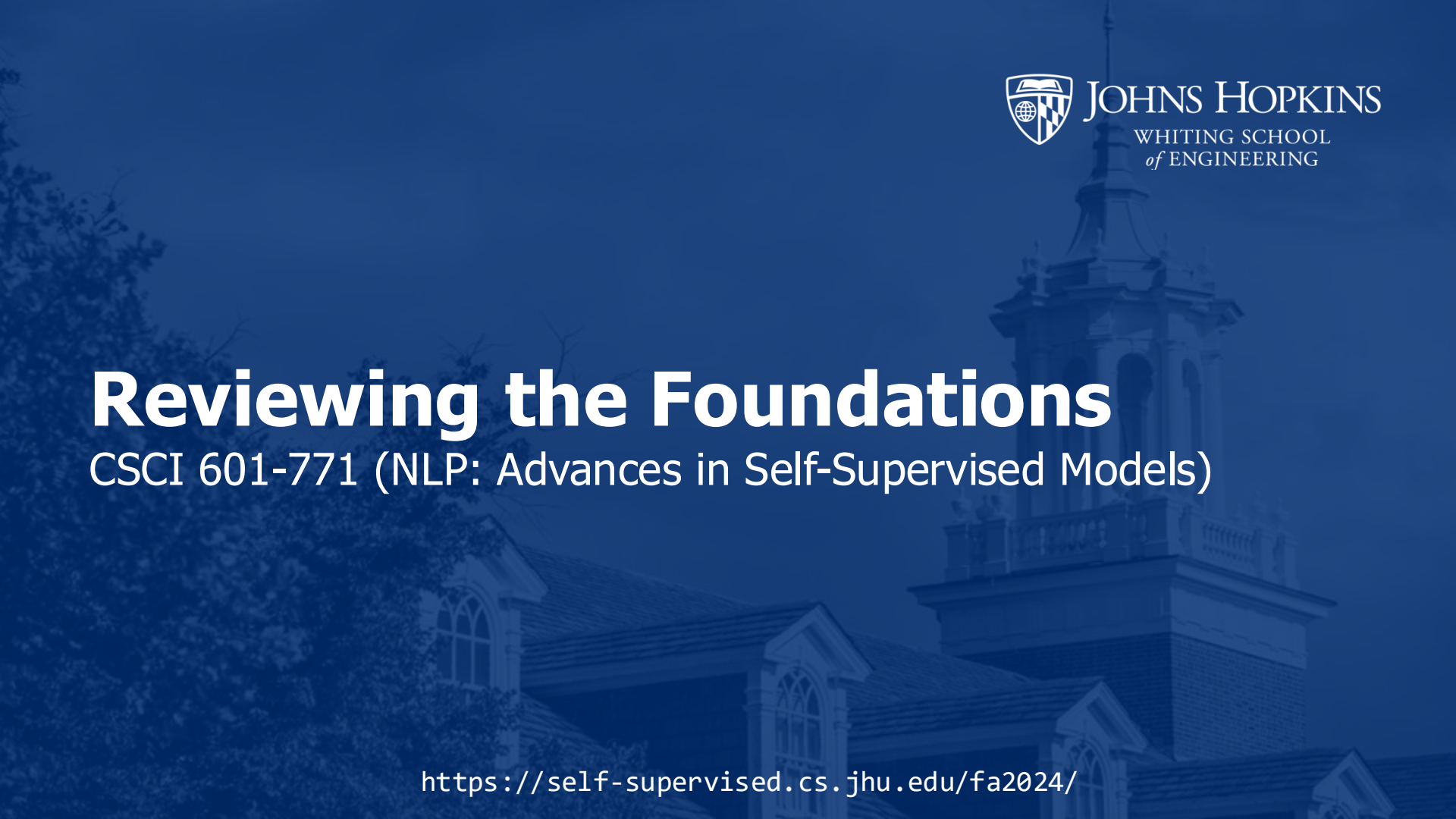


Reviewing the Foundations

CSCI 601-771 (NLP: Advances in Self-Supervised Models)

https://self-supervised.cs.jhu.edu/fa2024/

Language Modeling: Definitions and History

The

The cat

The cat sat

The cat sat on

The cat sat on __?__

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

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

next word

context or prefix

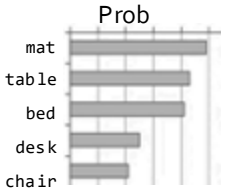
LMs as a Marginal Distribution

- Directly we train models on "marginals":

next word | context

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

"The cat sat on the [MASK]" → *Language Model* →



LMs as Implicit Joint Distribution over Language

- While language modeling involves learning the marginals, we are implicitly learning the full/joint distribution of language.
  - Remember the chain rule:

$$\mathbf{P}(X_1, \ldots, X_t) = \mathrm{P}(X_1) \prod_{i=1}^{t} \mathrm{P}(X_i \,|X_1, X_2 \ldots, X_i)$$

- 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"

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.

next word context

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

- 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!

next word context

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

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

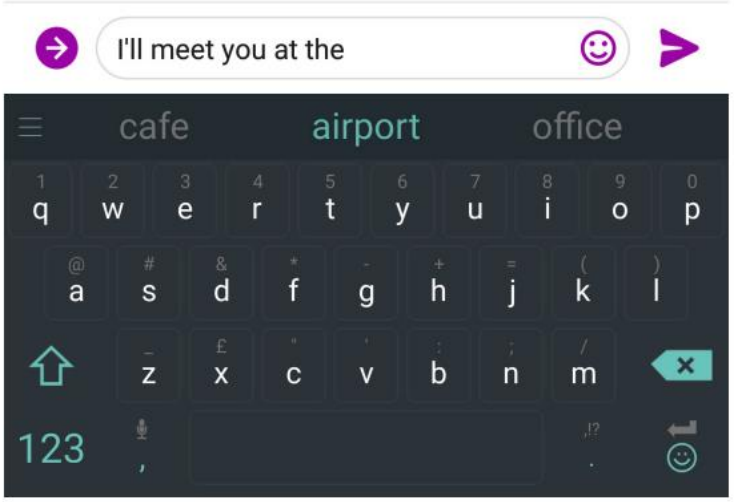
1. Sample from **P**(X|*"Johns Hopkins is "*) →"located"
2. Sample from **P**(X|*"Johns Hopkins is located"*) →"at"
3. Sample from **P**(X|*"Johns Hopkins is located at"*) →"the"
4. Sample from **P**(X|*"Johns Hopkins is located at the"*) →"state"
5. Sample from **P**(X|*"Johns Hopkins is located at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

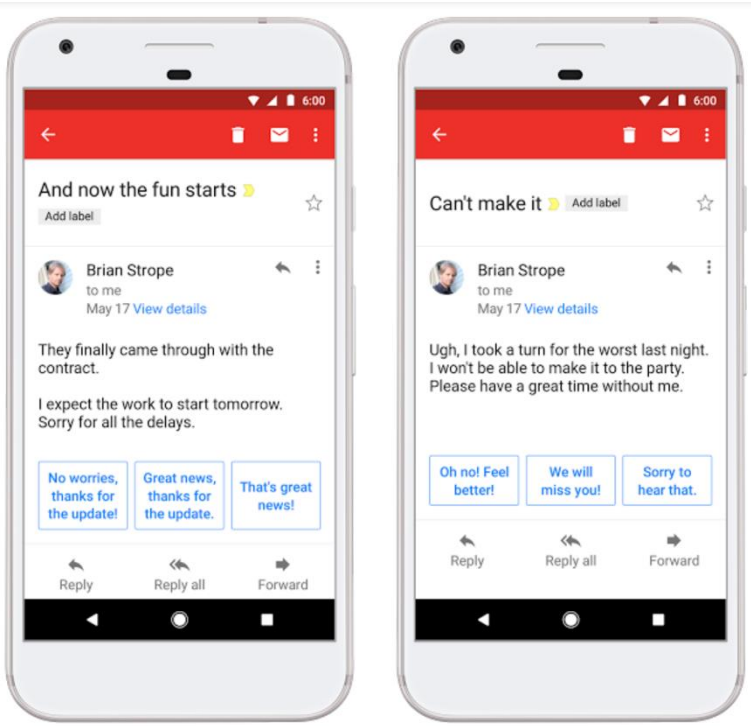
Why Care About Language Modeling?

- Language Modeling is a ~~subcomponent~~ superset of many tasks:
  - Summarization
  - Machine translation
  - Spelling correction
  - Dialogue etc.

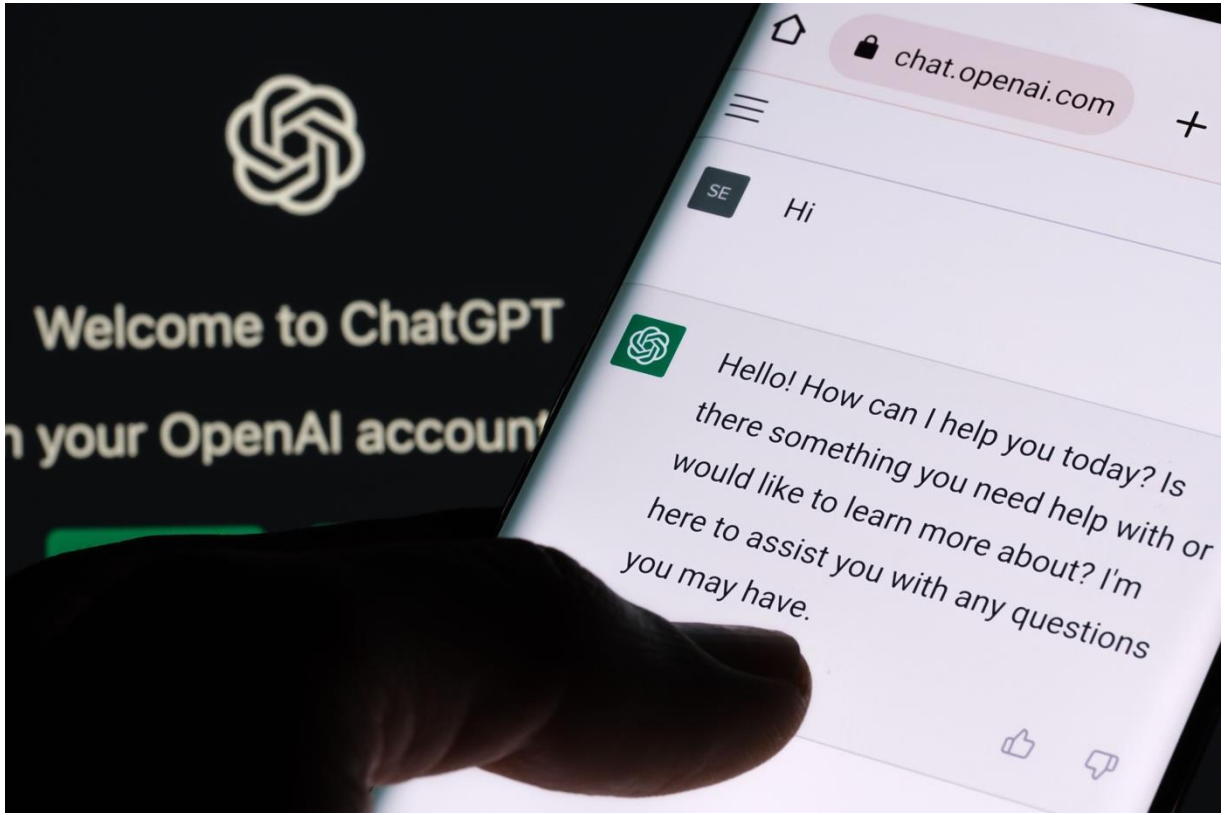
- Language Modeling is an effective proxy for language understanding.
  - Effective ability to predict forthcoming words requires on understanding of context/prefix.

You use Language Models every day!





You use Language Models every day!



Summary

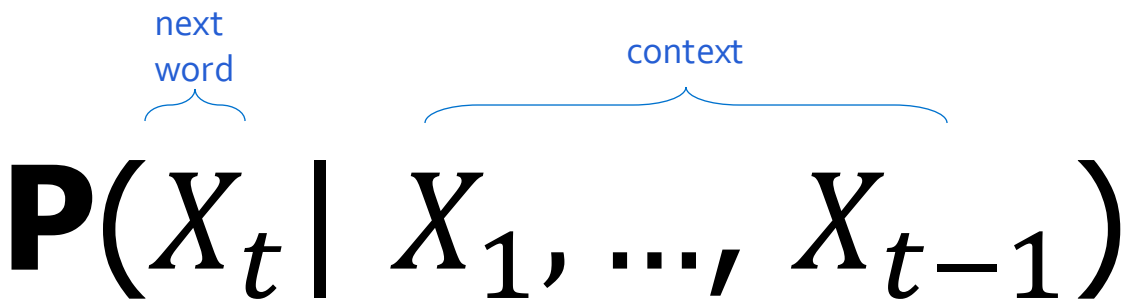
- **Language modeling:** building probabilistic distribution over language.

- An accurate distribution of language enables us to solve many important tasks that involve language communication.

- **The remaining question**: how do you actually estimate this distribution?

Language Modeling with Counting

LMs as a Marginal Distribution



- Now the question is, how to estimate this distribution.

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

How do we estimate these probabilities?
Let's just count!

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

Count how often
“the cat sat on the mat”
has appeared in the world (internet)!

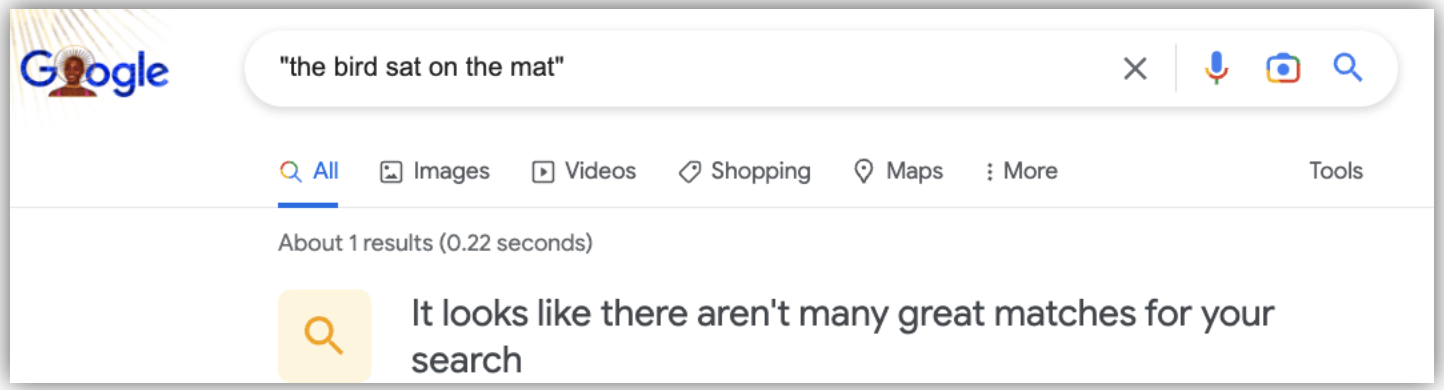
Divide that by, the count of
“the cat sat on the”
in the world (internet)!

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

How do we estimate these probabilities?

Let's just count!

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



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

How do we estimate these probabilities?
Let's just count!

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

Challenge: Increasing n makes sparsity problems worse.
Typically, we can't have n bigger than 5.

Some partial solutions (e.g., smoothing and backoffs)
though still an open problem.

Language Models: A History

- Shannon (1950): The redundancy and predictability (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) built an approximate language model with word co-occurrences.

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

1st order approximation:

1 element

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

[Prediction and Entropy of Printed English, Shanon 1950]

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

Andrey Markov

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

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

2nd order approximation:

2 elements

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

[Prediction and Entropy of Printed English, Shanon 1950]

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

Andrey Markov

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

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

3rd order approximation:

3 elements

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

[Prediction and Entropy of Printed English, Shanon 1950]

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

Andrey Markov

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

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

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

[Prediction and Entropy of Printed English, Shanon 1950]

Understanding Sparsity: A Thought Experiment

- How common are zero-probabilities? 🤔
- **Example:** Shakespeare as a text corpus
  - The size vocab used by Shakespeare: |V|=29,066
  - Shakespeare produced: ~300,000 bigrams
    - Out of |V|^2= 844 million possible bigrams
      - (some of them don't make sense, but ok!)
- So, 99.96% of the possible bigrams are never seen (hence, have zero entries for bigram counts).

[Slide credit: Mohit Iyyer]

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]

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*

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

[adopted from Chris Manning]

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

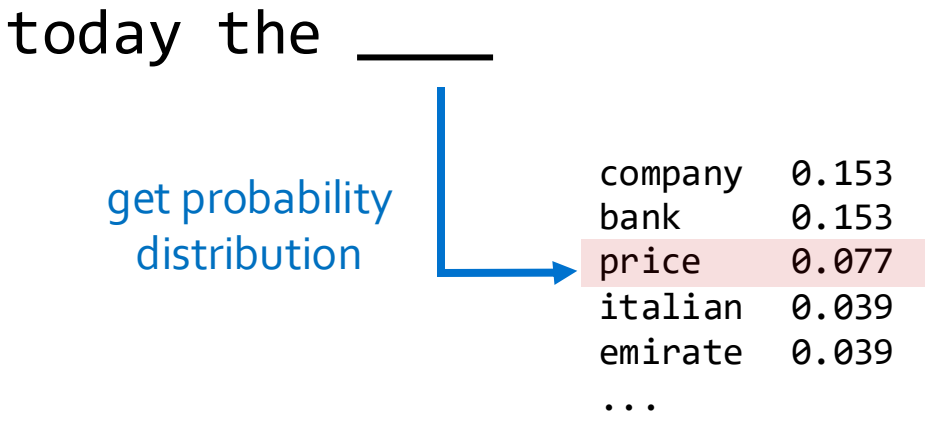
Otherwise, seems reasonable!

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

[adopted from Chris Manning]

Generation from N-Gram Models

- Now we can sample from this mode:



Sparsity problem: not much granularity in the probability distribution

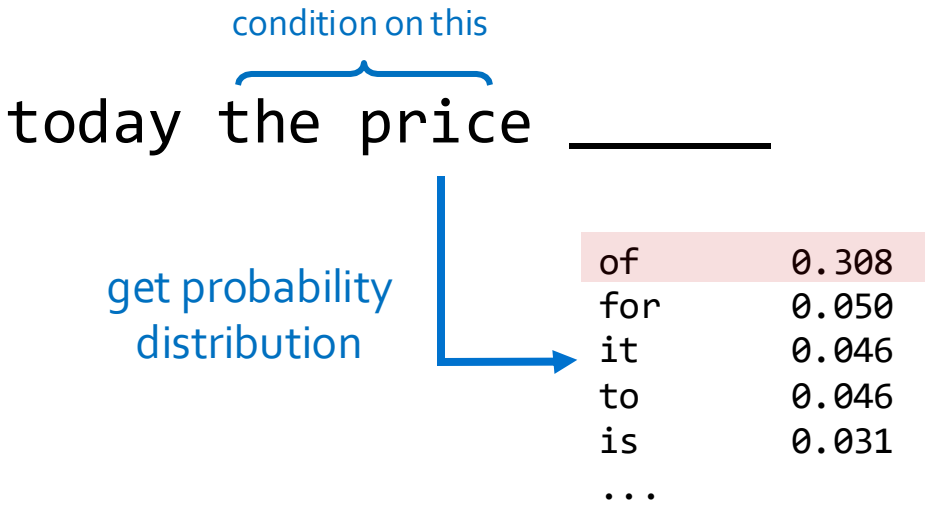
Otherwise, seems reasonable!

