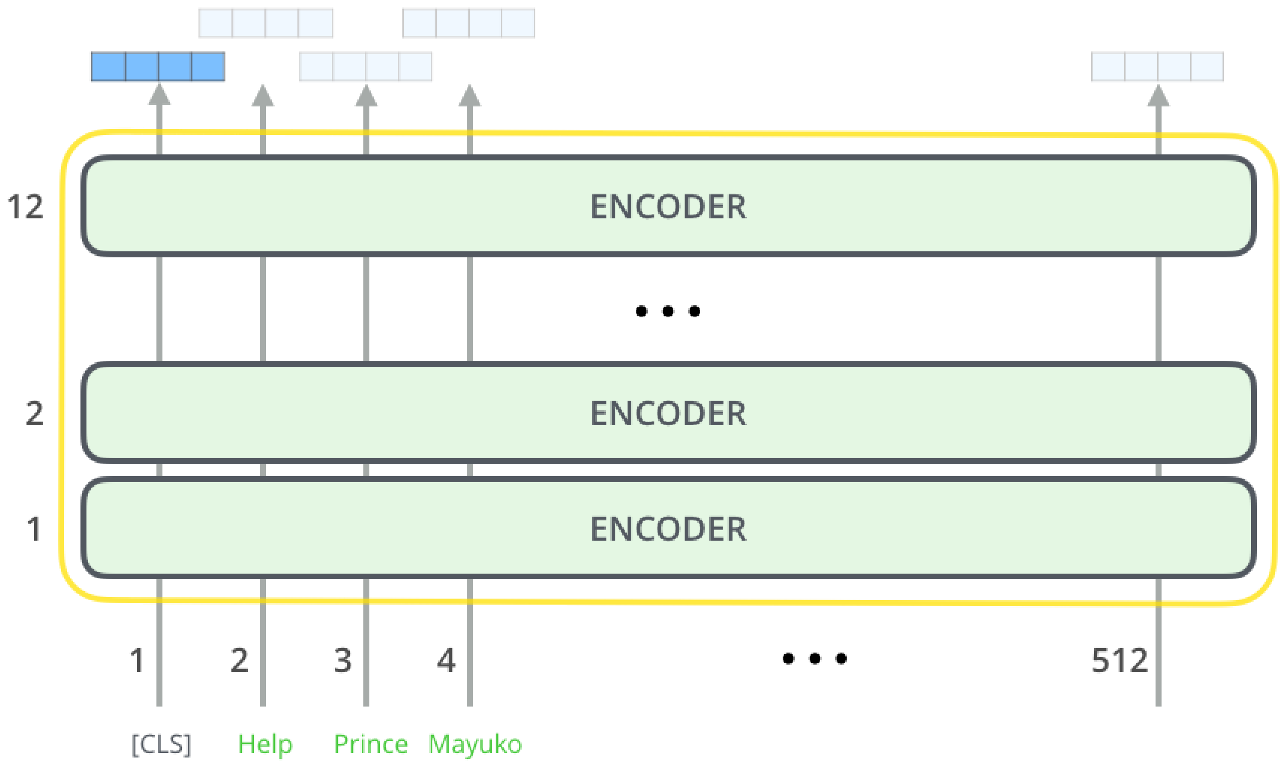


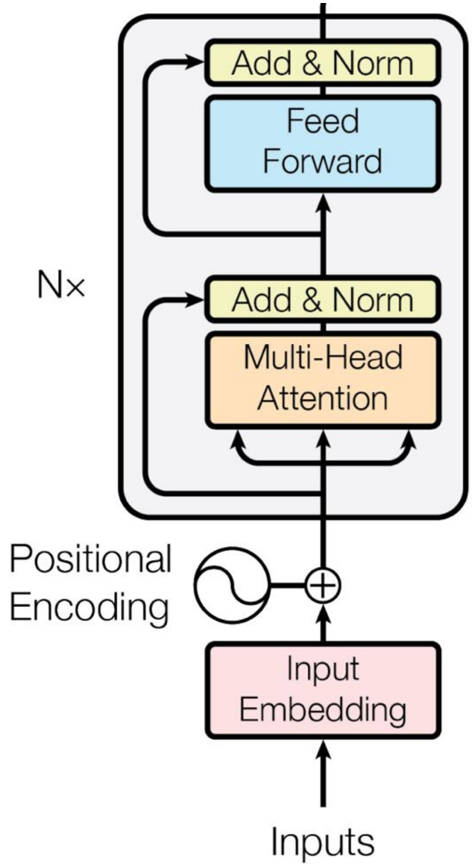


[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]