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

[adopted from Chris Manning]

Generation from N-Gram Models

- Now we can sample from this mode:



Sparsity problem: not much granularity in the probability distribution

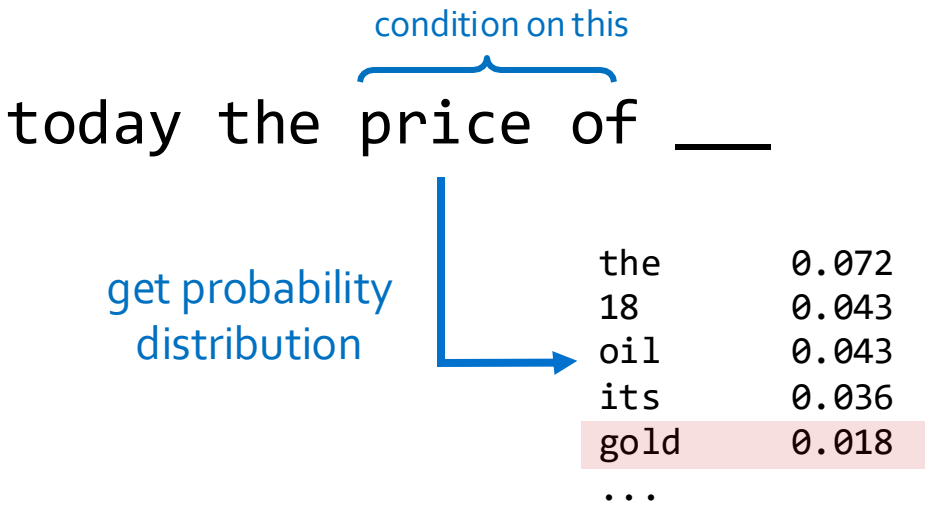
Otherwise, seems reasonable!

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

[adopted from Chris Manning]

Generation from N-Gram Models

- Now we can sample from this mode:



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]

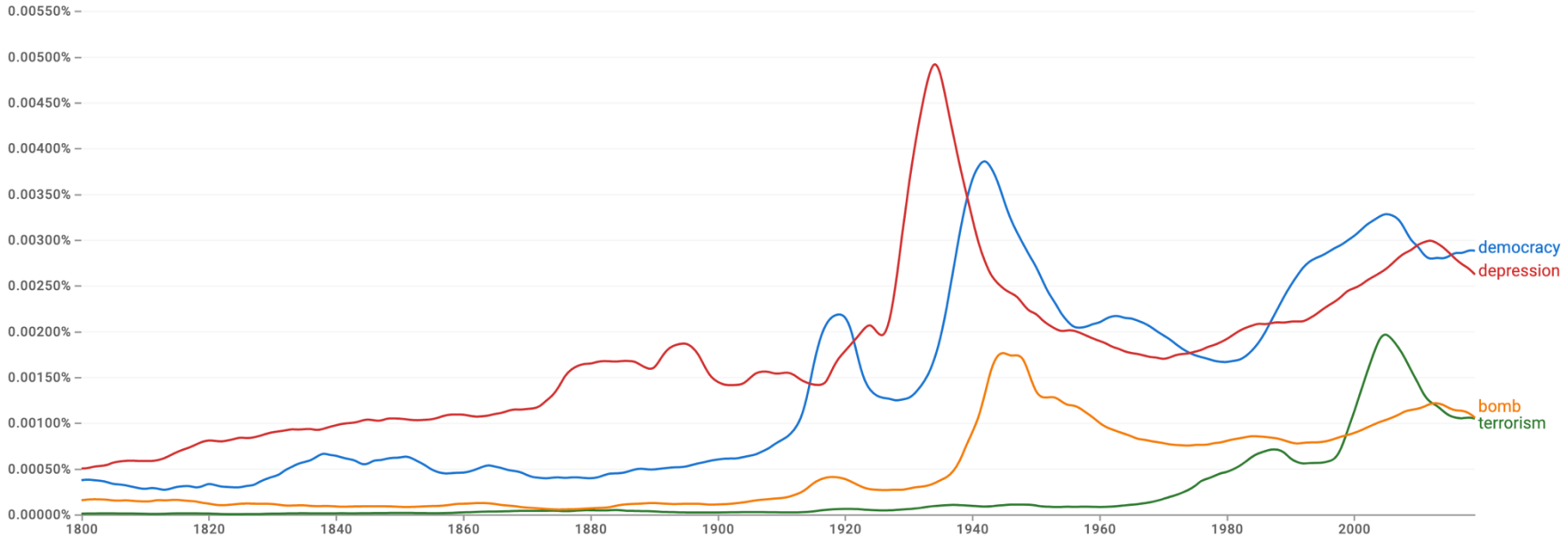
N-Grams LMs and Long-range Dependencies

- In general, count-based LMs are insufficient models of language because language has long-distance dependencies:

"The computer which I had just put into the machine room on the fifth floor crashed."

Pre-Computed N-Grams

Google Books Ngram Viewer

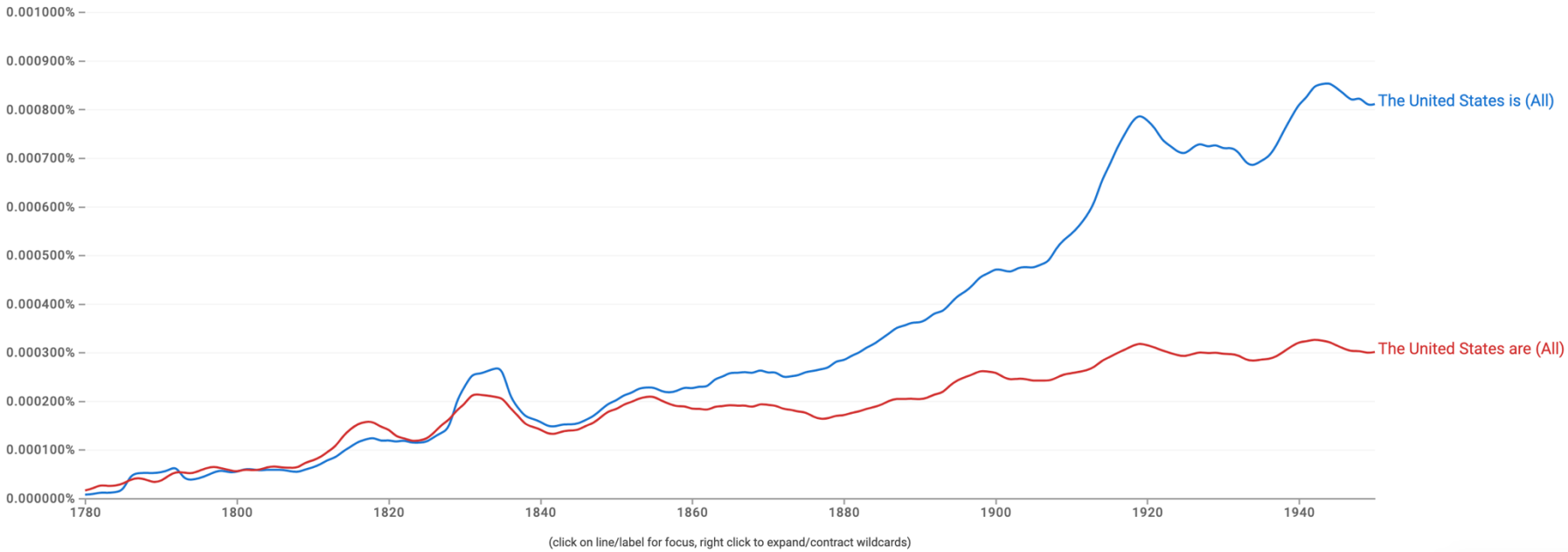


Google n-gram viewer https://books.google.com/ngrams/

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

Pre-Computed N-Grams

Google Books Ngram Viewer

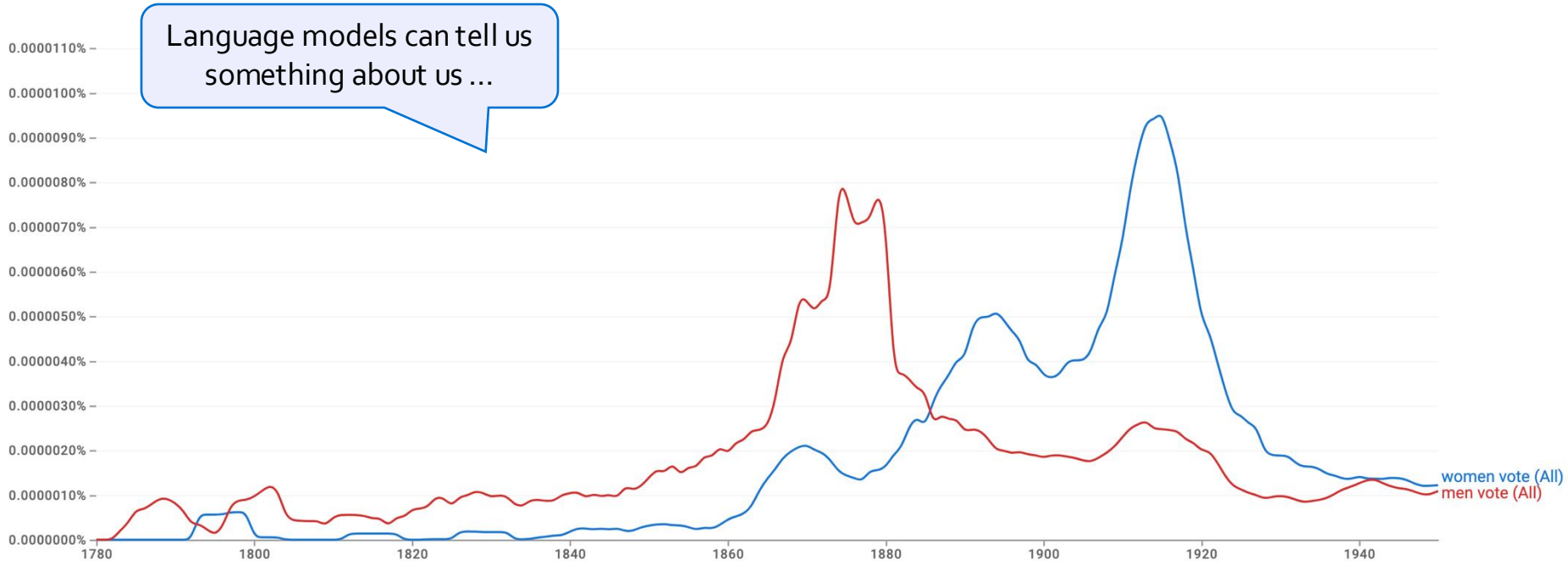


Google n-gram viewer https://books.google.com/ngrams/

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

Pre-Computed N-Grams

Google Books Ngram Viewer

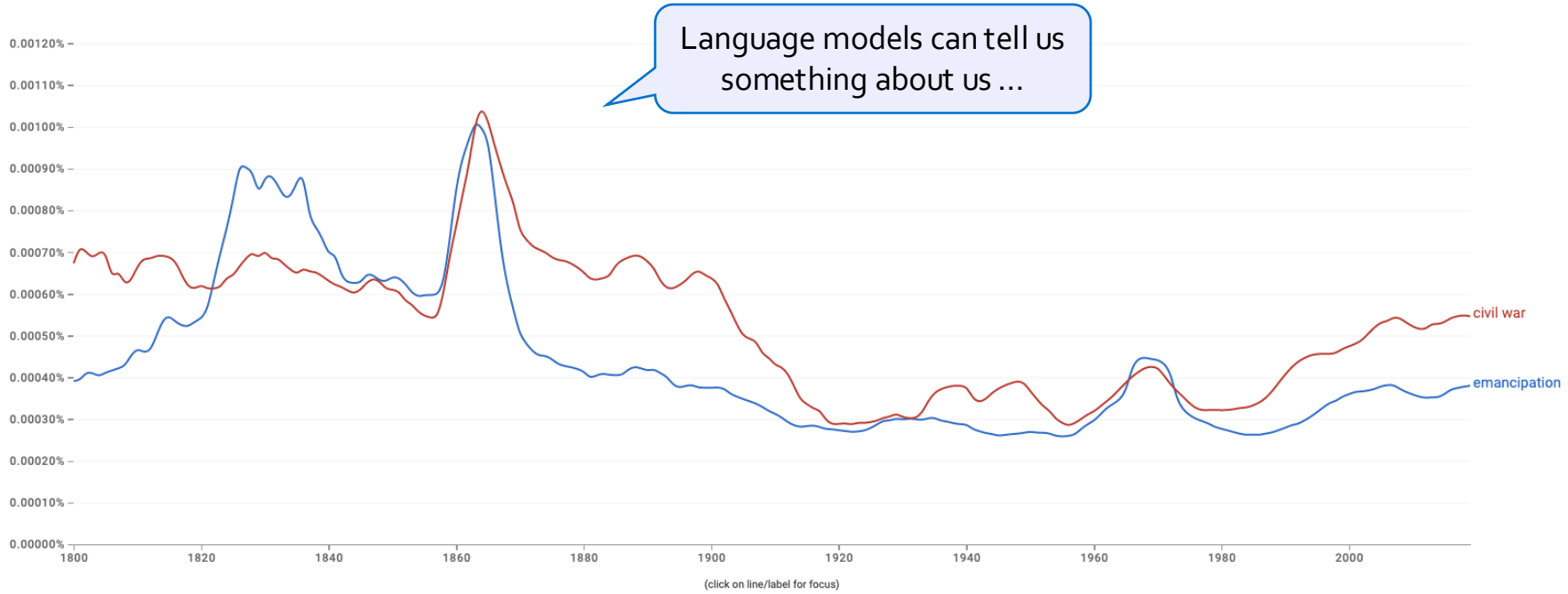


Google n-gram viewer https://books.google.com/ngrams/

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

N-Gram Language Models, A Historical Highlight

"Every time I fire a linguist, the performance of the speech recognizer goes up"!!

Fred Jelinek
(1932-2010)

- Probabilistic n-gram models of text generation [Jelinek+ 1980's, …]
  - Applications: Speech Recognition, Machine Translation

Continuous Speech Recognition by Statistical Methods

FREDERICK JELINEK, FELLOW, IEEE

Abstract—Statistical methods useful in automatic recognition of continuous speech are described. They concern modeling of a speaker and of an acoustic processor, extraction of the models' statistical parameters, and hypothesis search procedures and likelihood computations of linguistic decoding. Experimental results are presented that indicate the power of the methods.

utterance models used will incorporate more grammatical features, and statistics will have been grafted onto grammatical models. Most methods presented here concern modeling of the speaker's and acoustic processor's performance and should, therefore, be universally useful.

Automatic recognition of continuous (English) speech is an

Summary

- Learning a language model ~ learning conditional probabilities over language.
- One approach to estimating these probabilities: counting word co-occurrences.

- Challenges:
  - Word co-occurrences become rare for long sequences. (the sparsity issue)
  - But language understanding requires long-range dependencies.

- We need a better alternative! 😖

- **Next:** Measuring quality of language models.

How Good are Language Models?

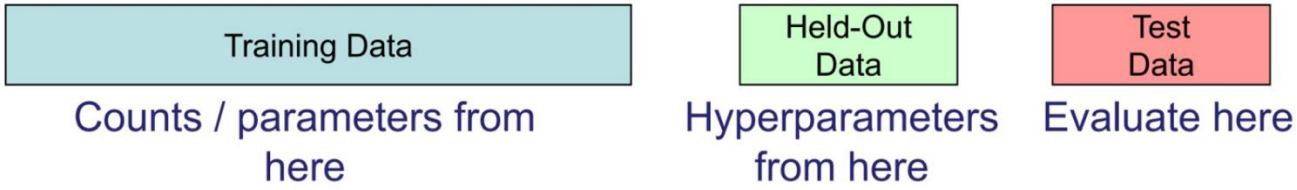
Evaluating Language Models

- Does our language model prefer good sentences to bad ones?
  - Assign higher probability to "real" or "frequently observed" sentences
  - Than "ungrammatical" or "rarely observed" sentences?
- We test the model's performance on data we haven't seen.

Evaluating Language Models

Setup:

- Train it on a suitable training documents.
- Evaluate their predictions on different, unseen documents.
- An evaluation metric tells us how well our model does on the test set.



train

count("on the mat")

Evaluating Language Models: Example

Setup:

- Train it on a suitable training documents.
- Evaluate their predictions on different, unseen documents.
- An evaluation metric tells us how well our model does on the test set.

Example: I use a bunch of New York Times articles to build a bigram probability table

train

count("on the mat")

A good language model should assign a high probability to held-out text!

eval

Now I'm going to evaluate the probability of some heldout data using our bigram table

Be Careful About Data Leakage!

Advice from a grandpa:

- Don't allow test sentences to leak into into training set.
- Otherwise, you will assign it an artificially high probability (==cheating).

Example: I use a bunch of New York Times articles to build a bigram probability table

train

count("on the mat")

A good language model should assign a high probability to held-out text!

eval

Now I'm going to evaluate the probability of some heldout data using our bigram table

Evaluating Language Models: Intrinsic vs Extrinsic

- Intrinsic: measure how good we are at modeling language
- Extrinsic: build a new language model, use it for some task (MT, ASR, etc.)

Example: I use a bunch of New York Times articles to build a bigram probability table

train

count("on the mat")

extrinsic eval

Google Translate

eval

Now I'm going to evaluate the probability of some heldout data using our bigram table

Evaluation Metric for Language Modeling: Perplexity

- **Perplexity** is the inverse probability of the test set, normalized by the number of words:

$$\text{ppl}(w_1, \ldots, w_n) = \mathbf{P}(w_1, w_2, \ldots, w_n)^{-\frac{1}{n}}$$

- A measure of predictive quality of a language model.
- Minimizing perplexity is the same as maximizing probability

Evaluation Metric for Language Modeling: Perplexity

- **Perplexity** is the inverse probability of the test set, normalized by the number of words:

$$\text{ppl}(w_1, \ldots, w_n) = \mathbf{P}(w_1, w_2, \ldots, w_n)^{-\frac{1}{n}}$$

Evaluation Metric for Language Modeling: Perplexity

- **Perplexity** is the inverse probability of the test set, normalized by the number of words:

$$\text{ppl}(w_1, \ldots, w_n) = \mathbf{P}(w_1, w_2, \ldots, w_n)^{-\frac{1}{n}}$$

$$= \sqrt[n]{\frac{1}{\mathbf{P}(w_1, w_2, \ldots, w_n)}} = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{\mathbf{P}(w_i | w_{<i})}} \quad \text{chain rule}$$

$$= 2^H, \text{where}$$

$$H = -\frac{1}{n} \sum_{i=1}^{n} \log_2 \mathbf{P}(w_i | w_1, \ldots, w_{i-1})$$

Evaluation Metric for Language Modeling: Perplexity

- In practice, we prefer to use log-probabilities (also known as "logits")
- We can rewrite perplexity formula in terms of log-probs:

$$\text{ppl}(w_1, \ldots, w_n) = 2^H, \text{ where } H = -\frac{1}{n}\sum_{i=1}^{n} \log_2 \mathbf{P}(w_i | w_1, \ldots, w_{i-1})$$

Recap: Definition of cross-entropy between two distributions:

$$H(p, q) = -\sum_{x \in \mathcal{X}} p(x) \log q(x)$$

Can be interpreted as cross-entropy between LM prob and language prob. **Why?**

Evaluation Metric for Language Modeling: Perplexity

- In practice, we prefer to use log-probabilities (also known as "logits")
- We can rewrite perplexity formula in terms of log-probs:

$$\text{ppl}(w_1, \ldots, w_n) = 2^H, \text{ where } H = -\frac{1}{n}\sum_{i=1}^{n} \log_2 \mathbf{P}(w_i | w_1, \ldots, w_{i-1})$$

- **Perplexity** for n-grams:
  - Unigrams: $H = -\frac{1}{n}\sum_{i=1}^{n} \log_2 \mathbf{P}(w_i)$
  - Bigrams: $H = -\frac{1}{n}\sum_{i=1}^{n} \log_2 \mathbf{P}(w_i | w_{i-1})$
  - Trigrams: $H = -\frac{1}{n}\sum_{i=1}^{n} \log_2 \mathbf{P}(w_i | w_{i-2}, w_{i-1})$
  - …

Intuition-building Quizzes (1)

- In practice, we prefer to use log-probabilities (also known as "logits")
- We can rewrite perplexity formula in terms of log-probs:

$$\text{ppl}(w_1, \ldots, w_n) = 2^H, \text{ where } H = -\frac{1}{n}\sum_{i=1}^{n} \log_2 \mathbf{P}(w_i | w_1, \ldots, w_{i-1})$$

- **Quiz:** let's suppose we have a sentence $w_1, \ldots, w_n$ and it's fixed. Our model will correctly guess each word with probability 1/5. What is perplexity of our model?

$$H = -\frac{1}{n}\left[\log_2\left(\frac{1}{5}\right) + \cdots + \log_2\left(\frac{1}{5}\right)\right] = -\log\left(\frac{1}{5}\right) \Rightarrow \text{ppl(D)} = 5$$

Intuition: the model is indecisive among 5 choices.

Intuition-building Quizzes (2)

- In practice, we prefer to use log-probabilities (also known as "logits")
- We can rewrite perplexity formula in terms of log-probs:

$$\text{ppl}(w_1, \ldots, w_n) = 2^H, \text{ where } H = -\frac{1}{n}\sum_{i=1}^{n} \log_2 \mathbf{P}(w_i | w_1, \ldots, w_{i-1})$$

- **Quiz:** let's we evaluate an exact (!!) model of language, i.e., our model always knows what exact word should follow a given context. What is the perplexity of this model?

$$\forall w \in V: \mathbf{P}(w_i | w_{1:i-1}) = 1 \;\Rightarrow\; \text{ppl}(D) = 2^{-\frac{1}{2} n \log_2 1} = 1$$

Intuition: the model is indecisive among 1 (the right!) choice!

Intuition-building Quizzes (3)

- In practice, we prefer to use log-probabilities (also known as "logits")
- We can rewrite perplexity formula in terms of log-probs:

$$\text{ppl}(w_1, \ldots, w_n) = 2^H, \text{ where } H = -\frac{1}{n}\sum_{i=1}^{n} \log_2 \mathbf{P}(w_i | w_1, \ldots, w_{i-1})$$

- **Quiz:** let's we evaluate a confused (!!) model of language, i.e., our model has no idea what word should follow each context—it always chooses a uniformly random word. What is the perplexity of this model?

$$\forall w \in V: \mathbf{P}(w | w_{1:i-1}) = \frac{1}{|V|} \Rightarrow \text{ppl}(D) = 2^{-\frac{1}{n} n \log_2 \frac{1}{|V|}} = |V|$$

Intuition: the model is indecisive among all the vocabulary terms.

Perplexity: Summary

$$\text{ppl}(w_1, \ldots, w_n) = 2^H, \text{ where } H = -\frac{1}{n}\sum_{i=1}^{n} \log_2 \mathbf{P}(w_i | w_1, \ldots, w_{i-1})$$

- Perplexity is a measure of model's uncertainty about next word (aka "average branching factor").
  - The larger the number of vocabulary, the more options there to choose from.
  - (the choice of atomic units of language impacts PPL — more on this later)

- Perplexity ranges between 1 and |V|.

- We prefer LMs with lower perplexity.

Lower perplexity == Better Model

- Training on 38 million words, test 1.5 million words, Wall Street Journal

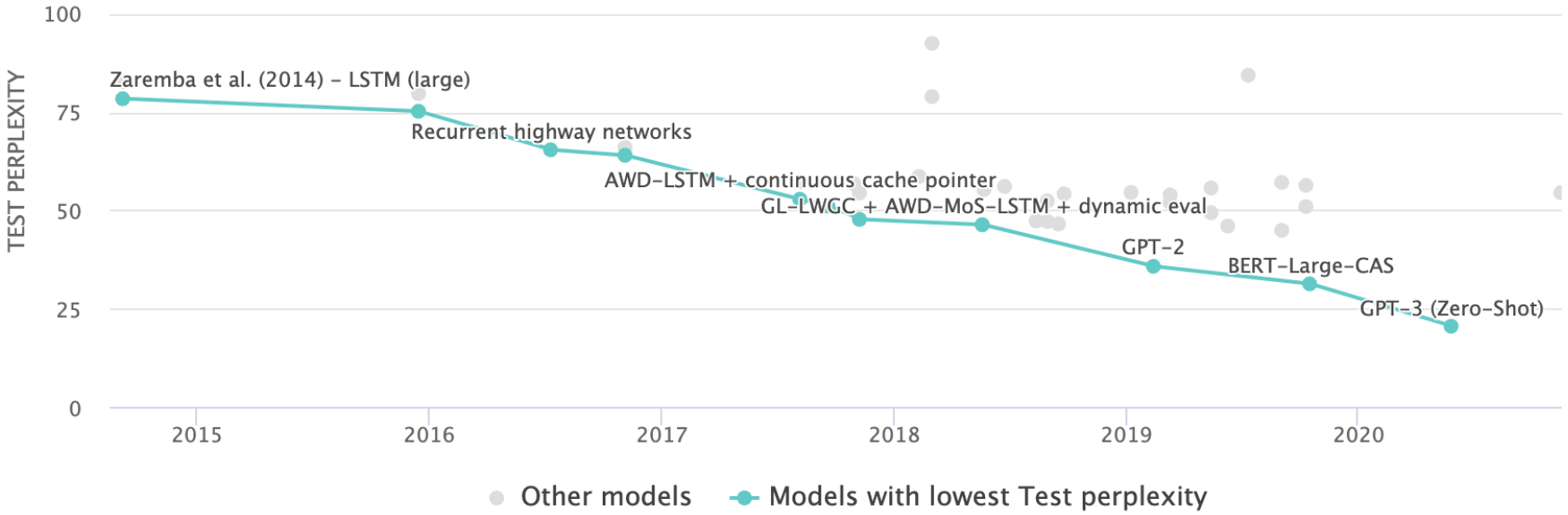
N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Lower is better

Note these evaluations are done on data that was not used for "counting." (no cheating!!)

Lower perplexity == Better Model

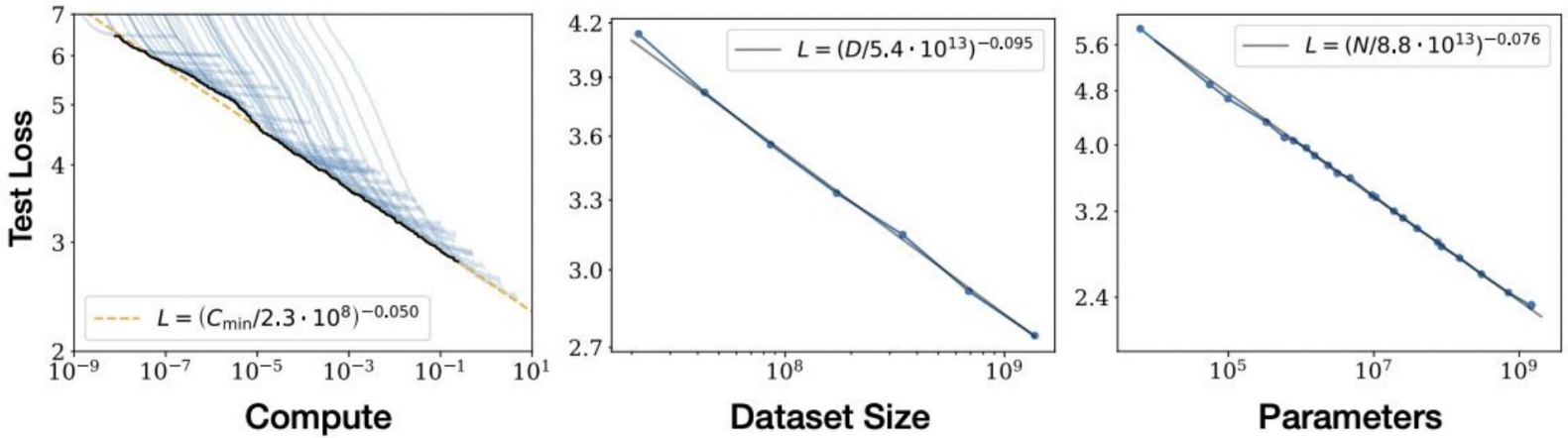
The PPL of modern language models have consistently been going down.



[Language Modelling on Penn Treebank (Word Level)]

Lower perplexity == Better Model

The PPL of modern language models have consistently been going down.



[Scaling Laws, Jared Kaplan et al.]

Summary

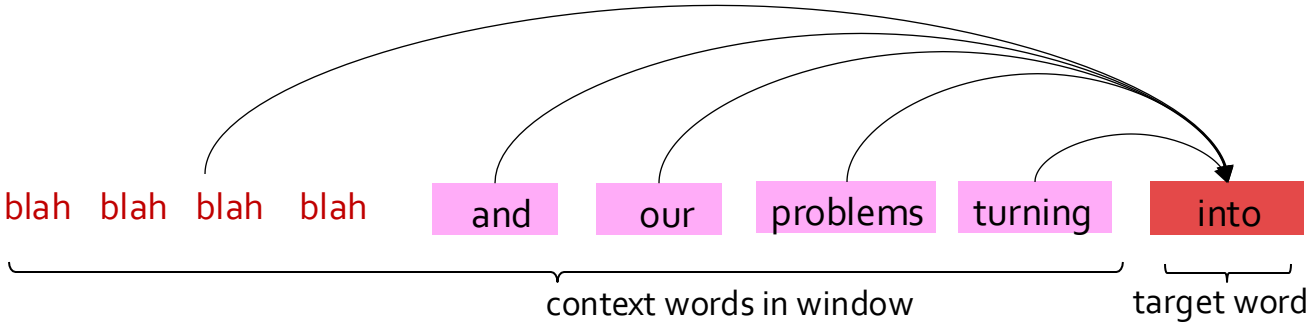
- Language Models (LM): distributions over language
- Measuring LM quality: use perplexity on held-out data.
- Count-based LMs have limitations.
  - Challenge with large N's: sparsity problem — many zero counts/probs.
  - Challenge with small N's: lack of long-range dependencies.
- Next: Rethinking language modeling as a statistical learning problem.

Beyond Counting:
Language Models as a Learning Problem

LM as a Machine Learning Problem

- Given the embeddings of the context, predict the word on the right side.
  - Dropping the right context for simplicity -- not a fundamental limitation.

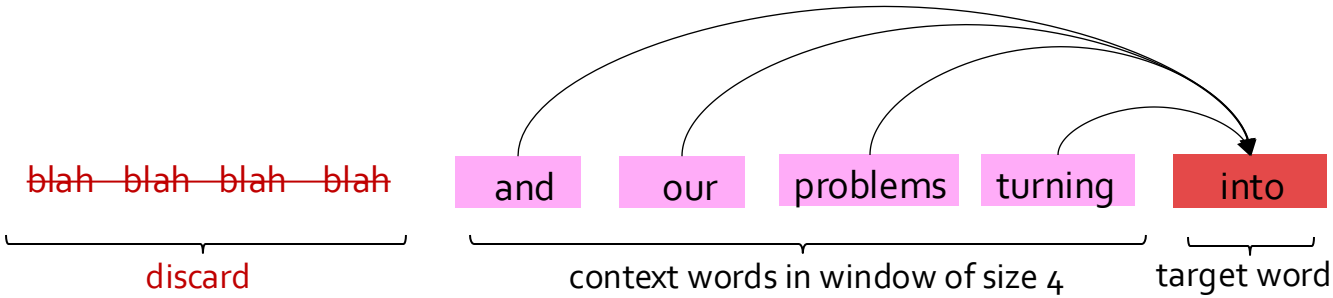
- Discard anything beyond its context window



LM as a Machine Learning Problem

- Given the embeddings of the context, predict the word on the right side.
  - Dropping the right context for simplicity -- not a fundamental limitation.

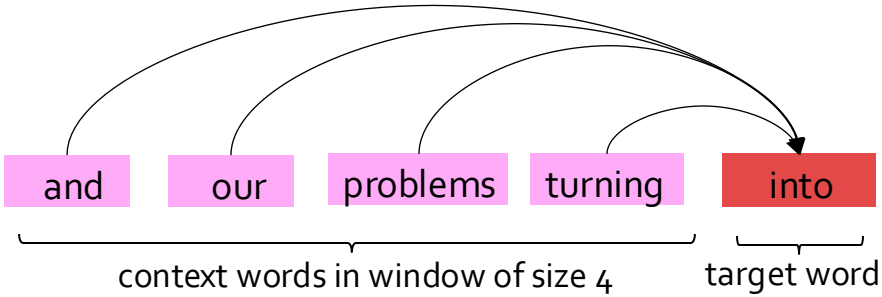
- Discard anything beyond its context window



LM as a Machine Learning Problem

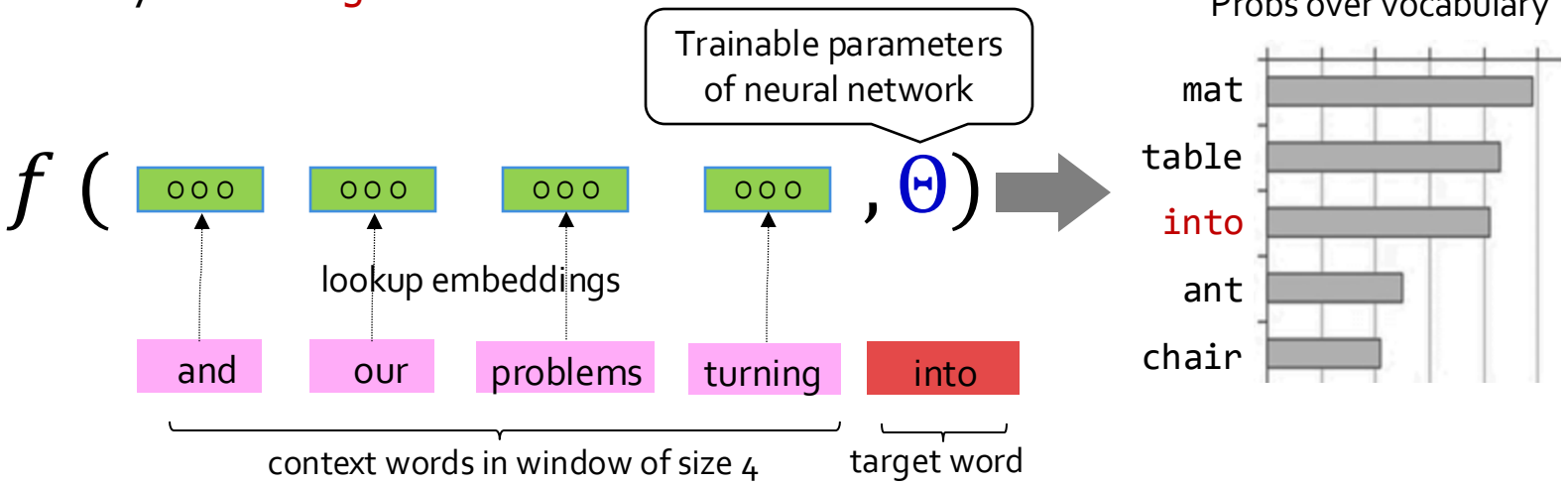
- Given the embeddings of the context, predict the word on the right side.
  - Dropping the right context for simplicity -- not a fundamental limitation.

- Discard anything beyond its context window



A Fixed-Window Neural LM

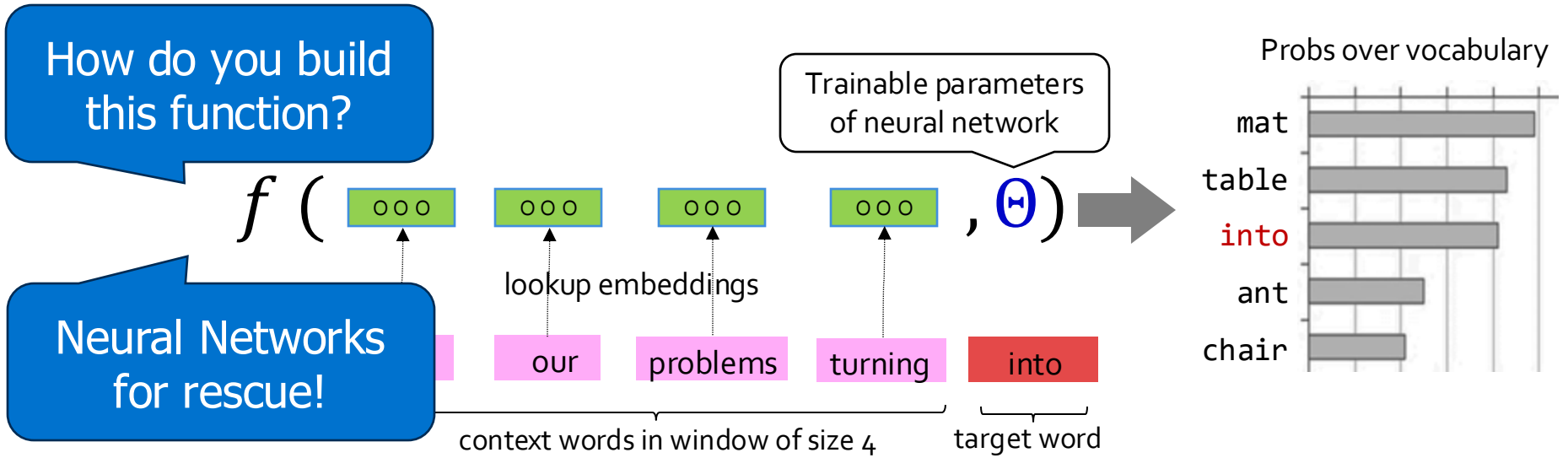
- Given the embeddings of the context, predict a target word on the right side.
  - Dropping the right context for simplicity -- not a fundamental limitation.
- Training this model is basically optimizing its parameters θ such that it assigns high probability to the target word.



[Bengio et al. 2003]

A Fixed-Window Neural LM

- It will also lay the foundation for the future models (recurrent nets, transformers, ...)
- But first we need to figure out how to train neural networks!



[Bengio et al. 2003]

From Counting (N-Gram) to Neural Models

- n-gram models of text generation [Jelinek+ 1980's, …]
  - Applications: Speech Recognition, Machine Translation

- "Shallow" statistical/neural language models (2000's) [Bengio+ 1999 & 2001, …]

NeurIPS 2000

A Neural Probabilistic Language Model

Yoshua Bengio; Réjean Ducharme and Pascal Vincent
Département d'Informatique et Recherche Opérationnelle
Centre de Recherche Mathématiques
Université de Montréal
Montréal, Québec, Canada, H3C 3J7
{*bengioy,ducharme,vincentp*}*@iro.umontreal.ca*

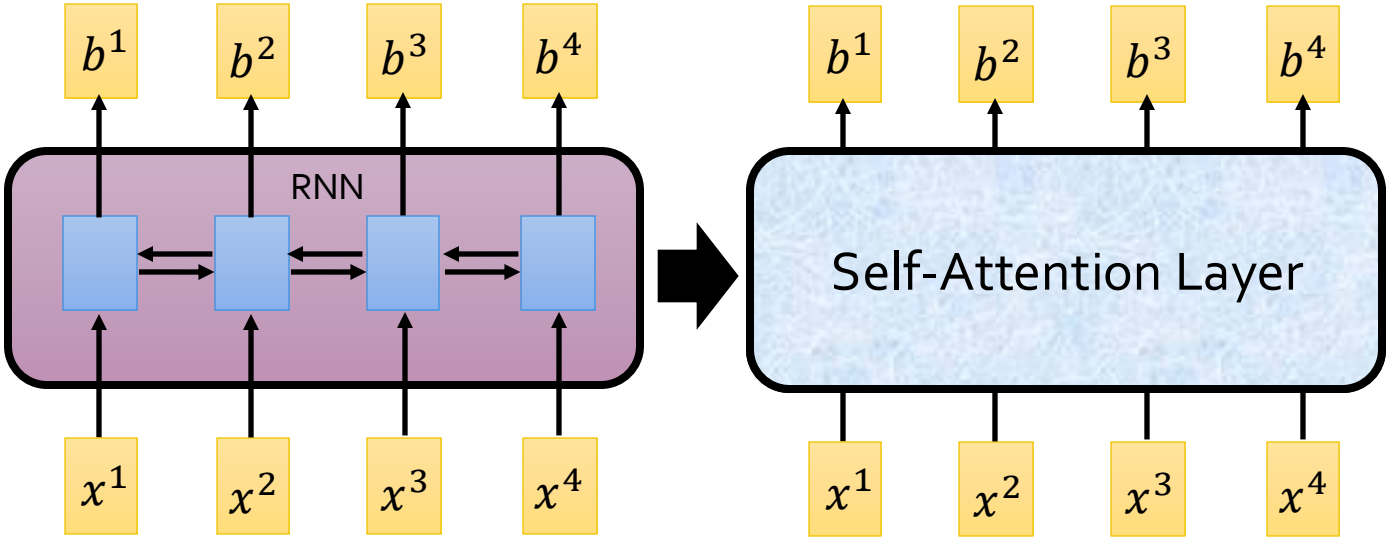
Summary

- Language Modeling (LM), a useful predictive objective for language
- Perplexity, a measure of an LM's predictive ability
- N-gram models (~1980 to early 2000's),
  - Early instances of LMs
  - Difficult to scale to large window sizes
- Shallow neural LMs (early and mid-2000's),
  - We will need in coming sessions that one can build these models with neural networks.
  - These will be effective predictive models based on feed-forward networks

Self-Attention

Self-Attention

- b^i is obtained based on the whole input sequence.
- can be parallelly computed.