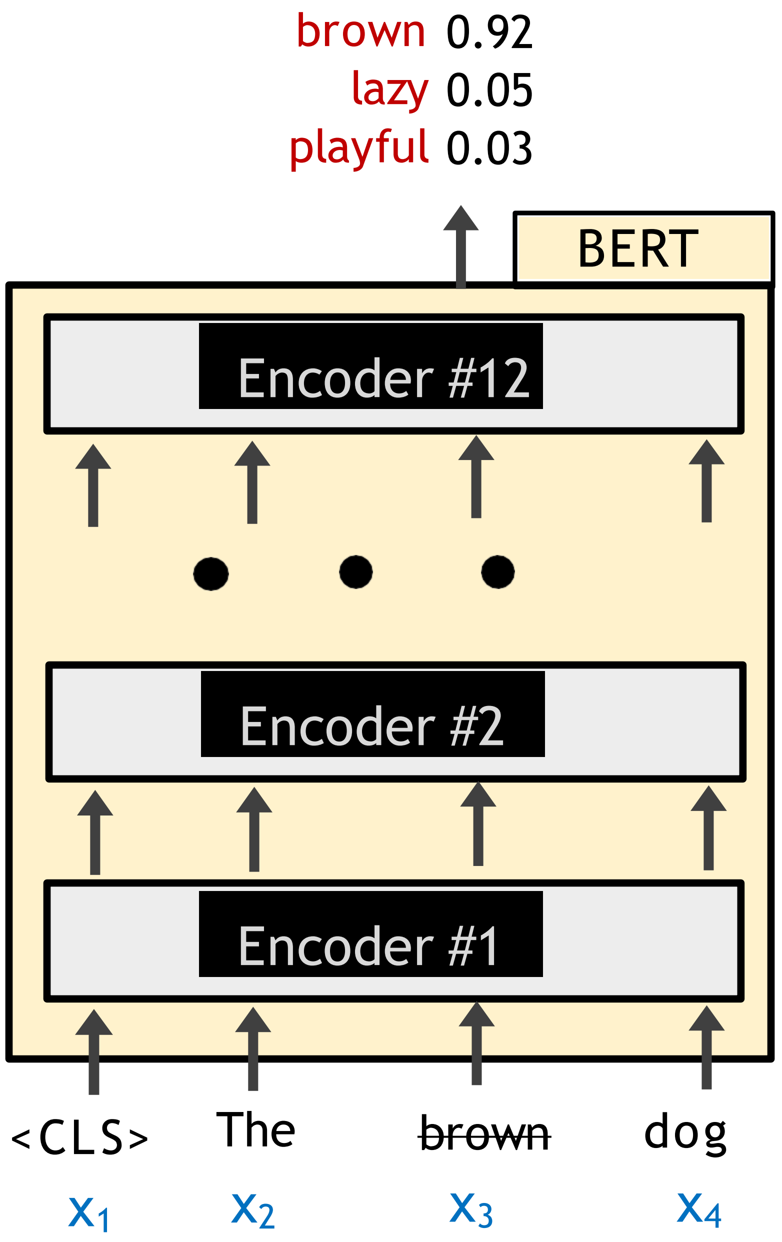
BERT: Architecture

- Model output dimension: 512





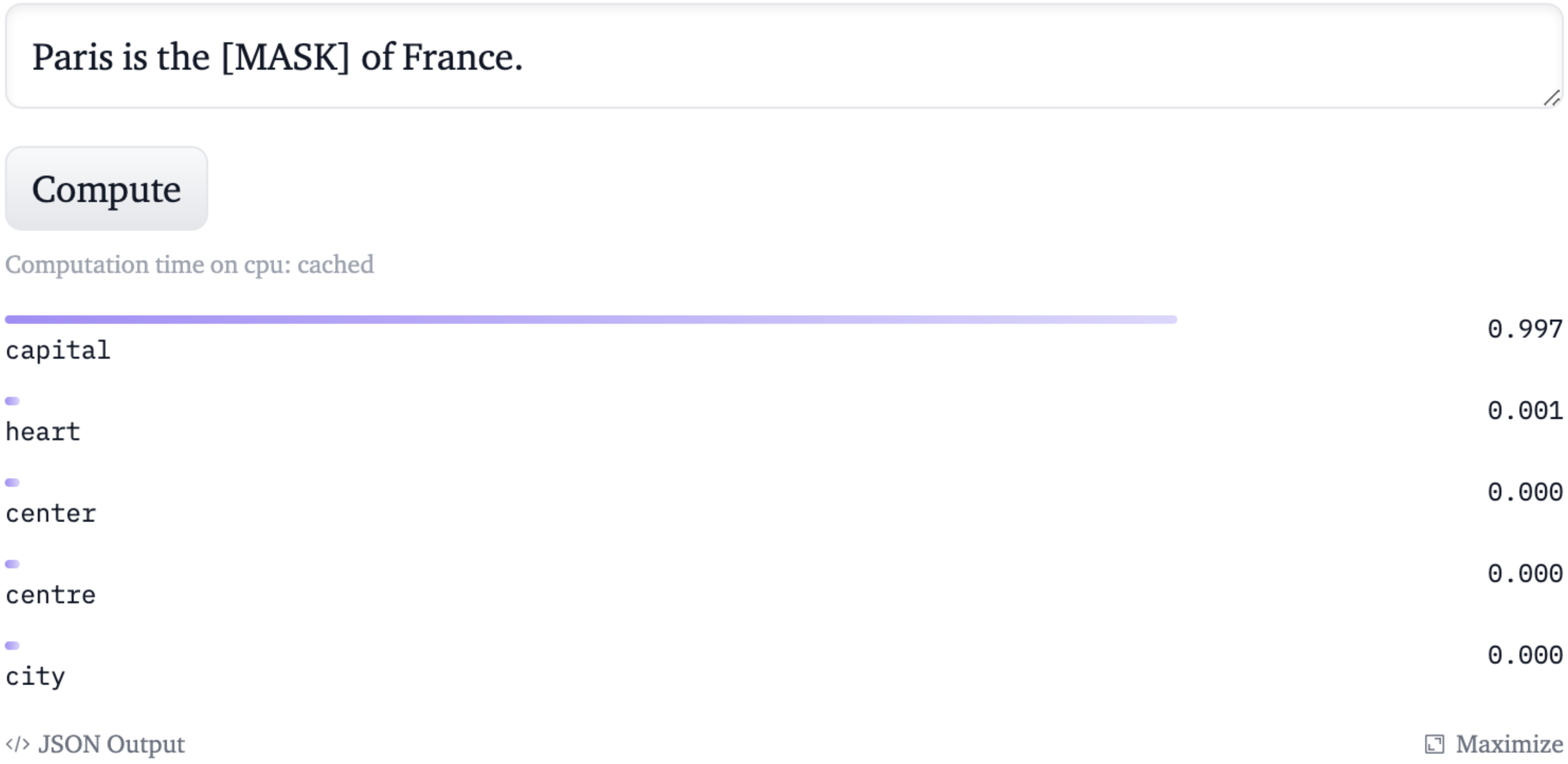
[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]



BERT is trained to uncover masked tokens.