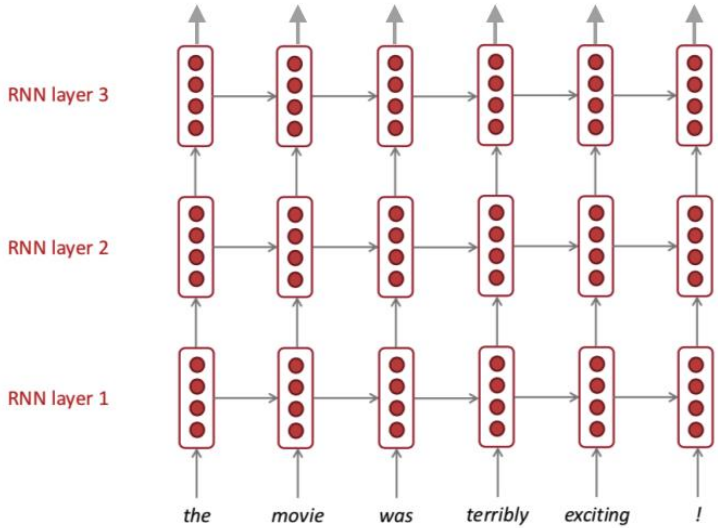


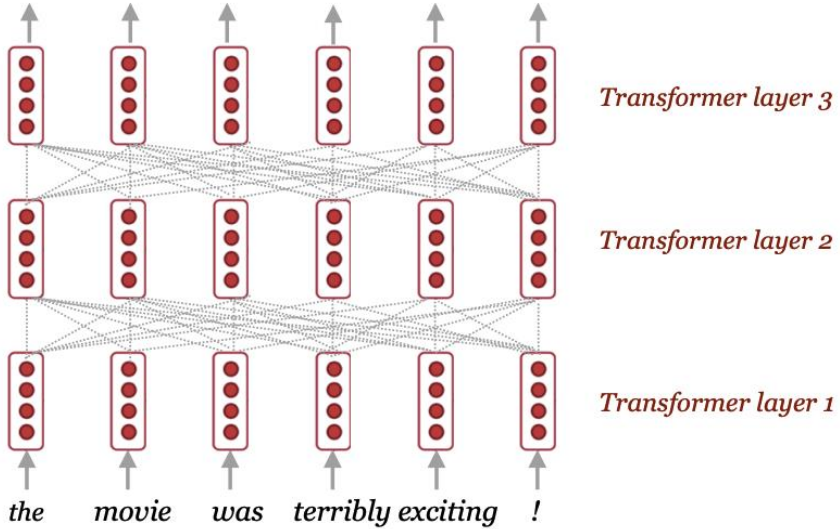
Idea: replace any thing done by RNN with self-attention.

"Neural machine translation by jointly learning to align and translate" Bahdanau etl. 2014;
"Attention is All You Need" Vaswani et al. 2017

[adopted from Hung-yi Lee]

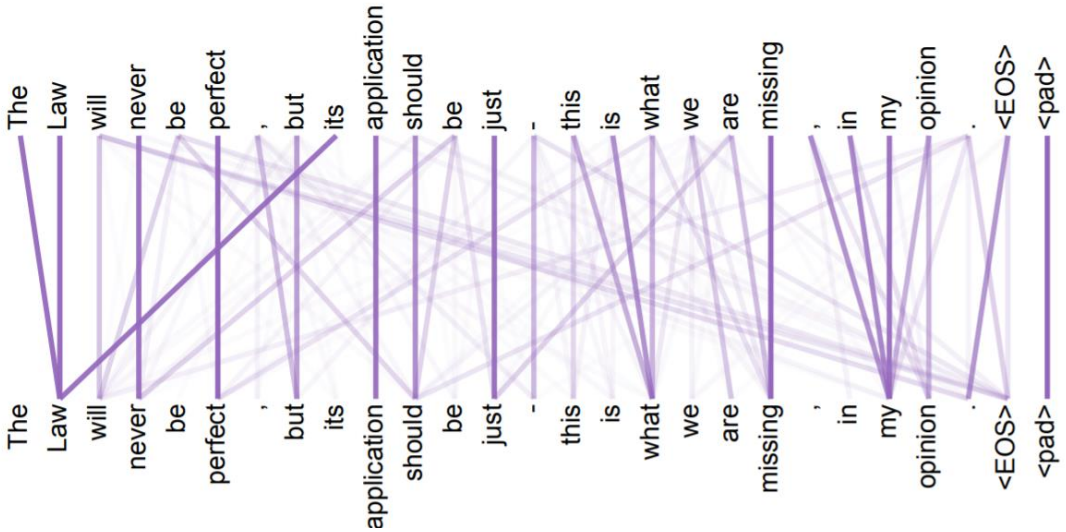
RNN vs Transformer





Attention

- Core idea: build a mechanism to focus ("attend") on a particular part of the context.



[Attention Is All You Need, Vaswani et al. 2017]

Defining Self-Attention

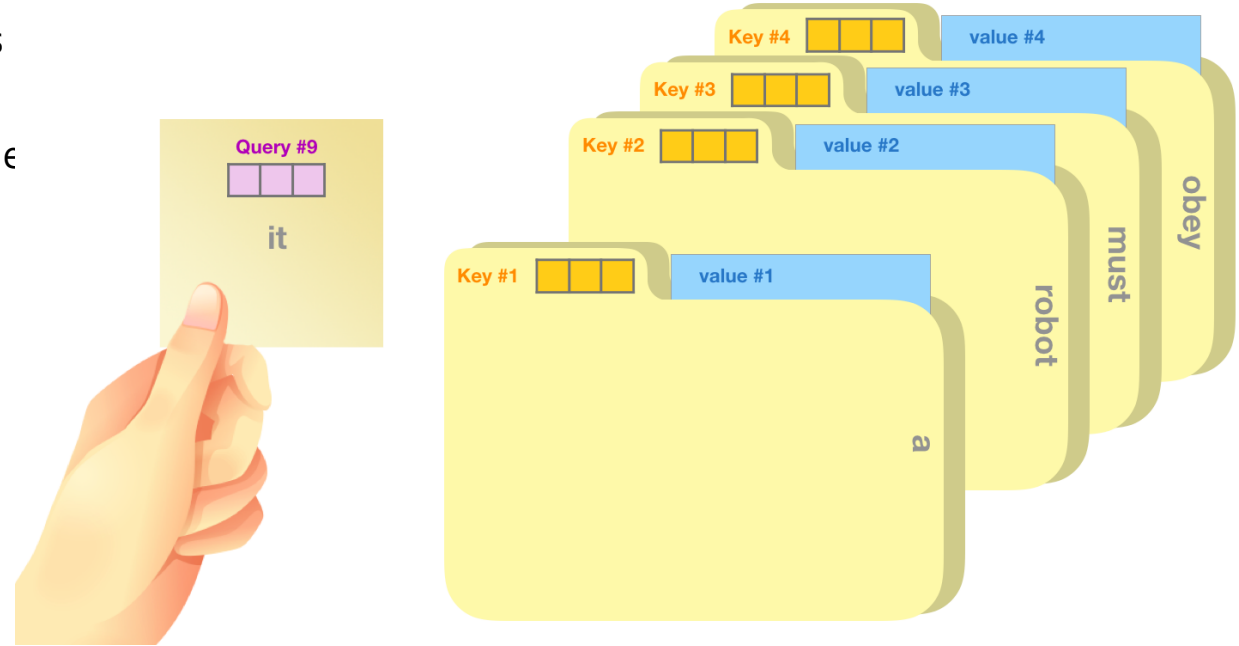
- Terminology:
  - Query: to match others
  - Key: to be matched
  - Value: information to be extracted

[Vaswani et al. 2017: https://arxiv.org/abs/1706.03762]

Defining Self-Attention

An analogy ....

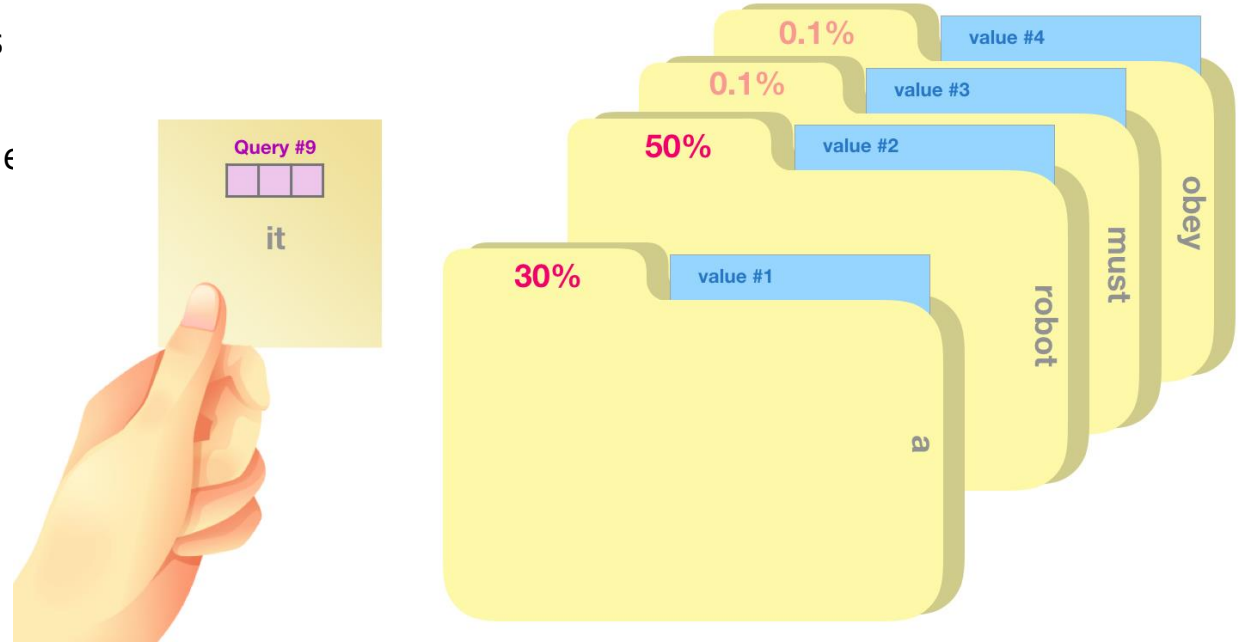
- Terminology:
  - Query: to match others
  - Key: to be matched
  - Value: information to be



[Vaswani et al. 2017: https://arxiv.org/abs/1706.03762]

Defining Self-Attention

- Terminology:
  - Query: to match others
  - Key: to be matched
  - Value: information to be



[Vaswani et al. 2017: https://arxiv.org/abs/1706.03762]

q: query (to match others)

$$q_i = W^q x_i$$

k: key (to be matched)

$$k_i = W^k x_i$$

v: value (information to be extracted)

$$v_i = W^v x_i$$

q: query (to match others)

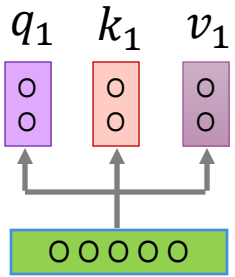
$$q_i = W^q x_i$$

k: key (to be matched)

$$k_i = W^k x_i$$

v: value (information to be extracted)

$$v_i = W^v x_i$$



The

q: query (to match others)

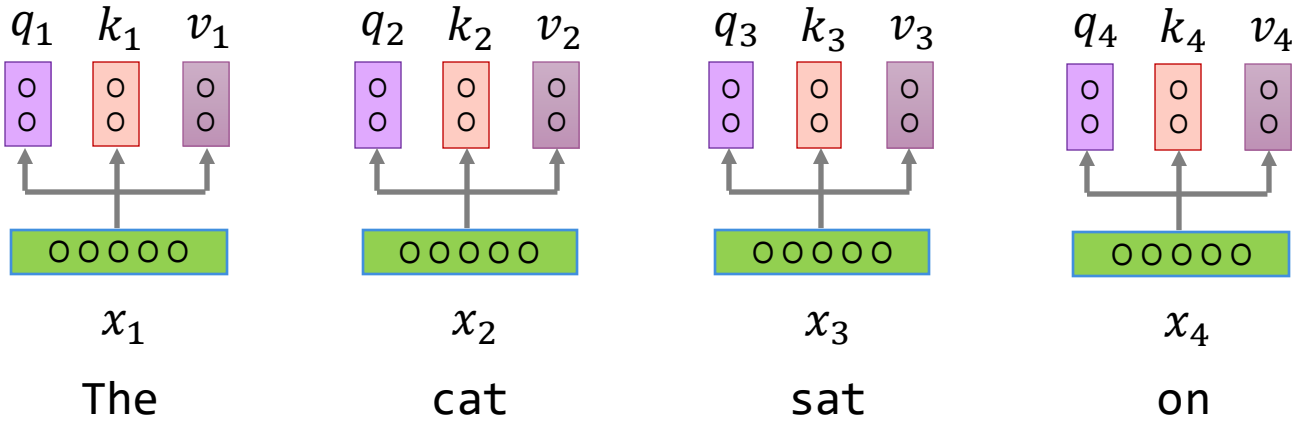
$$q_i = W^q x_i$$

k: key (to be matched)

$$k_i = W^k x_i$$

v: value (information to be extracted)

$$v_i = W^v x_i$$



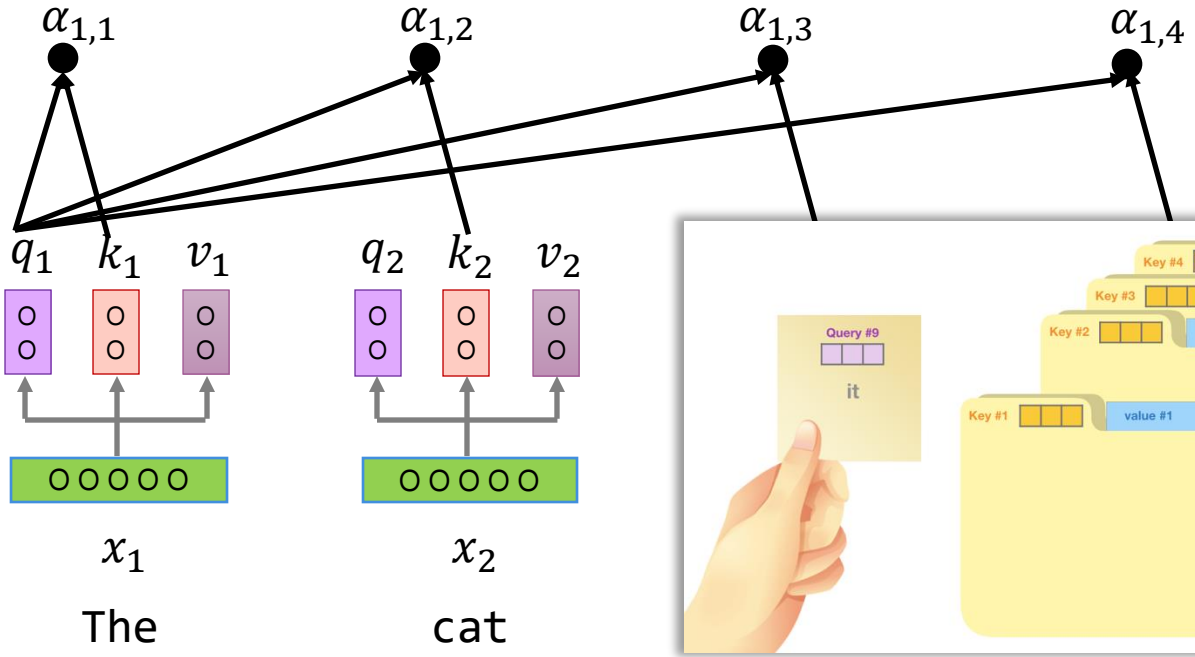
$$\alpha_{1,i} = \underbrace{q^1 \cdot k^i / \sqrt{d}}_{\text{Scaled dot product}}$$

q: query (to match others)

k: key (to be matched)

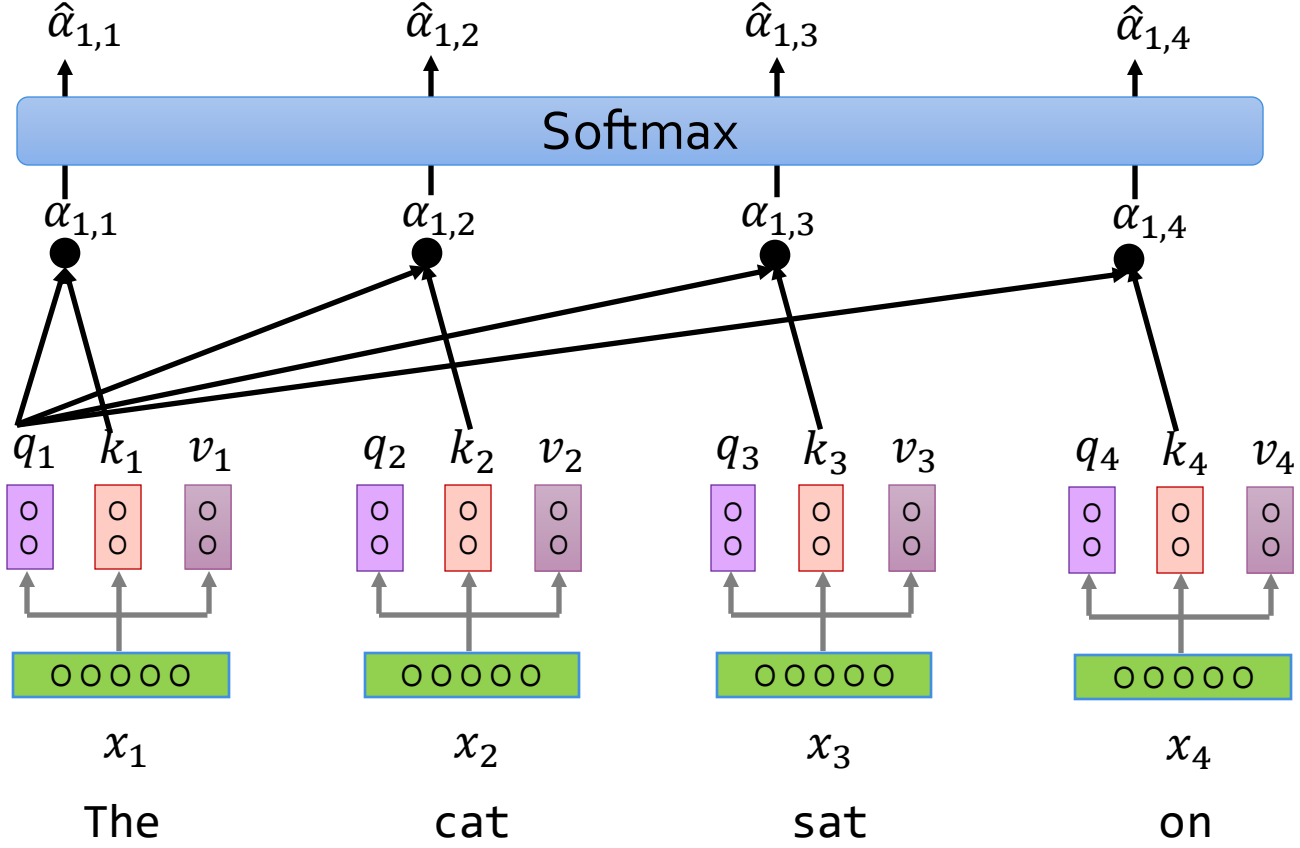
v: value (information to be extracted)