Probing BERT Masked LM

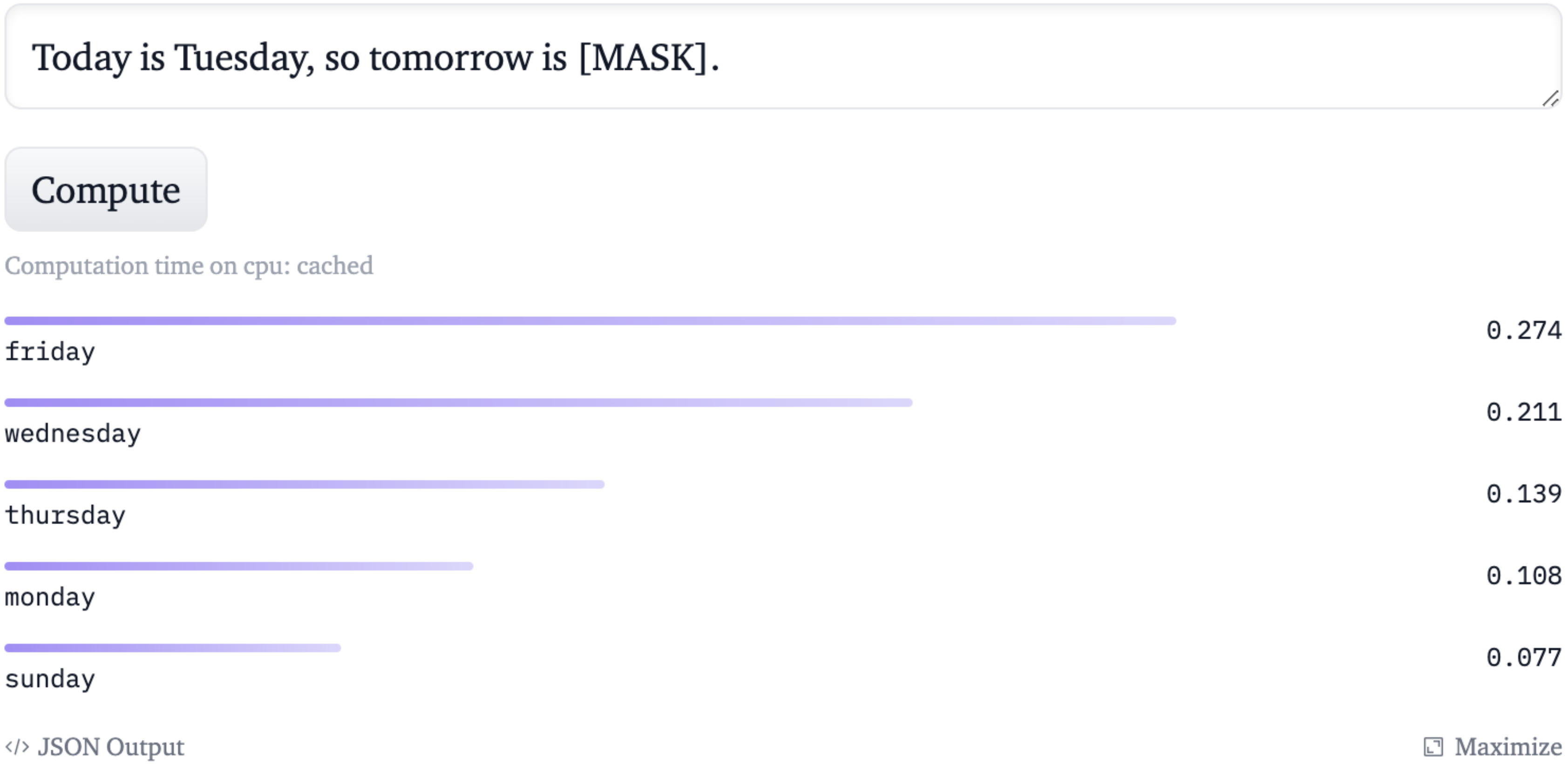
- Making words forces BERT to use context in both directions to predict the masked word.



https://huggingface.co/bert-base-uncased

Probing BERT Masked LM

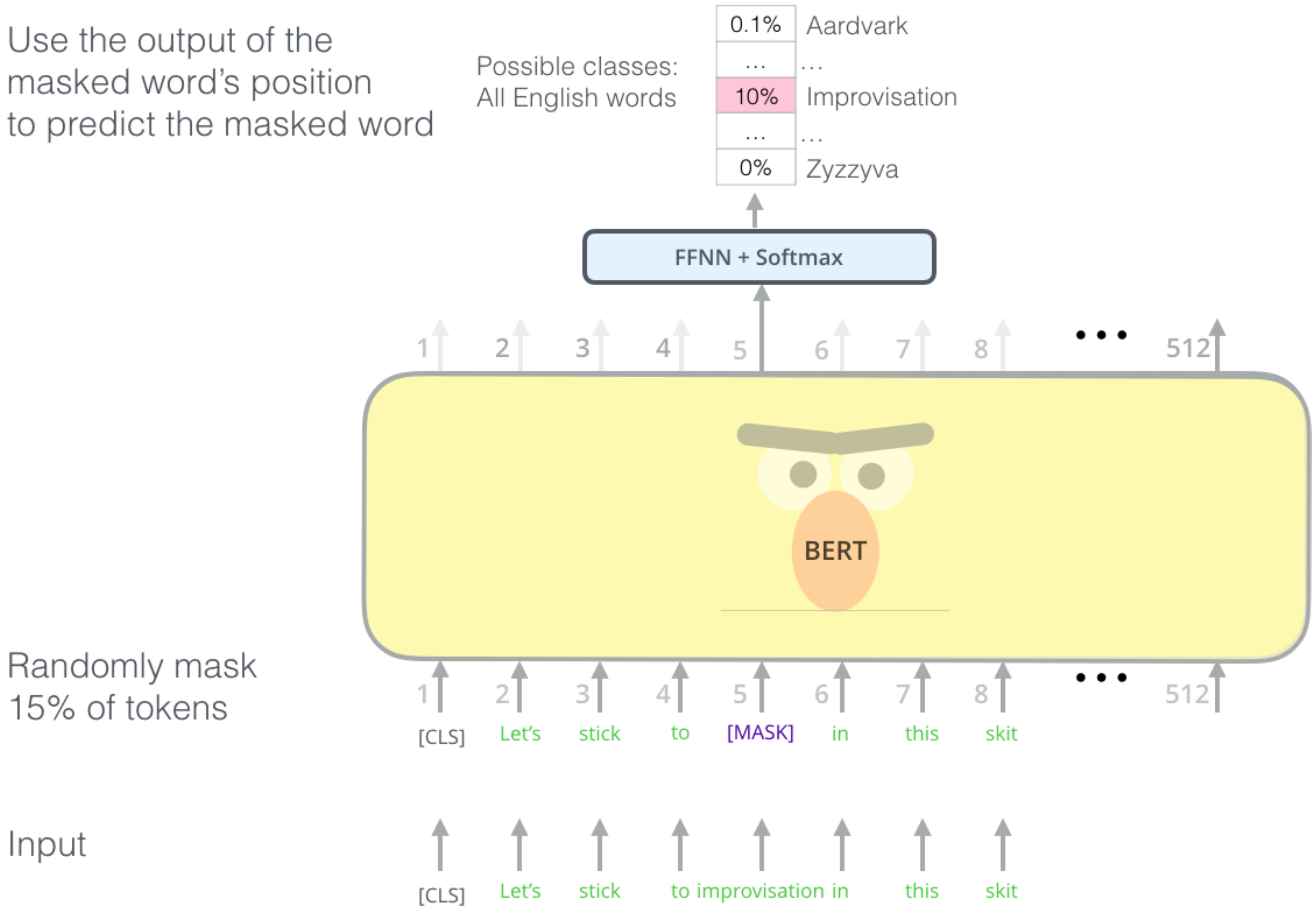
- Making words forces BERT to use context in both directions to predict the masked word.