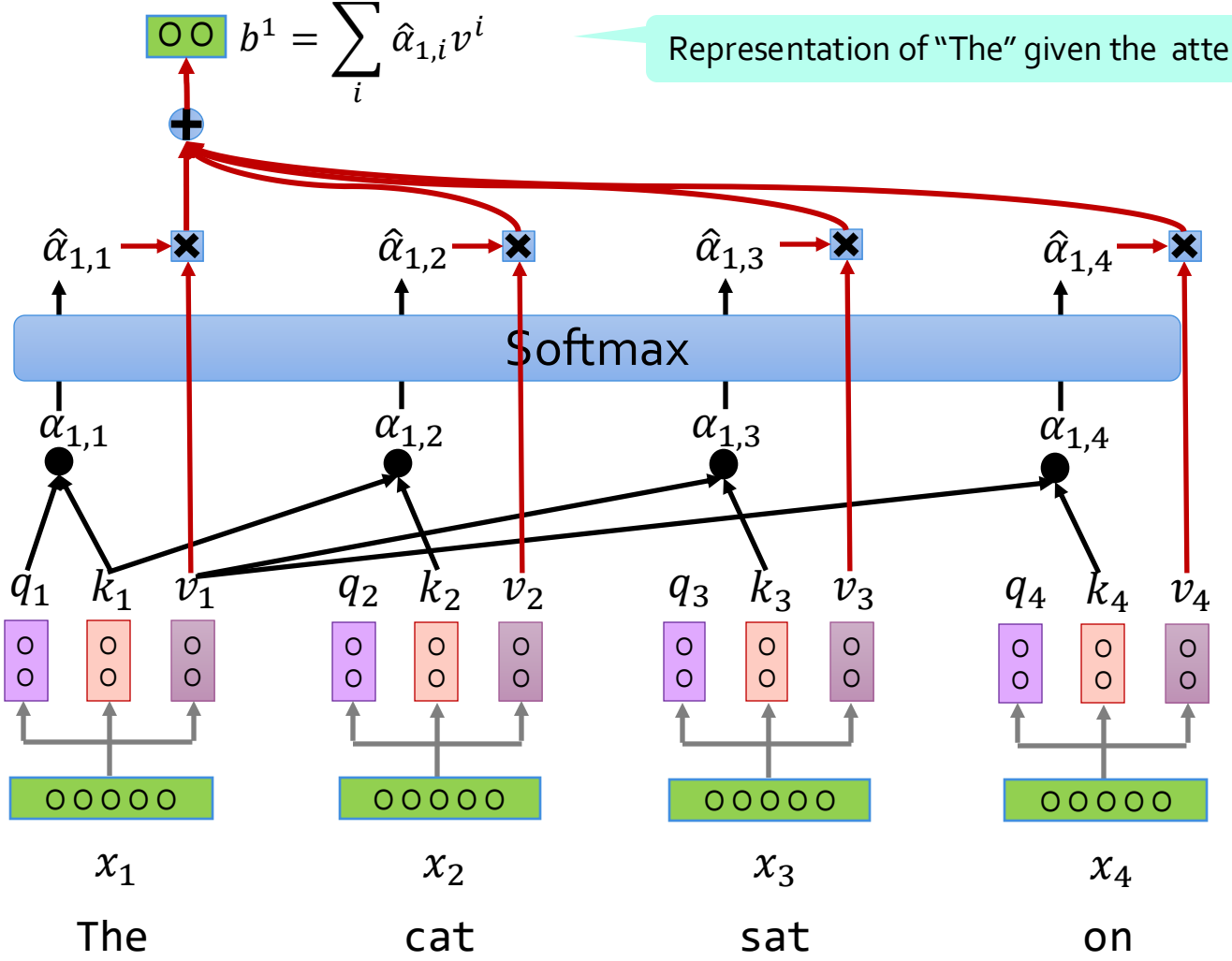
How much should "The" attend to other positions?



$$\sigma(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

How much should "The" attend to other positions?




$b^1 = \sum_i \hat{\alpha}_{1,i} v^i$
Representation of "The" given the attention weights
$\hat{\alpha}_{1,1}$
$\hat{\alpha}_{1,2}$
$\hat{\alpha}_{1,3}$
$\hat{\alpha}_{1,4}$
Softmax
$\alpha_{1,1}$
$\alpha_{1,2}$
$\alpha_{1,3}$
$\alpha_{1,4}$
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
x_1 x_2 x_3 x_4
The cat sat on

Question

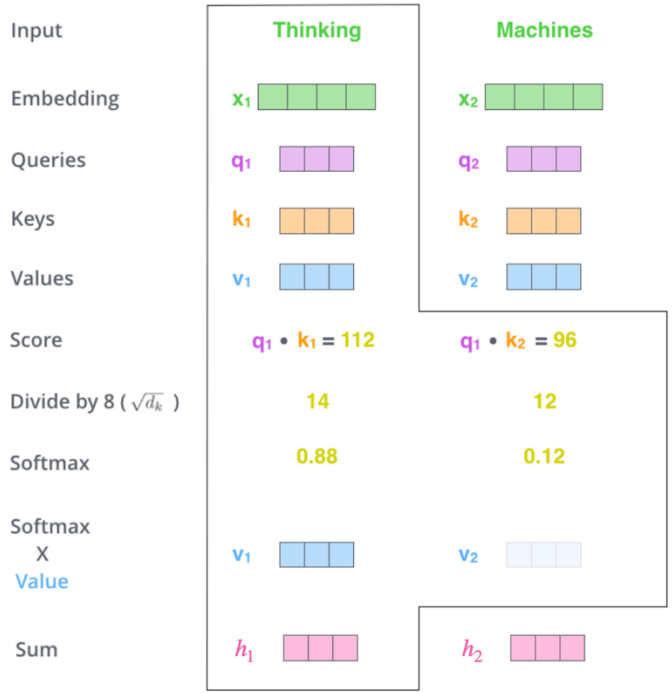
- What would be the output vector for the word "Thinking"?

(a) $0.5\mathbf{v}_1 + 0.5\mathbf{v}_2$

(b) $0.54\mathbf{v}_1 + 0.46\mathbf{v}_2$

(c) $0.88\mathbf{v}_1 + 0.12\mathbf{v}_2$

(d) $0.12\mathbf{v}_1 + 0.88\mathbf{v}_2$



[Slide credit: Danqi Chen]

Self-Attention: Matrix Notation

$X \in \mathbb{R}^{n \times d_1}$ (n = input length)

$Q = XW^Q \quad K = XW^K \quad V = XW^V$

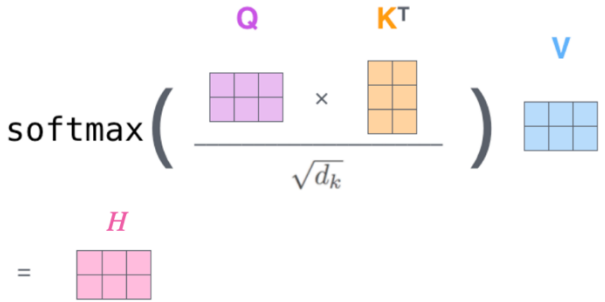
$W^Q \in \mathbb{R}^{d_1 \times d_q}, W^K \in \mathbb{R}^{d_1 \times d_k}, W^V \in \mathbb{R}^{d_1 \times d_v}$

$n \times d_q$ $d_k \times n$

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

$n \times$

Q: What is this softmax operation?



[Slide credit: Danqi Chen]

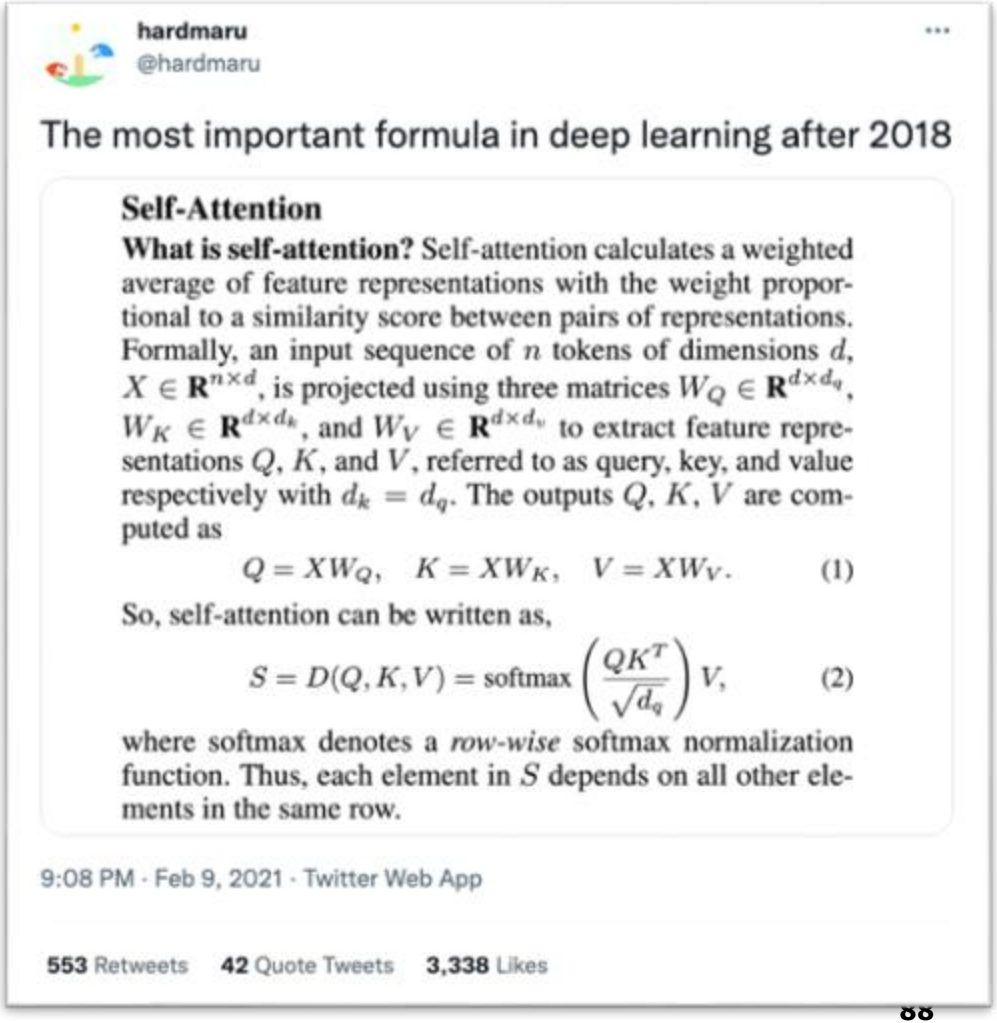
Self-Attention

- Can write it in matrix form:

- Given input $\mathbf{x}$:

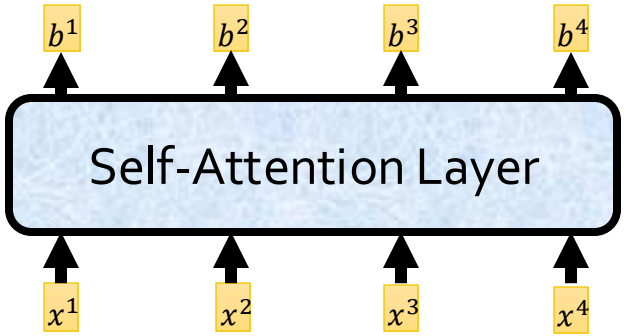
$$Q = \mathbf{W}^q\mathbf{x}$$
$$K = \mathbf{W}^k\mathbf{x}$$
$$V = \mathbf{W}^v\mathbf{x}$$

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d}}\right)V$$



Self-Attention: Back to Big Picture

- Attention is a powerful mechanism to create context-aware representations
- A way to focus on select parts of the input



- Better at maintaining long-distance dependencies in the context.

[Attention Is All You Need, Vaswani et al. 2017]

89

Properties of Self-Attention

Layer Type	Complexity per Layer	Sequential Operations
Self-Attention	$O(n^2 \cdot d)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$

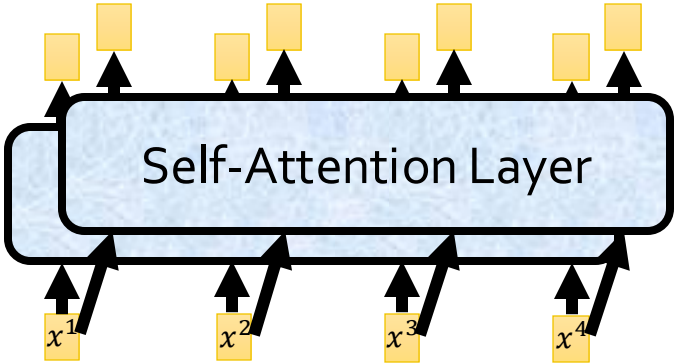
- n = sequence length, d = hidden dimension
- Quadratic complexity, but:
  - O(1) sequential operations (not linear like in RNN)

- Efficient implementations

[Attention Is All You Need, Vaswani et al. 2017]

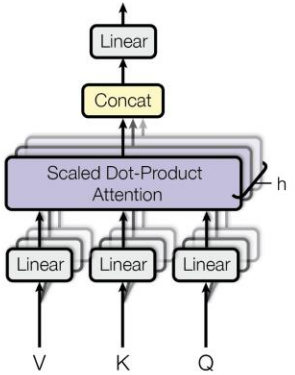
Multi-Headed Self-Attention

- Multiple parallel attention layers is quite common.
  - Each attention layer has its own parameters.
  - Concatenate the results and run them through a linear projection.



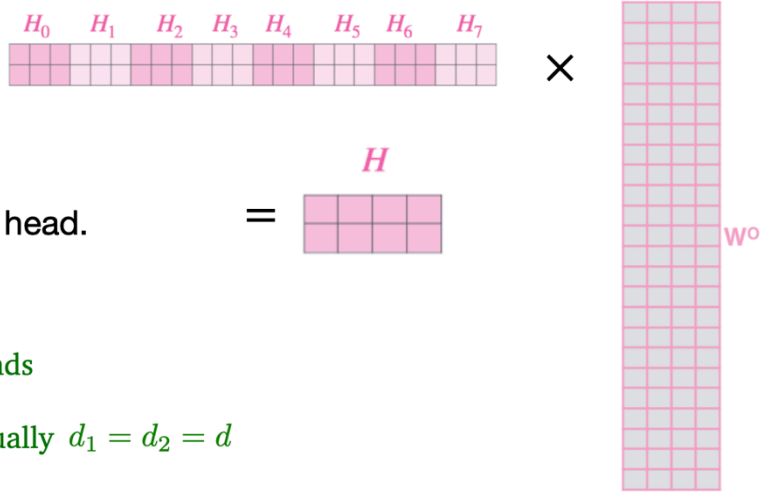
[Attention Is All You Need, Vaswani et al. 2017]

Multi-Headed Self-Attention



- Just concatenate all the heads and apply an output projection matrix.

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,...,\text{head}_\text{h})W^O$$
$$\text{head}_i = \text{Attention}(XW_i^Q, XW_i^K, XW_i^V)$$



- In practice, we use a *reduced* dimension for each head.

$$W_i^Q \in \mathbb{R}^{d_1 \times d_q}, W_i^K \in \mathbb{R}^{d_1 \times d_k}, W_i^V \in \mathbb{R}^{d_1 \times d_v}$$

$$d_q = d_k = d_v = d/m \quad d = \text{hidden size}, m = \#\text{ of heads}$$

$$W^O \in \mathbb{R}^{d \times d_2} \quad \text{If we stack multiple layers, usually } d_1 = d_2 = d$$

- The total computational cost is similar to that of single-head attention with full dimensionality.

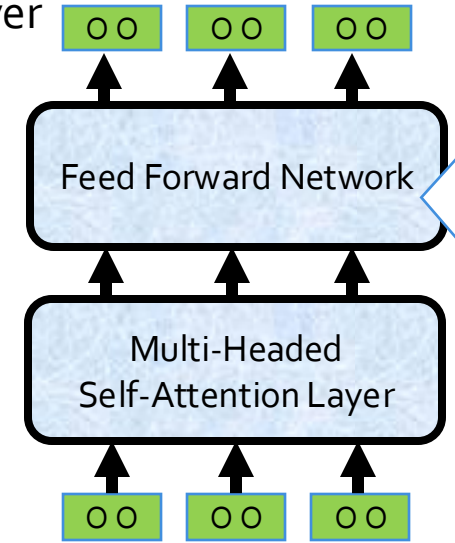
[Slide credit: Danqi Chen]

92

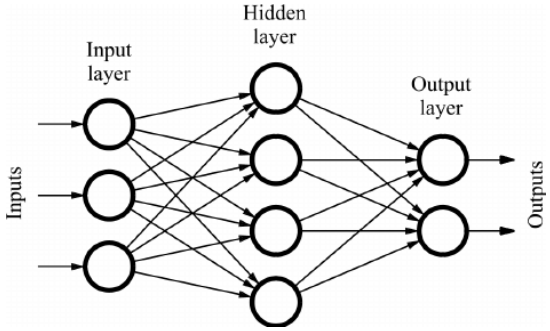
Combine with FFN

- Add a feed-forward network on top it to add more expressivity.
  - This allows the model to apply another transformation to the contextual representations (or "post-process" them).
  - Usually, the dimensionality of the hidden feedforward layer is 2-8 times larger than the input dimension.

$$\text{FFN}(\mathbf{x}) = f(cW_1 + b_1)W_2 + b_2$$



Feedforward Net: Refresher

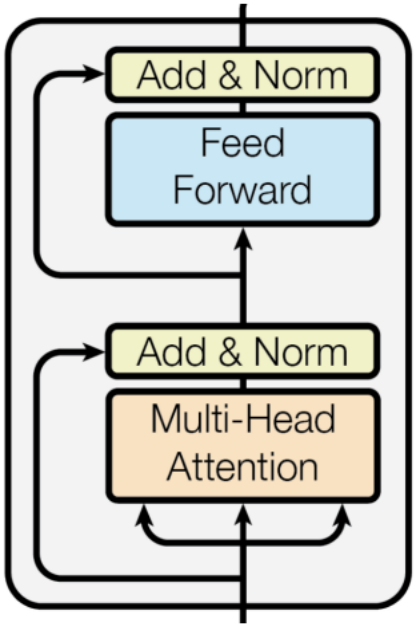


A fully-connected network of nodes and weights.

How Do We Prevent Vanishing Gradients?

- Residual connections let the model "skip" layers
  - These connections are particularly useful for training deep networks

- Use layer normalization to stabilize the network and allow for proper gradient flow



[Attention Is All You Need, Vaswani et al. 2017]

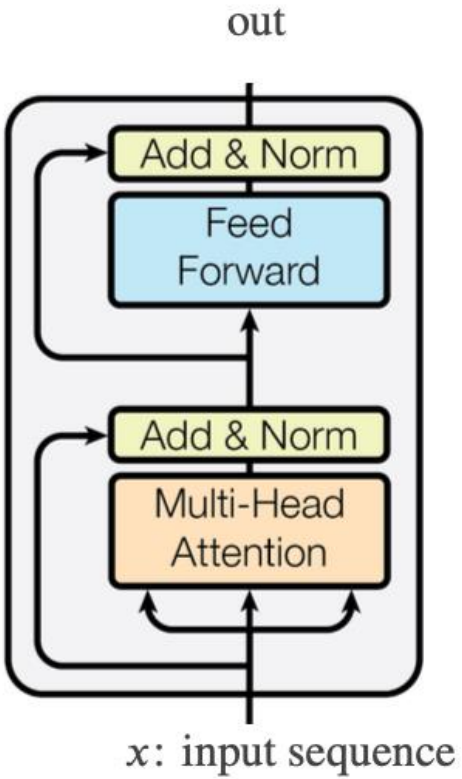
94

Putting it Together: Self-Attention Block

Given input $\mathbf{x}$:

$$\text{out} = LN(\tilde{\boldsymbol{c}} + \boldsymbol{c}')$$
$$\tilde{\boldsymbol{c}} = \text{FFN}(\boldsymbol{c}') = f(\boldsymbol{c}' W_1 + b_1) W_2 + b_2$$

$$\boldsymbol{c}' = LN(\boldsymbol{c} + \boldsymbol{x})$$
$$\boldsymbol{c} = \text{MultiHeadedAttention}(\boldsymbol{x}; \mathbf{W}^q, \mathbf{W}^k, \mathbf{W}^v)$$



[Attention Is All You Need, Vaswani et al. 2017]

Summary: Self-Attention Block

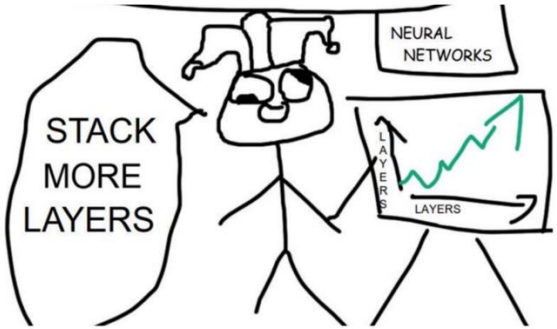
- **Self-Attention:** A critical building block of modern language models.
  - The idea is to compose meanings of words weighted according some similarity notion.

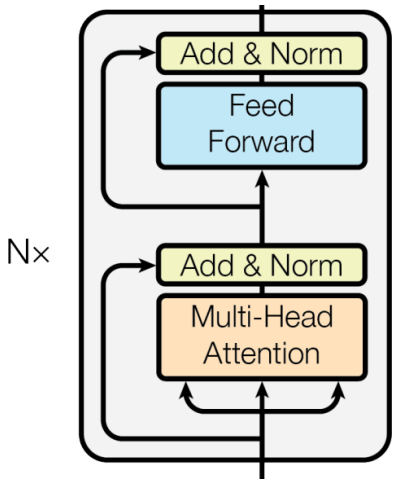
- **Next:** We will combine self-attention blocks to build various architectures known as Transformer.

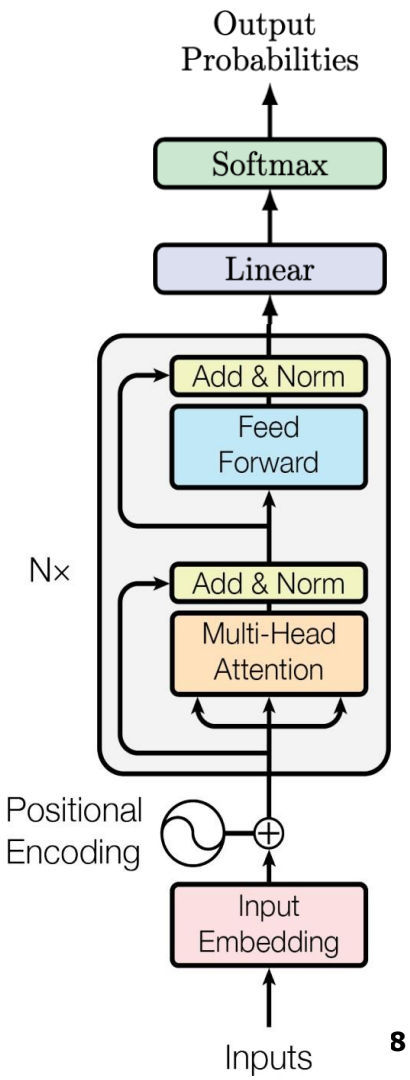
Transformer

How Do We Make it Deep?

- Stack more layers!







[Attention Is All You Need, Vaswani et al. 2017]

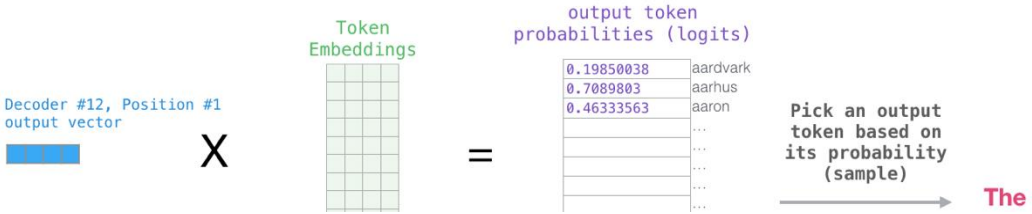
From Representations to Prediction

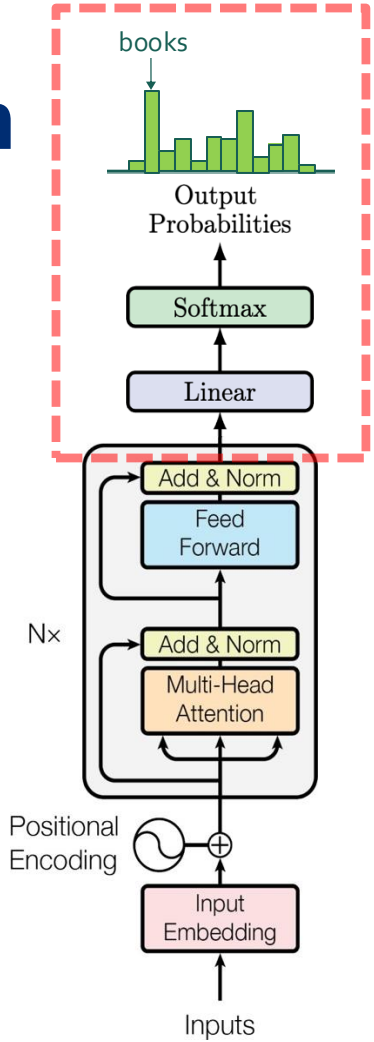
- To perform prediction, add a classification head on top of the final layer of the transformer.
- This can be per token (Language modeling)
- Or can be for the entire sequence (only one token)

$$\text{out} \in \mathbb{R}^{S \times d} \quad \text{(S: Sequence length)}$$

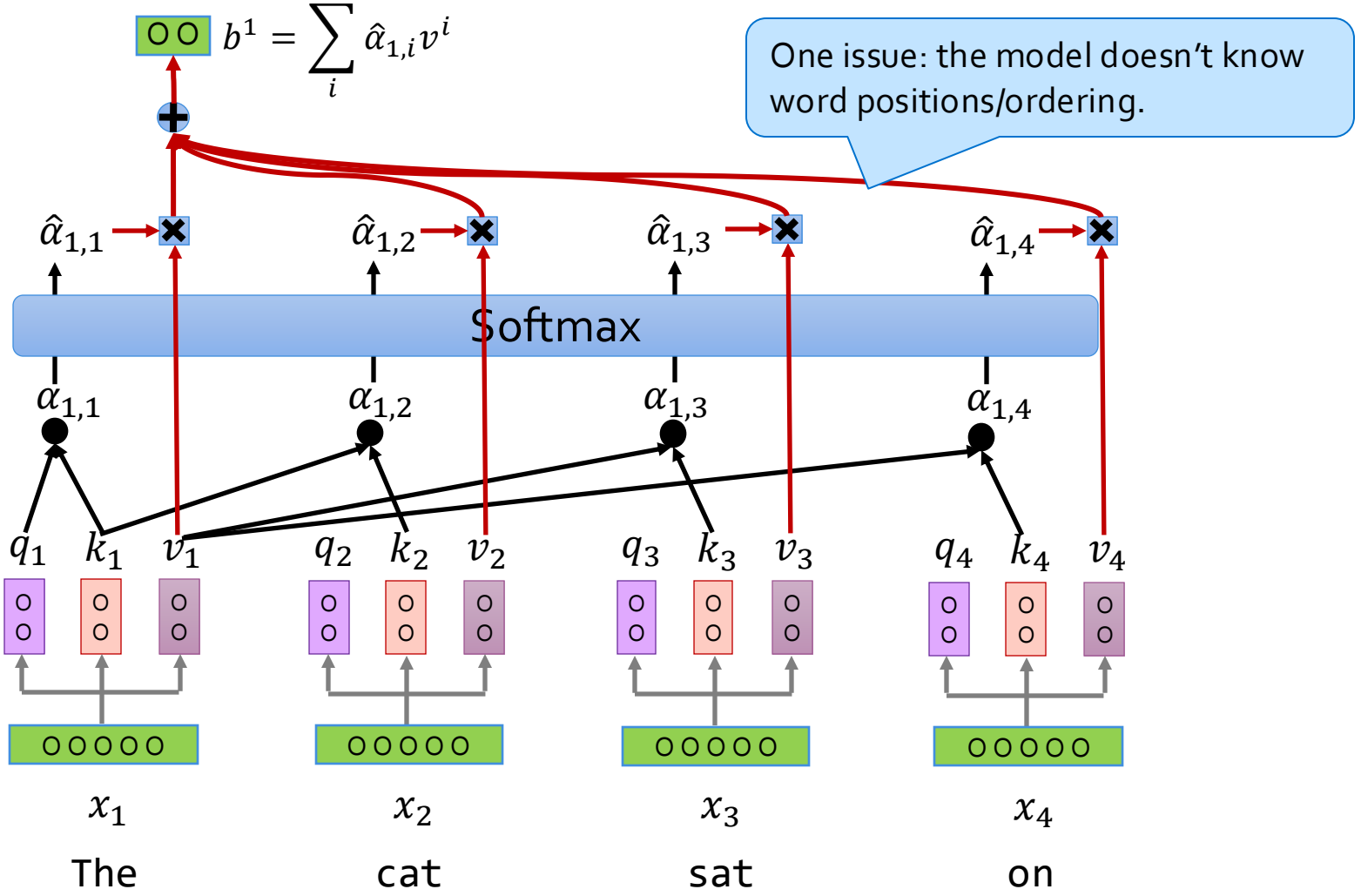
$$\text{logits} = \text{Linear}_{(\text{d, V})}(out) = f\left(out \,.\, W_V\right) \in \mathbb{R}^{S \times V}$$

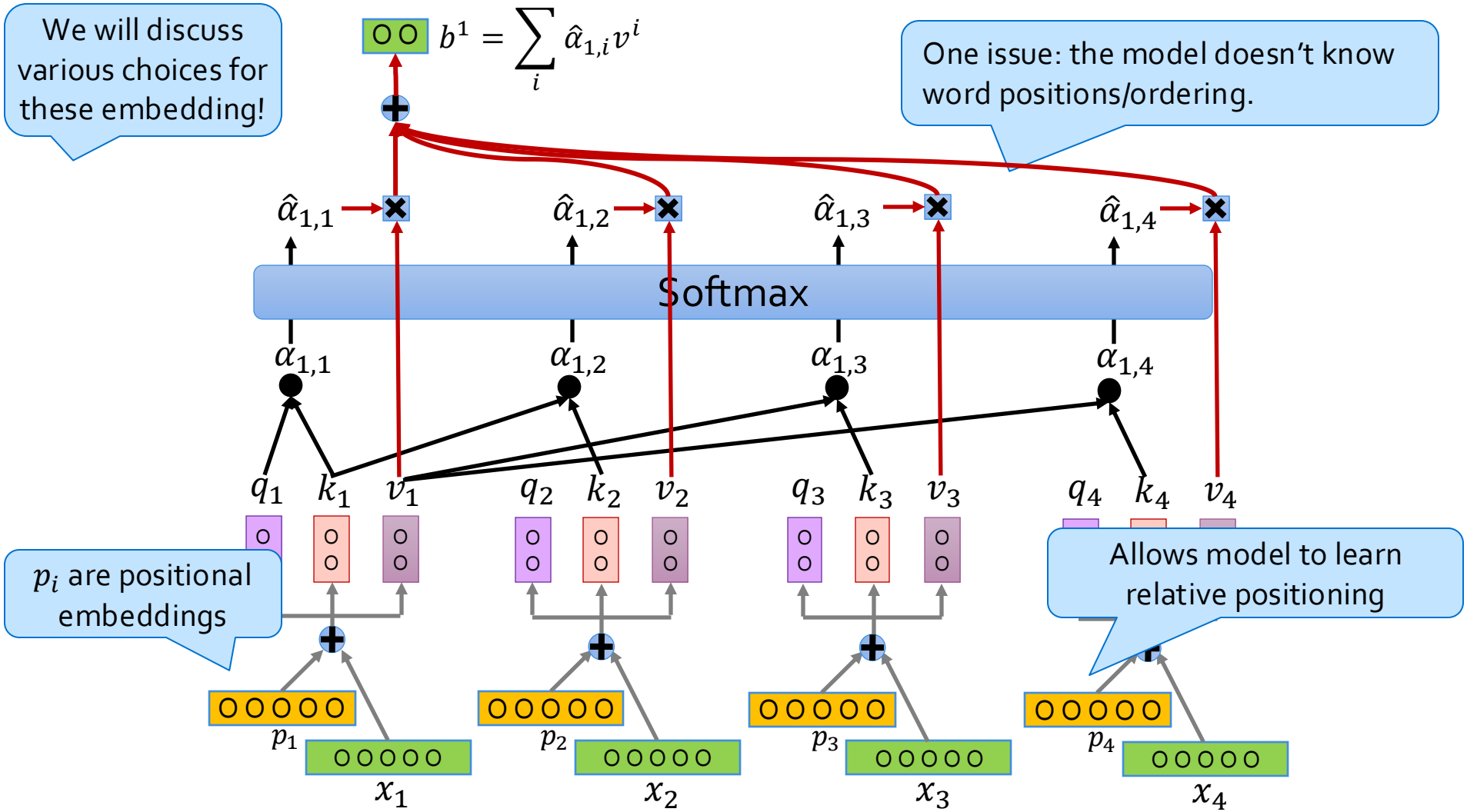
$$\text{probabilies} = \text{softmax}(\text{logits}) \in \mathbb{R}^{S \times V}$$





One last wrinkle though …

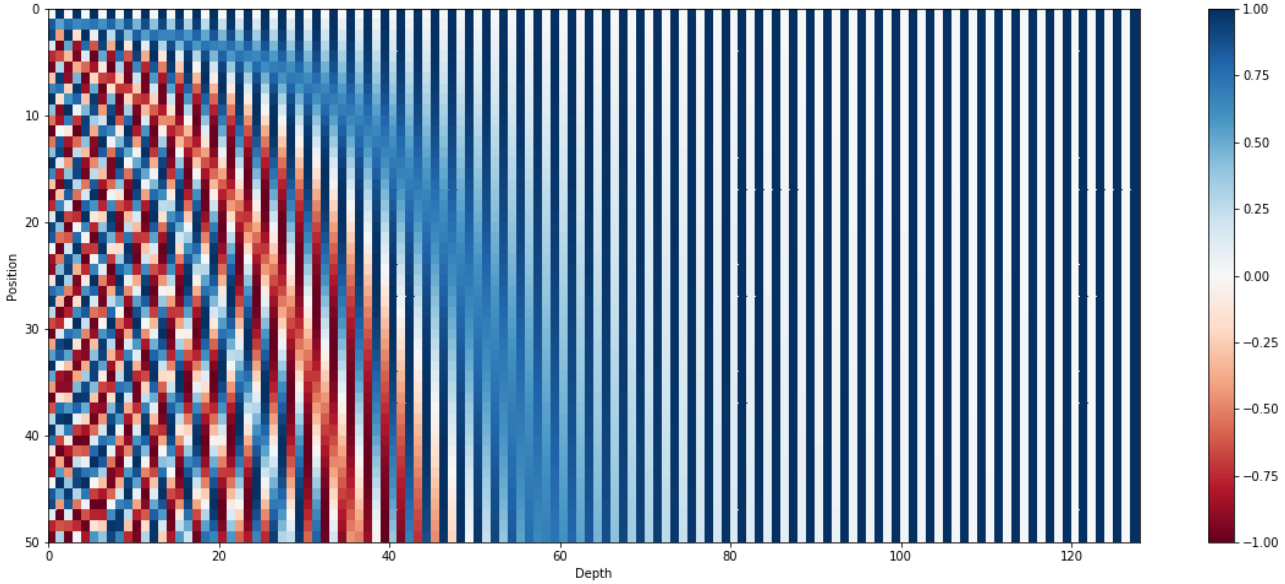

$b^1 = \sum_i \hat{\alpha}_{1,i} v^i$
One issue: the model doesn't know word positions/ordering.
$\hat{\alpha}_{1,1}$
$\hat{\alpha}_{1,2}$
$\hat{\alpha}_{1,3}$
$\hat{\alpha}_{1,4}$
Softmax
$\alpha_{1,1}$
$\alpha_{1,2}$
$\alpha_{1,3}$
$\alpha_{1,4}$
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
x_1 x_2 x_3 x_4
The cat sat on

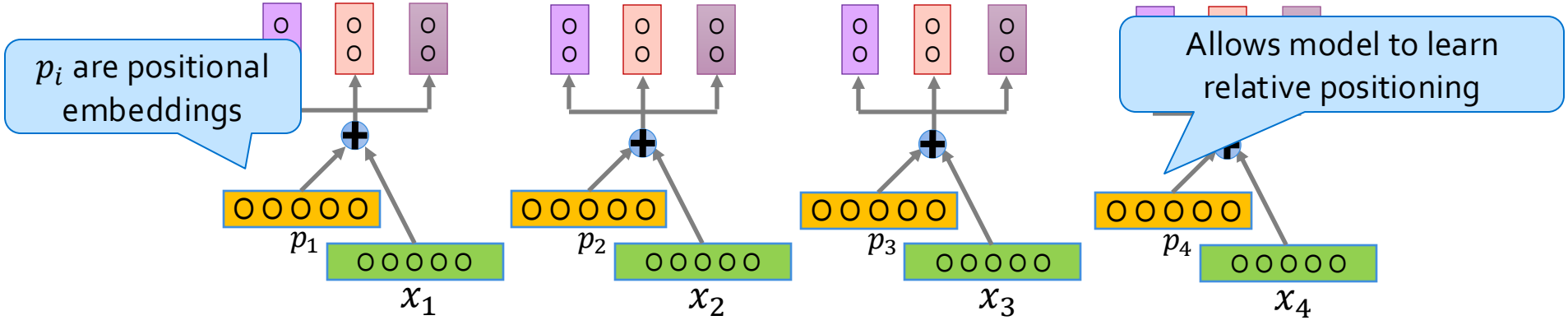

We will discuss various choices for these embedding!
$b^1 = \sum_i \hat{\alpha}_{1,i} v^i$
One issue: the model doesn't know word positions/ordering.
$\hat{\alpha}_{1,1}$
$\hat{\alpha}_{1,2}$
$\hat{\alpha}_{1,3}$
$\hat{\alpha}_{1,4}$
Softmax
$\alpha_{1,1}$
$\alpha_{1,2}$
$\alpha_{1,3}$
$\alpha_{1,4}$
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
p_i are positional embeddings
Allows model to learn relative positioning
p_1
p_2
p_3
p_4
x_1
x_2
x_3
x_4

An approach:

Sine/Cosine encoding

$$p_i = \begin{pmatrix} \sin(i/10000^{2*1/d}) \\ \cos(i/10000^{2*1/d}) \\ \vdots \\ \sin(i/10000^{2*\frac{d}{2}/d}) \\ \cos(i/10000^{2*\frac{d}{2}/d}) \end{pmatrix}$$

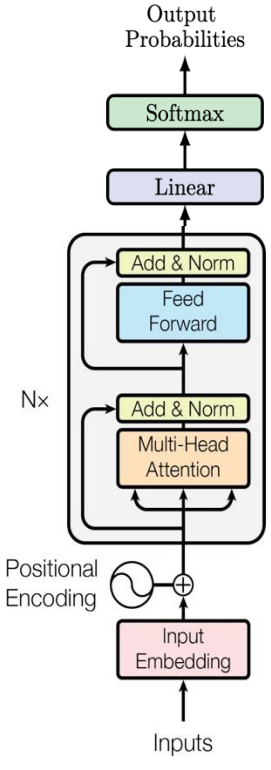




The Transformer Stack in PyTorch

```python
class Block(nn.Module):
    def __init__(self, config):
        super().__init__()
        self.ln_1 = LayerNorm(config.n_embd, bias=config.bias)
        self.attn = CausalSelfAttention(config)
        self.ln_2 = LayerNorm(config.n_embd, bias=config.bias)
        self.mlp = MLP(config)

    def forward(self, x):
        x = x + self.attn(self.ln_1(x))
        x = x + self.mlp(self.ln_2(x))
        return x
```

```python
self.transformer = nn.ModuleDict(
    dict(
        wte=nn.Embedding(config.vocab_size, config.n_embd),
        wpe=nn.Embedding(config.block_size, config.n_embd),
        drop=nn.Dropout(config.dropout),
        h=nn.ModuleList([Block(config) for _ in range(config.n_layer)]),
        ln_f=LayerNorm(config.n_embd, bias=config.bias),
    )
)
self.lm_head = nn.Linear(config.n_embd, config.vocab_size, bias=False)
```