https://huggingface.co/bert-base-uncased

BERT: Pre-training Objective (1): Masked Tokens

- Randomly mask 15% of the tokens and train the model to predict them.



[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]

BERT: Pre-training Objective (1): Masked Tokens

store

Galon

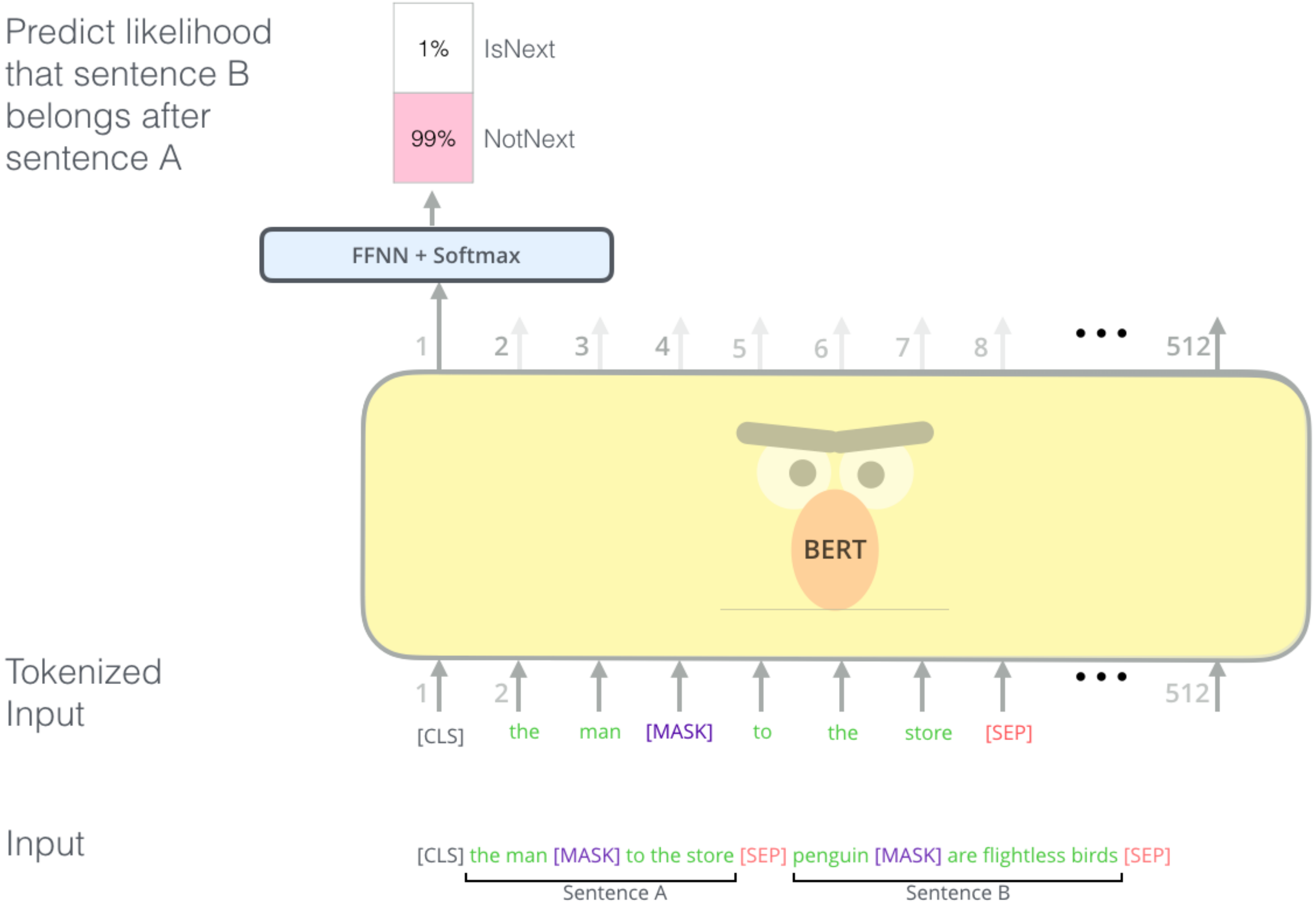
`the man went to the [MASK] to buy a [MASK] of milk`

- Too little masking: Too expensive to train
- Too much masking: Underdefined (not enough context)

[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]

BERT: Pre-training Objective (2): Sentence Ordering

- Predict sentence ordering
- 50% correct ordering, and 50% random incorrect ones



[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]