Transformer-based Language Modeling

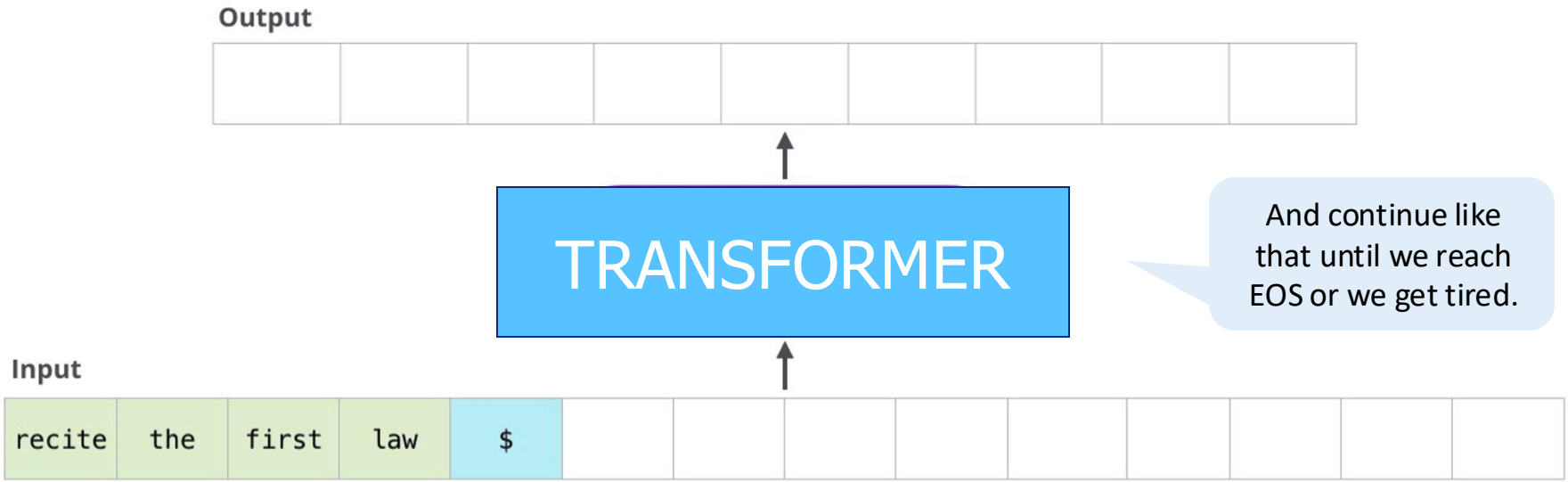
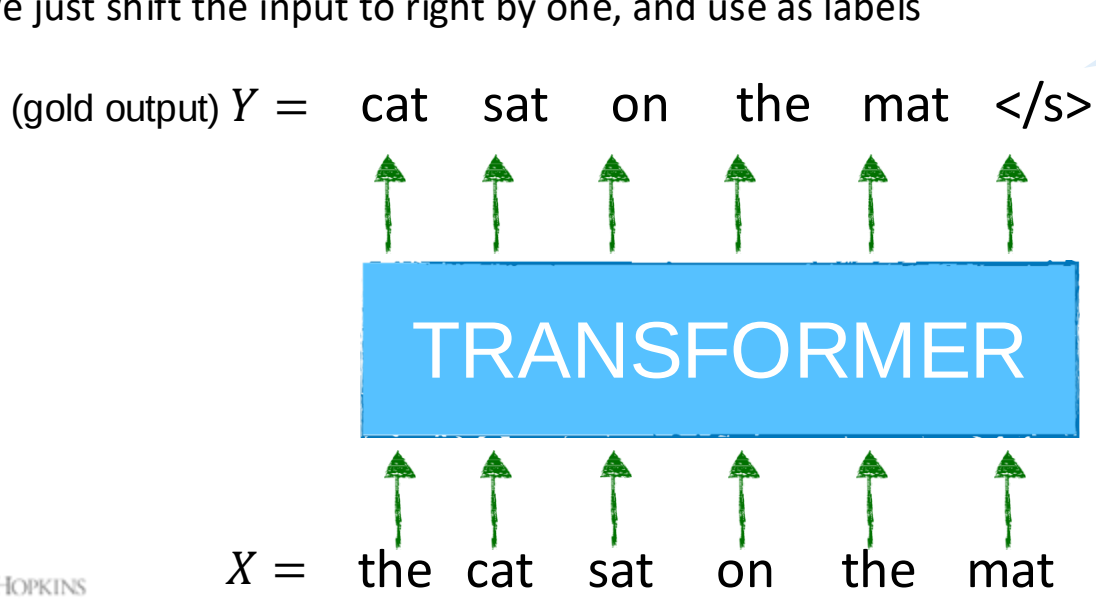


Image by http://jalammar.github.io/illustrated-gpt2/

Training a Transformer Language Model

- **Goal:** Train a Transformer for language modeling (i.e., predicting the next word).
- **Approach:** Train it so that each position is predictor of the next (right) token.
  - We just shift the input to right by one, and use as labels



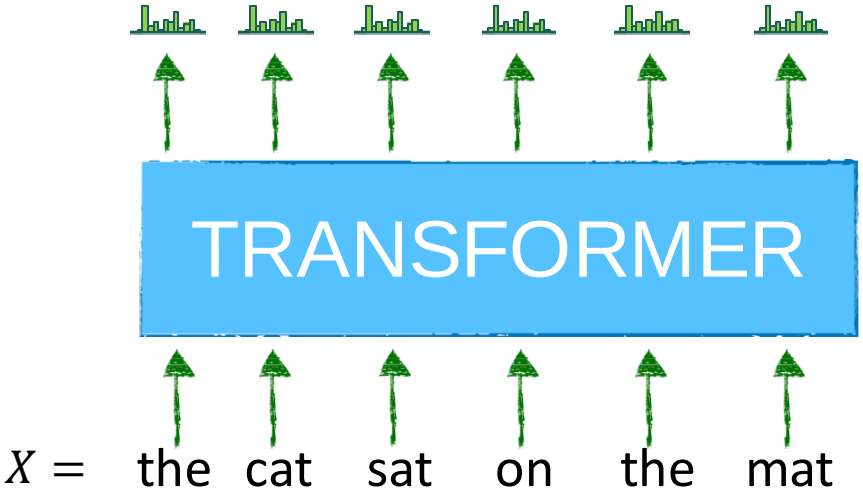

```
X = text[:, :-1]
Y = text[:, 1:]
```

[Slide credit: Arman Cohan]

Training a Transformer Language Model

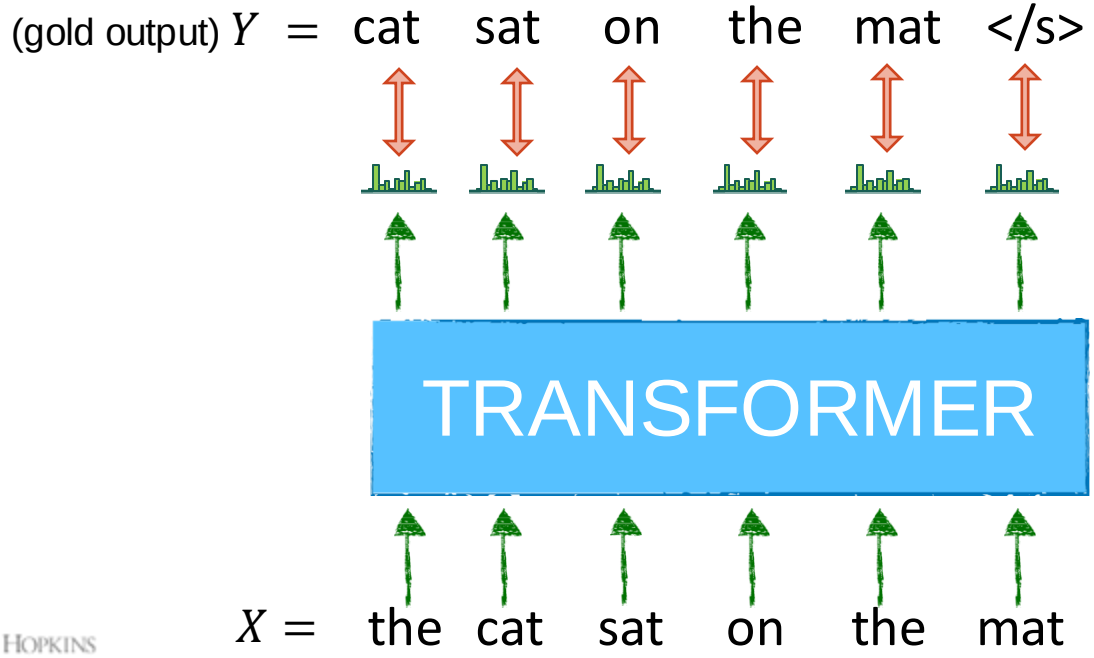
- For each position, compute their corresponding **distribution** over the whole vocab.

(gold output) Y = cat sat on the mat </s>



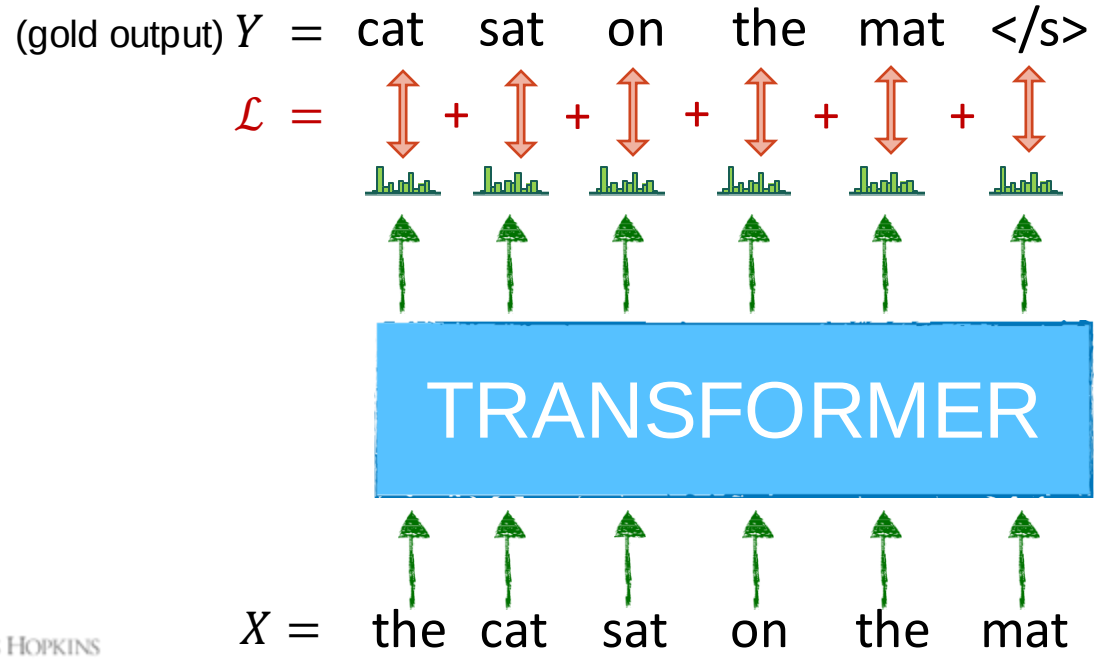
Training a Transformer Language Model

- For each position, compute the **loss** between the distribution and the gold output label.



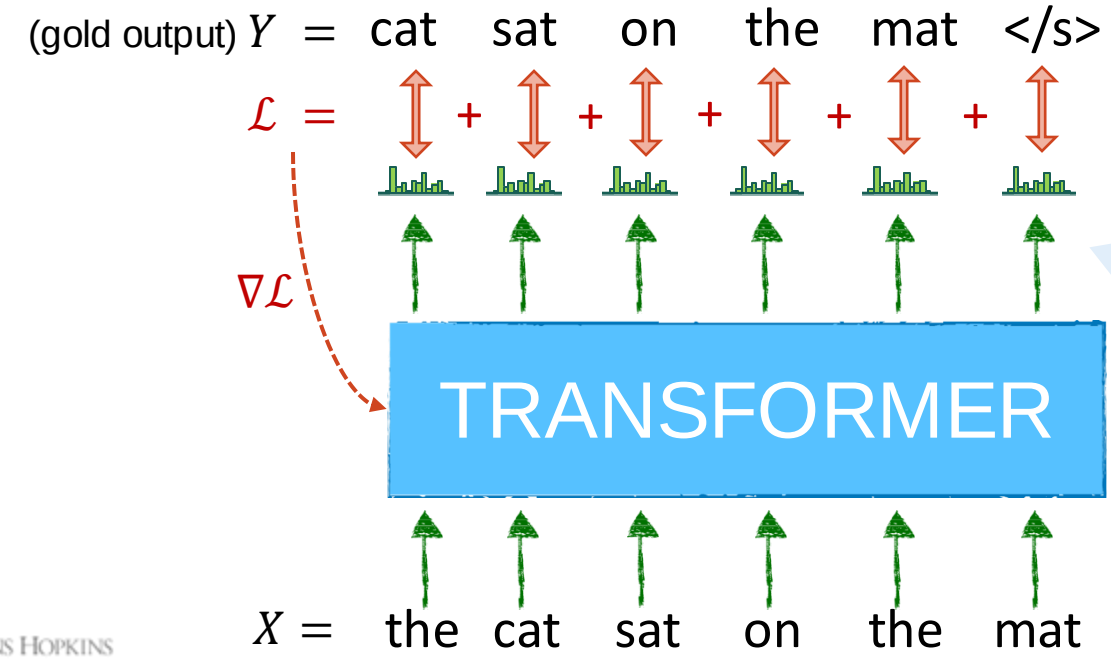
Training a Transformer Language Model

- Sum the position-wise loss values to a obtain a **global loss**.



Training a Transformer Language Model

- Using this loss, do **Backprop** and **update** the Transformer parameters.



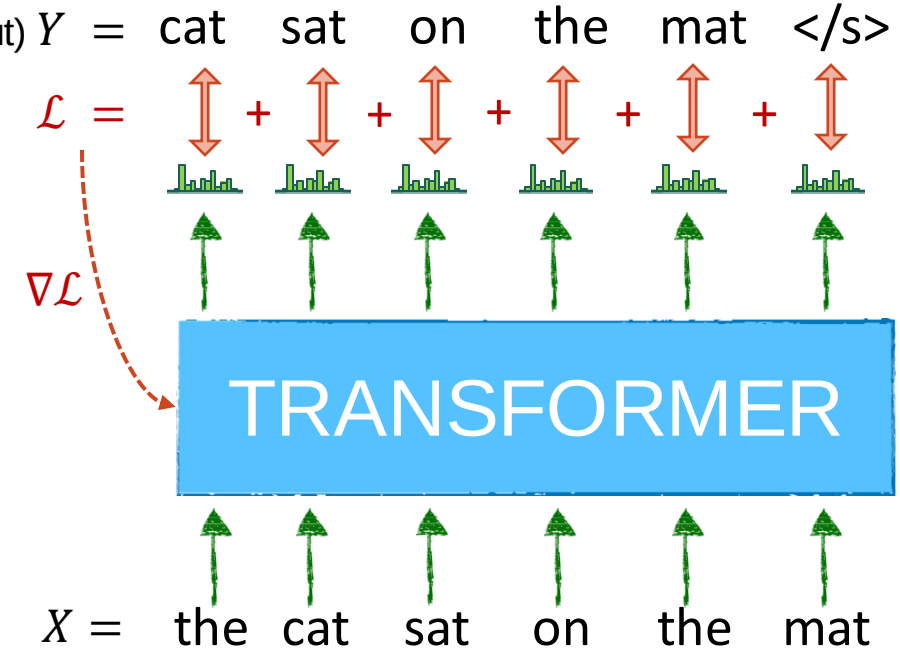
Well, this is not quite right 🤡

…

what is the problem with this?

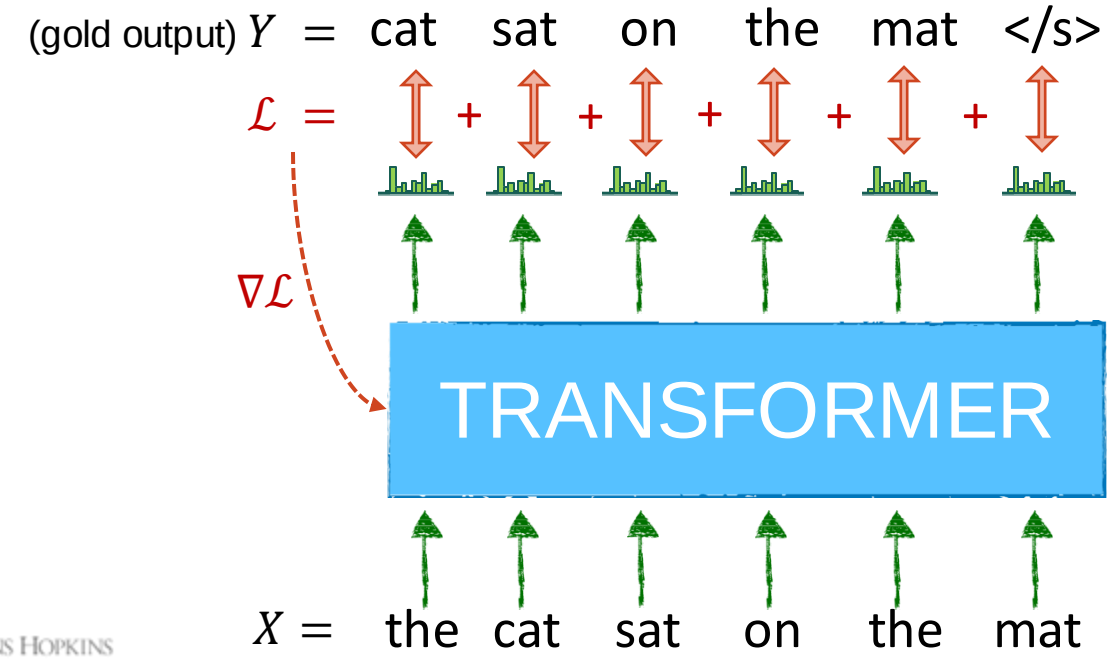
Training a Transformer Language Model

- The model would solve the task by copying the next token to output (data leakage).
  - Does not learn anything useful



Training a Transformer Language Model

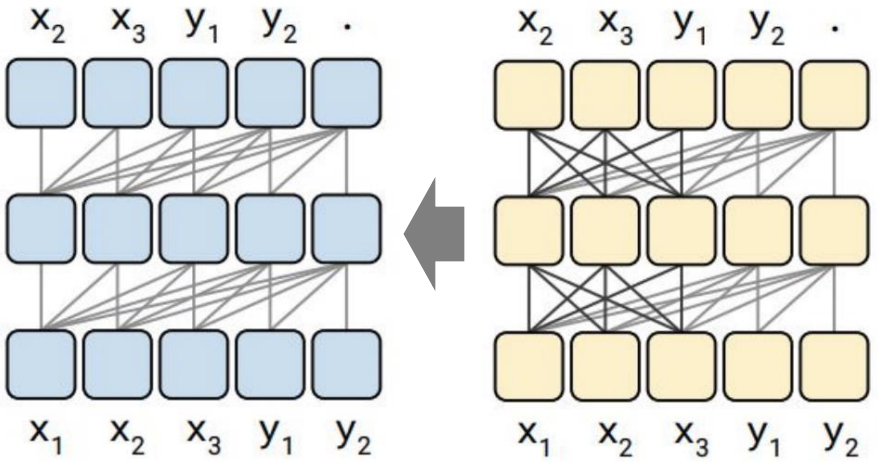
- We need to **prevent information leakage** from future tokens! How?