BERT: Pre-training Objective (2): Sentence Ordering

- Learn relationships between sentences, predict whether Sentence B is actual sentence that proceeds Sentence A, or a random sentence

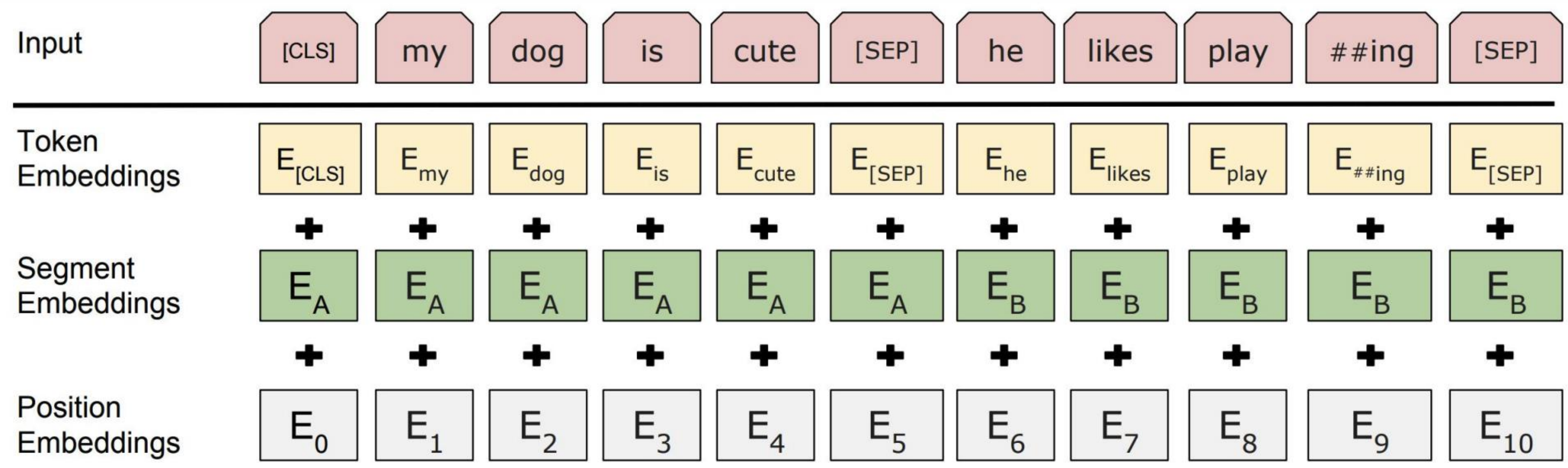
Sentence A = The man went to the store.
Sentence B = He bought a gallon of milk.
Label = IsNextSentence

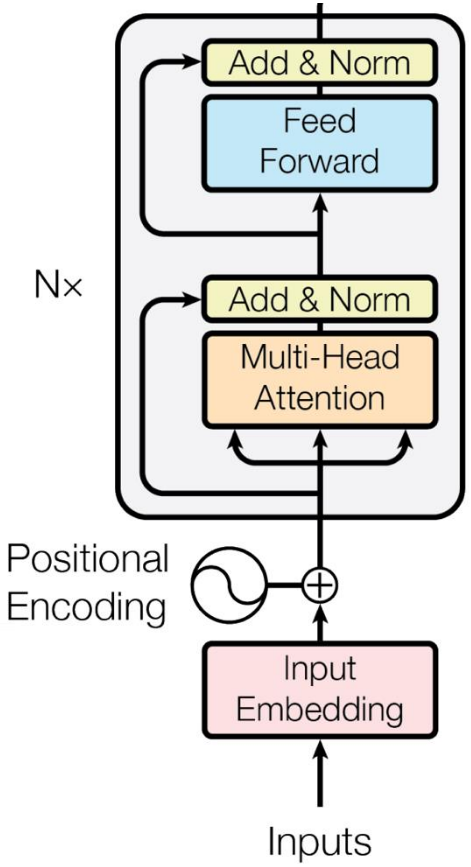
Sentence A = The man went to the store.
Sentence B = Penguins are flightless.
Label = NotNextSentence

[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]

BERT: Input Representation

- Use 30,000 WordPiece vocabulary on input.
- Each token is sum of three embeddings
  - Addition to transformer encoder: sentence embedding





[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]

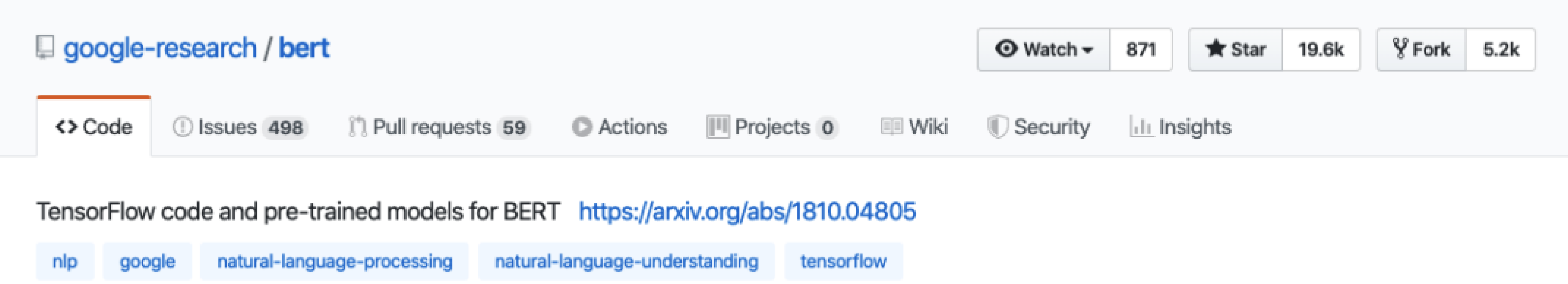
Training

- Trains model on unlabeled data over different pre-training tasks (self-supervised learning)
- **Data:** Wikipedia (2.5B words) + BookCorpus (800M words)
- **Training Time:** 1M steps (~40 epochs)
- **Optimizer:** AdamW, 1e-4 learning rate, linear decay
- **BERT-Base:** 12-layer, 768-hidden, 12-head
- **BERT-Large:** 24-layer, 1024-hidden, 16-head
- Trained on 4x4 or 8x8 TPUs for 4 days

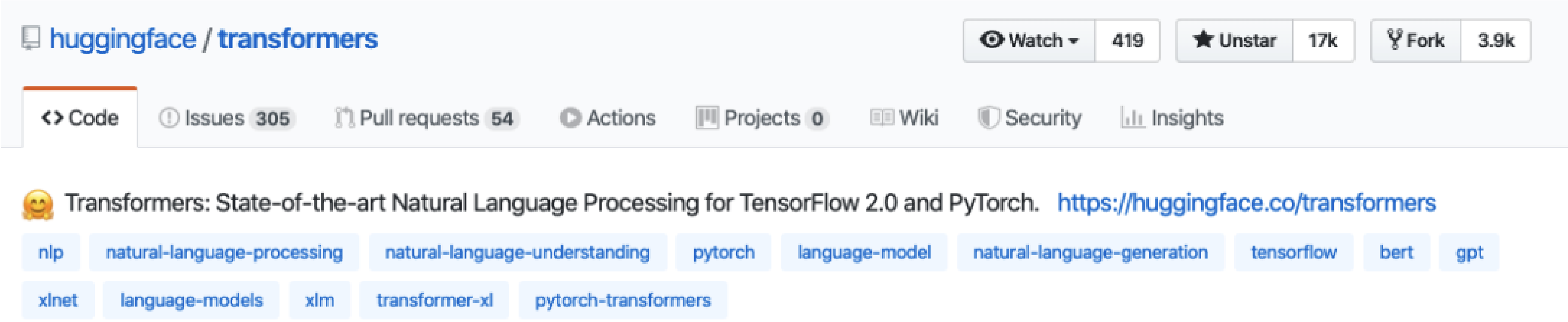
[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]

BERT in Practice

TensorFlow: https://github.com/google-research/bert

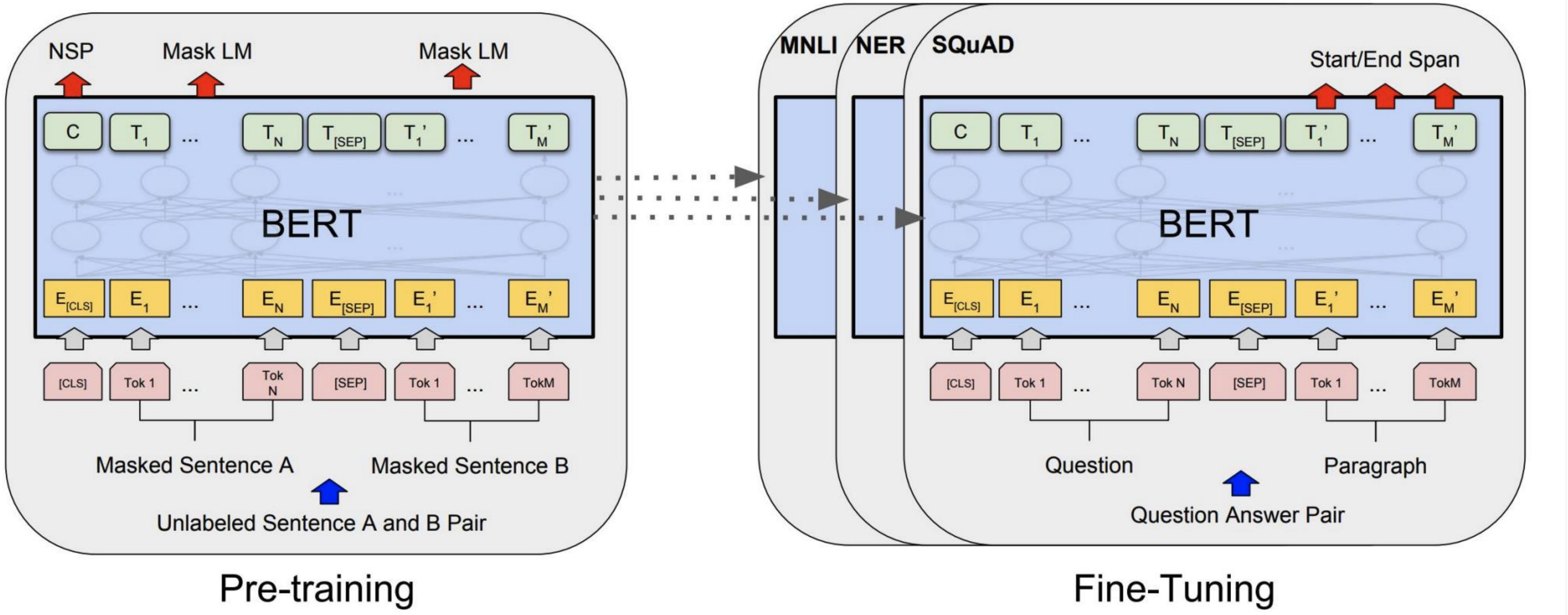


PyTorch: https://github.com/huggingface/transformers



Fine-tuning BERT "Pretrain once, finetune many times."

- **Idea:** Make pre-trained model **usable** in **downstream tasks**
- Initialized with pre-trained model parameters
- Fine-tune model parameters using labeled data from downstream tasks



[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]

An Example Result: SWAG

```
A girl is going across a set of monkey bars.  She
(i)  jumps up across the monkey bars.
(ii)  struggles onto the bars to grab her head.
(iii)  gets to the end and stands on a wooden plank.
(iv)  jumps up and does a back flip.
```

Leaderboard

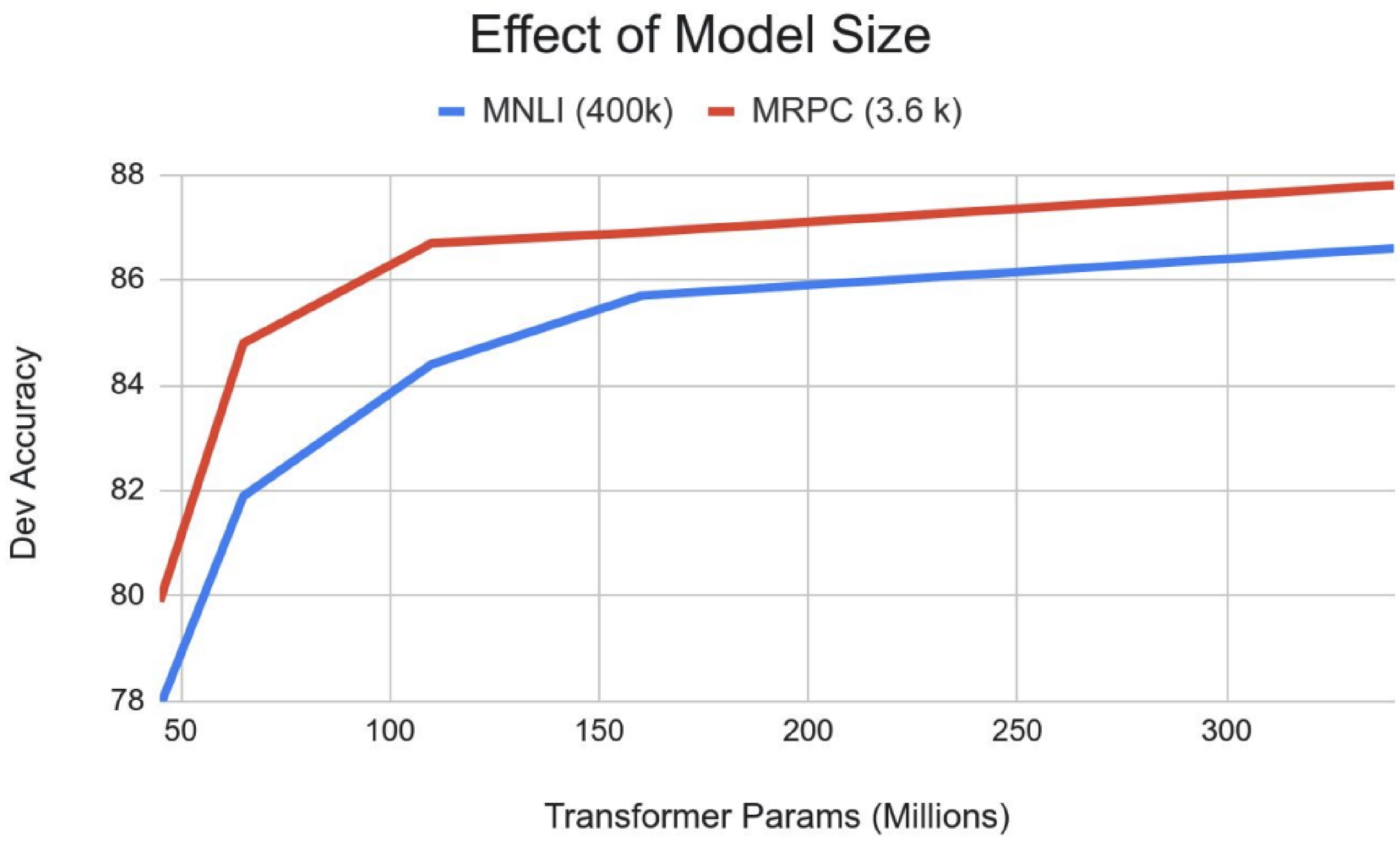
Human Performance (88.00%)
Running Best
Submissions

Rank	Model	Test Score
1	**BERT (Bidirectional Encoder Representations from Transfo...** *Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova* 10/11/2018	**86.28%**
2	**OpenAI Transformer Language Model** *Original work by Alec Radford, Karthik Narasimhan, Tim Salimans, ...* 10/11/2018	**77.97%**
3	**ESIM with ELMo** *Zellers, Rowan and Bisk, Yonatan and Schwartz, Roy and Choi, Yejin* 08/30/2018	**59.06%**
4	**ESIM with Glove** *Zellers, Rowan and Bisk, Yonatan and Schwartz, Roy and Choi, Yejin* 08/29/2018	**52.45%**

- Run each Premise + Ending through BERT.
- Produce logit for each pair on token 0 (`[CLS]`)

[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]

Effect of Model



- Big models help a lot
- Going from 110M -> 340M params helps even on datasets with 3,600 labeled examples
- Improvements have not *asymptoted*

[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]

Why did no one think of this before?

- Concretely, why wasn’t contextual pre-training popular before 2018 with ELMo?

- Good results on pre-training is >1,000x to 100,000 more expensive than supervised training.

What Happened After BERT?

- RoBERTa (Liu et al., 2019)
  - Drops the next sentence prediction loss!
  - Trained on 10x data (the original BERT was actually under-trained)
  - Much stronger performance than BERT (e.g., 94.6 vs 90.9 on SQuAD)
  - Still one of the most popular models to date

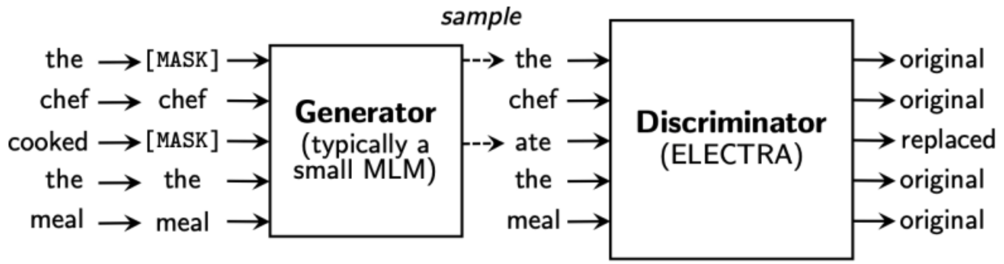
Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$BERT_{LARGE}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7

What Happened After BERT?

- RoBERTa (Liu et al., 2019)
  - Drops the next sentence prediction loss!
  - Trained on 10x data (the original BERT was actually under-trained)
  - Much stronger performance than BERT (e.g., 94.6 vs 90.9 on SQuAD)
  - Still one of the most popular models to date

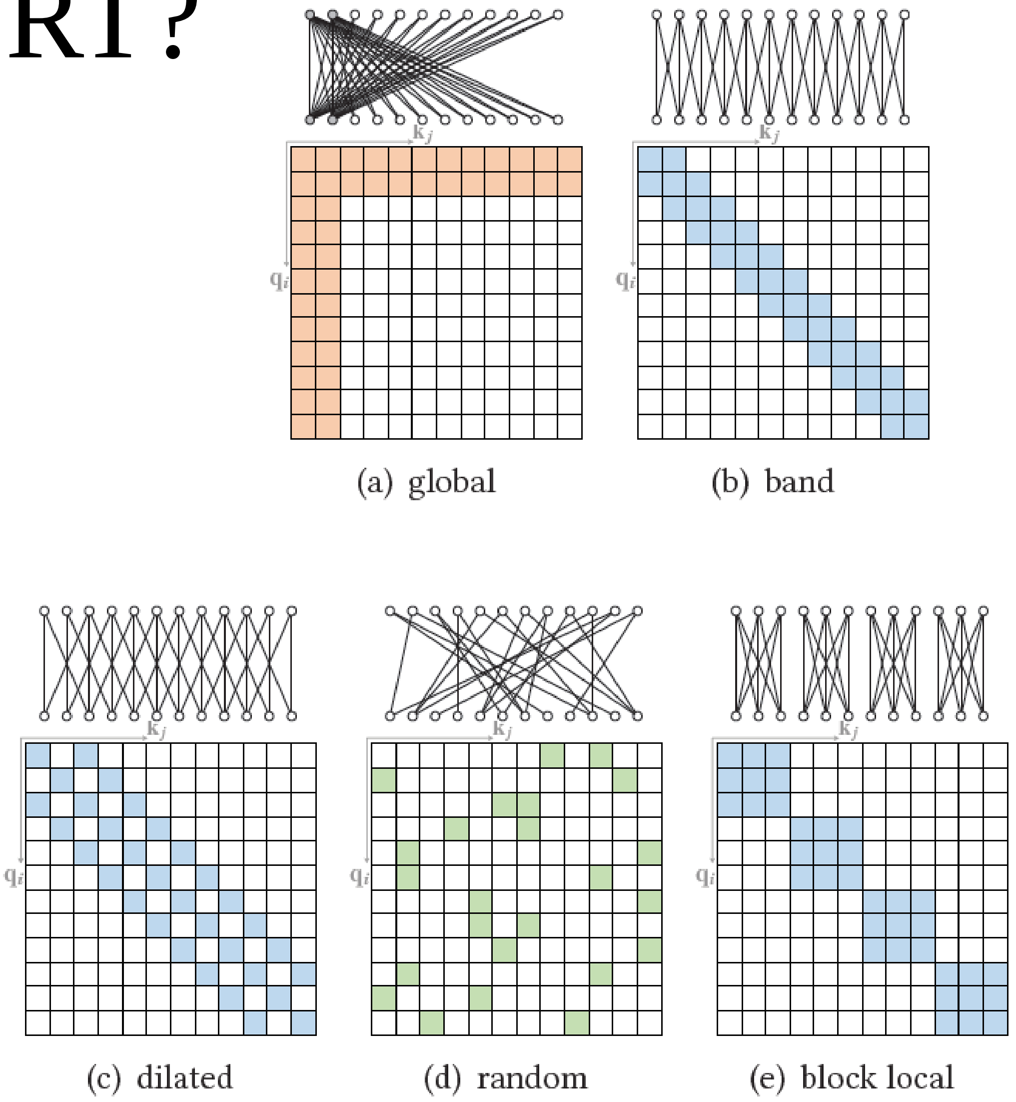
- ALBERT (Lan et al., 2020)
  - Increasing model sizes by sharing model parameters across layers
  - Less storage, much stronger performance but runs slower..

- ELECTRA (Clark et al., 2020)
  - Two models generator and discriminator
  - It provides a more efficient training method



What Happened After BERT?

- Models that handle long contexts (512 tokens)
  - Longformer, Big Bird, …
- Multilingual BERT
  - Trained single model on 104 languages from Wikipedia. Shared 110k WordPiece vocabulary
- BERT extended to different domains
  - SciBERT, BioBERT, FinBERT, ClinicalBERT, ..
- Making BERT smaller to use
  - DistillBERT, TinyBERT, …



Text generation using BERT

BERT has a Mouth, and It Must Speak: BERT as a Markov Random Field Language Model

Alex Wang
New York University
alexwang@nyu.edu

Kyunghyun Cho
New York University
Facebook AI Research
CIFAR Azrieli Global Scholar
kyunghyun.cho@nyu.edu

Mask-Predict: Parallel Decoding of Conditional Masked Language Models

Marjan Ghazvininejad* **Omer Levy*** **Yinhan Liu*** **Luke Zettlemoyer**
Facebook AI Research
Seattle, WA

Exposing the Implicit Energy Networks behind Masked Language Models via Metropolis--Hastings

Kartik Goyal, Chris Dyer, Taylor Berg-Kirkpatrick

src	Der Abzug der franzsischen Kampftruppen wurde am 20. November abgeschlossen .
$t = 0$	The departure of the French combat completed completed on 20 November .
$t = 1$	The departure of French combat troops was completed on 20 November .
$t = 2$	The withdrawal of French combat troops was completed on November 20th .

Leveraging Pre-trained Checkpoints for Sequence Generation Tasks

Sascha Rothe, Shashi Narayan, Aliaksei Severyn

Summary

1. What is self-supervised learning?

2. Examples of self-supervision in NLP
   - Word embeddings (e.g., word2vec)
   - Language models (e.g., GPT)
   - Masked language models (e.g., BERT)

3. Open challenges
   - Demoting bias
   - Capturing factual knowledge
   - Learning symbolic reasoning