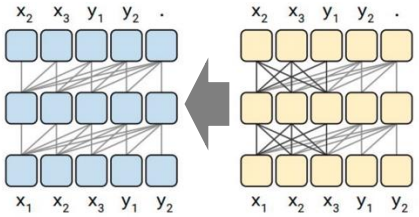
Attention mask

Attention raw scores

	1	2	3	4	5	6	7	8	9	10	11	12
0	-0.08	1.24	0.69	-0.98	1.43	-0.6	0.7	0.16	0.93	1.28	-1.61	-1.1
1	-0.09	-0.0	-0.7	0.06	0.25	0.23	0.26	0.18	0.78	-0.21	-1.01	1.01
2	0.86	1.19	1.59	0.86	-0.13	-0.15	-2.13	-0.98	-0.87	-1.72	1.87	-0.72
3	0.12	-0.03	-0.02	0.88	-0.46	-0.7	0.54	-0.42	-1.89	-0.38	0.04	-0.84
4	0.51	0.17	0.13	-1.64	0.24	-0.02	1.68	-0.36	0.64	0.36	0.27	0.66
5	0.24	-1.44	0.43	0.74	0.96	-1.21	-0.31	1.54	1.66	1.14	0.58	-1.44
6	0.26	-0.1	0.93	0.72	-0.38	1.65	0.47	-0.96	-0.17	-0.9	-1.57	0.22
7	-0.55	0.81	0.71	1.7	-0.8	-1.14	-0.32	1.78	-0.7	-0.04	1.54	0.81
8	0.74	-0.76	-0.44	-0.08	-1.38	-0.13	1.25	-1.37	1.84	0.3	0.57	0.74
9	-0.97	-0.91	0.15	0.35	-0.81	0.11	1.14	-1.52	1.06	1.87	0.5	-0.3
10	1.56	0.9	0.39	1.46	1.44	-1.05	0.9	-0.73	0.36	-0.67	-0.62	-0.43
11	0.32	0.74	0.44	-0.1	1.19	0.83	0.29	2.06	0.51	-0.26	1.51	0.11



Attention mask



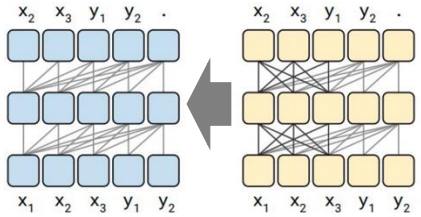
Attention raw scores

	1	2	3	4	5	6	7	8	9	10	11	12
0	-0.08	1.24	0.69	-0.98	1.43	-0.6	0.7	0.16	0.93	1.28	-1.61	-1.1
1	-0.09	-0.0	-0.7	0.06	0.25	0.23	0.26	0.18	0.78	-0.21	-1.01	1.01
2	0.86	1.19	1.59	0.86	-0.13	-0.15	-2.13	-0.98	-0.87	-1.72	1.87	-0.72
3	0.12	-0.03	-0.02	0.88	-0.46	-0.7	0.54	-0.42	-1.89	-0.38	0.04	-0.84
4	0.51	0.17	0.13	-1.64	0.24	-0.02	1.68	-0.36	0.64	0.36	0.27	0.66
5	0.24	-1.44	0.43	0.74	0.96	-1.21	-0.31	1.54	1.66	1.14	0.58	-1.44
6	0.26	-0.1	0.93	0.72	-0.38	1.65	0.47	-0.96	-0.17	-0.9	-1.57	0.22
7	-0.55	0.81	0.71	1.7	-0.8	-1.14	-0.32	1.78	-0.7	-0.04	1.54	0.81
8	0.74	-0.76	-0.44	-0.08	-1.38	-0.13	1.25	-1.37	1.84	0.3	0.57	0.74
9	-0.97	-0.91	0.15	0.35	-0.81	0.11	1.14	-1.52	1.06	1.87	0.5	-0.3
10	1.56	0.9	0.39	1.46	1.44	-1.05	0.9	-0.73	0.36	-0.67	-0.62	-0.43
11	0.32	0.74	0.44	-0.1	1.19	0.83	0.29	2.06	0.51	-0.26	1.51	0.11

Attention mask

	0	1	2	3	4	5	6	7	8	9	10	11
0	1.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
1	1.0	1.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
2	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
3	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
4	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf
5	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf	-inf
6	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf
7	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf
8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf
9	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf
10	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf
11	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Attention mask



Attention raw scores

	1	2	3	4	5	6	7	8	9	10	11
0	-0.08	1.24	0.69	-0.98	1.43	-0.6	0.7	0.16	0.93	1.28	-1.61	-1.1
1	-0.09	-0.0	-0.7	0.06	0.25	0.23	0.26	0.18	0.78	-0.21	-1.01	1.01
2	0.86	1.19	1.59	0.86	-0.13	-0.15	-2.13	-0.98	-0.87	-1.72	1.87	-0.72
3	0.12	-0.03	-0.02	0.88	-0.46	-0.7	0.54	-0.42	-1.89	-0.38	0.04	-0.84
4	0.51	0.17	0.13	-1.64	0.24	-0.02	1.68	-0.36	0.64	0.36	0.27	0.66
5	0.24	-1.44	0.43	0.74	0.96	-1.21	-0.31	1.54	1.66	1.14	0.58	-1.44
6	0.26	-0.1	0.93	0.72	-0.38	1.65	0.47	-0.96	-0.17	-0.9	-1.57	0.22
7	-0.55	0.81	0.71	1.7	-0.8	-1.14	-0.32	1.78	-0.7	-0.04	1.54	0.81
8	0.74	-0.76	-0.44	-0.08	-1.38	-0.13	1.25	-1.37	1.84	0.3	0.57	0.74
9	-0.97	-0.91	0.15	0.35	-0.81	0.11	1.14	-1.52	1.06	1.87	0.5	-0.3
10	1.56	0.9	0.39	1.46	1.44	-1.05	0.9	-0.73	0.36	-0.67	-0.62	-0.43
11	0.32	0.74	0.44	-0.1	1.19	0.83	0.29	2.06	0.51	-0.26	1.51	0.11

X

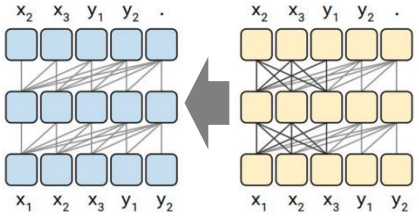
Attention mask

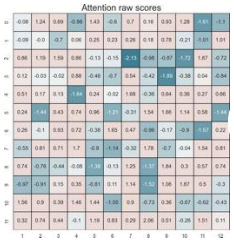
	0	1	2	3	4	5	6	7	8	9	10	11
0	1.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
1	1.0	1.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
2	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
3	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
4	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf
5	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf	-inf
6	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf
7	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf
8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf
9	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf
10	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf
11	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Note matrix multiplication is quite fast in GPUs.

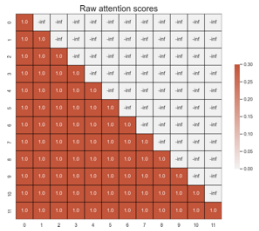
Slide credit: Arman Cohan

Attention mask





X



=

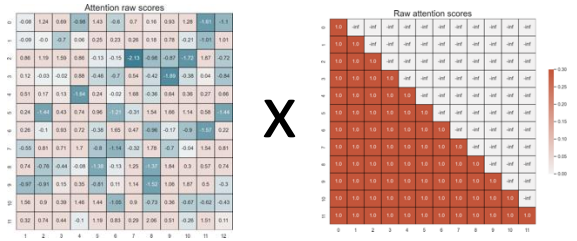
Masked attention raw scores

	1	2	3	4	5	6	7	8	9	10	11	12
0	-0.08	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
1	-0.09	-0.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
2	0.86	1.19	1.59	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
3	0.12	-0.03	-0.02	0.88	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
4	0.51	0.17	0.13	-1.64	0.24	-inf	-inf	-inf	-inf	-inf	-inf	-inf
5	0.24	-1.44	0.43	0.74	0.96	-1.21	-inf	-inf	-inf	-inf	-inf	-inf
6	0.26	-0.1	0.93	0.72	-0.38	1.65	0.47	-inf	-inf	-inf	-inf	-inf
7	-0.55	0.81	0.71	1.7	-0.8	-1.14	-0.32	1.78	-inf	-inf	-inf	-inf
8	0.74	-0.76	-0.44	-0.08	-1.38	-0.13	1.25	-1.37	1.84	-inf	-inf	-inf
9	-0.97	-0.91	0.15	0.35	-0.81	0.11	1.14	-1.52	1.06	1.87	-inf	-inf
10	1.56	0.9	0.39	1.46	1.44	-1.05	0.9	-0.73	0.36	-0.67	-0.62	-inf
11	0.32	0.74	0.44	-0.1	1.19	0.83	0.29	2.06	0.51	-0.26	1.51	0.11

Slide credit: Arman Cohan

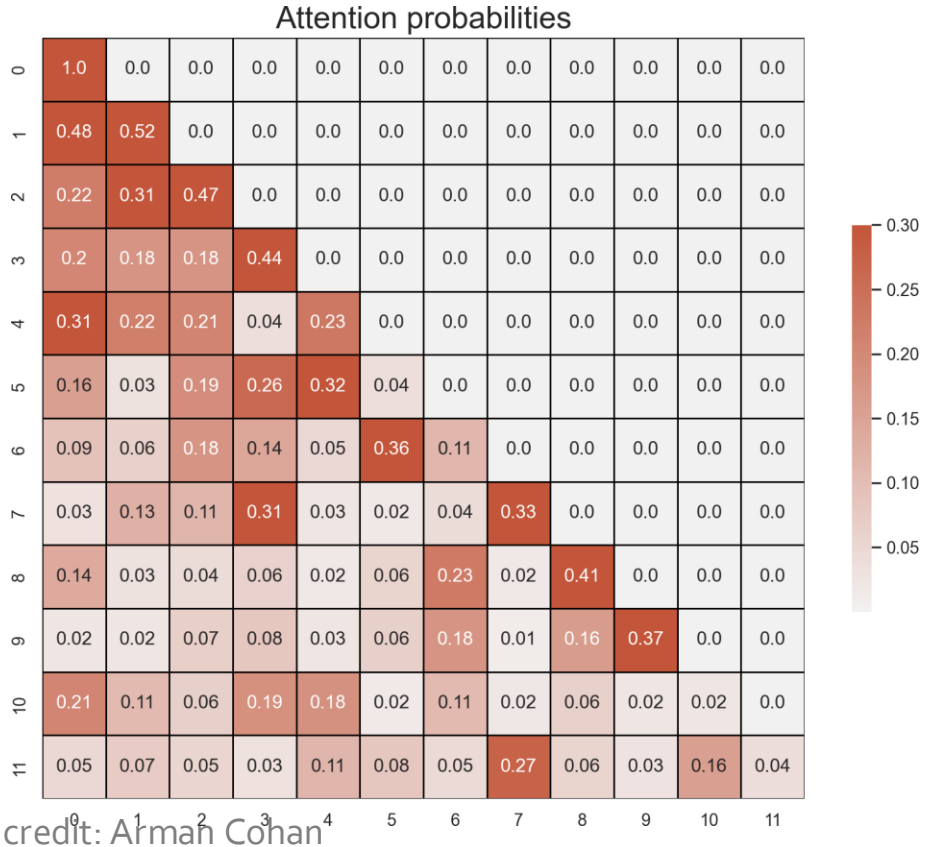
Attention mask

The effect is more than just pruning out some of the wirings in self-attention block.



X

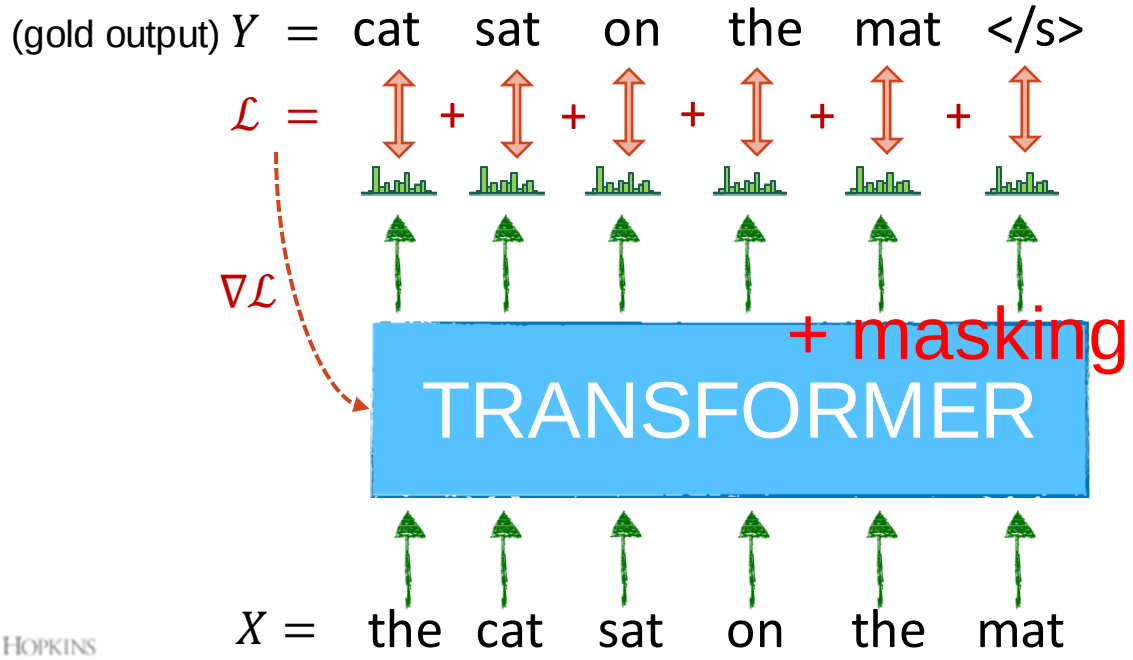
softmax



Slide credit: Arman Cohan

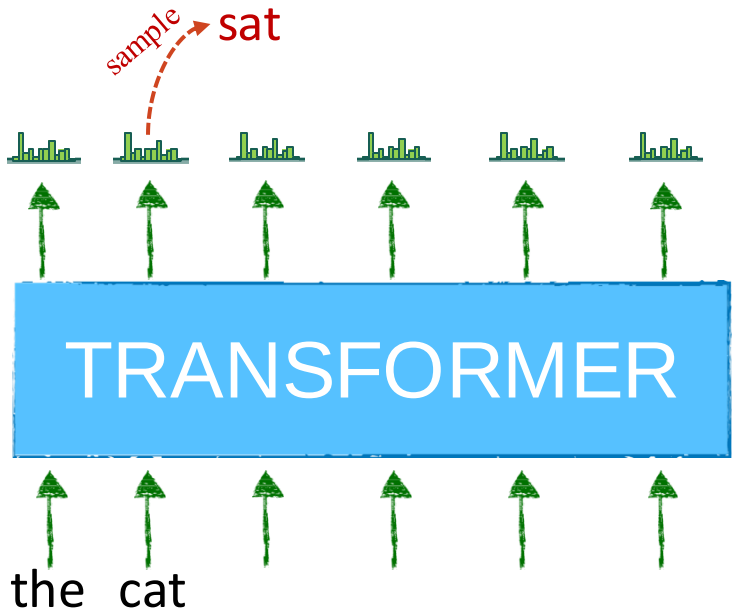
Training a Transformer Language Model

- We need to **prevent information leakage** from future tokens! How?



How to use the model to generate text?

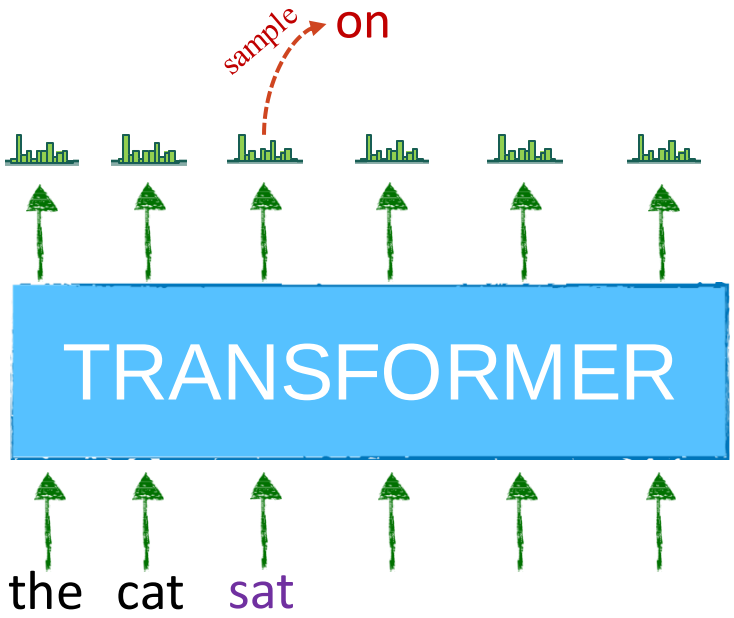
- Use the output of previous step as input to the next step repeatedly



How to use the model to generate text?

- Use the output of previous step as input to the next step repeatedly

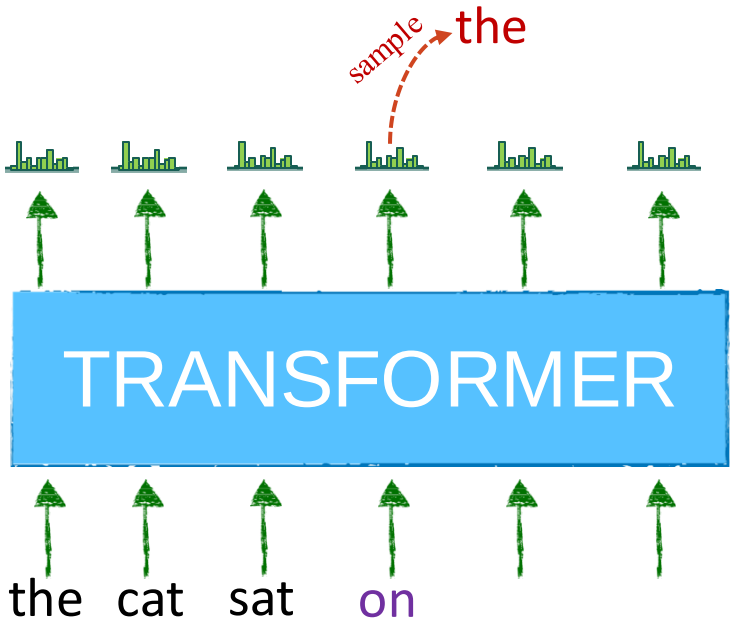
The probabilities get revised upon adding a new token to the input.



How to use the model to generate text?

- Use the output of previous step as input to the next step repeatedly

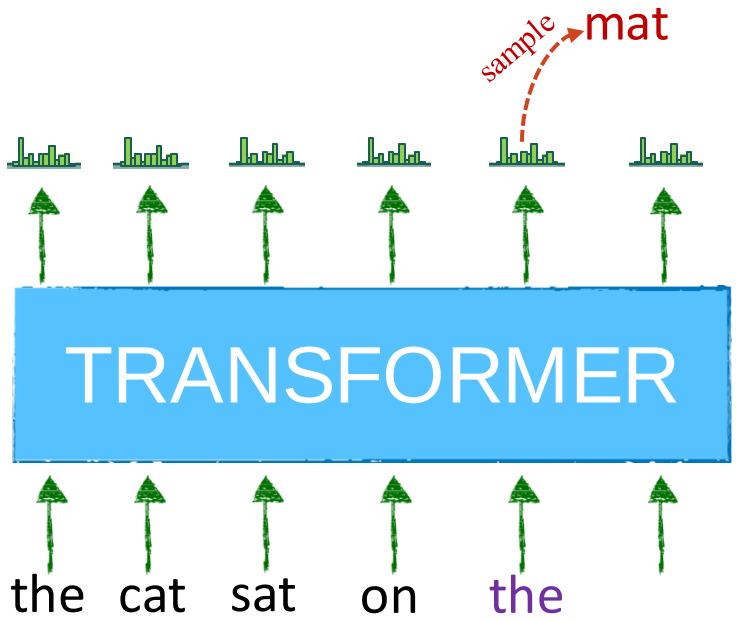
The probabilities get revised upon adding a new token to the input.



How to use the model to generate text?

- Use the output of previous step as input to the next step repeatedly

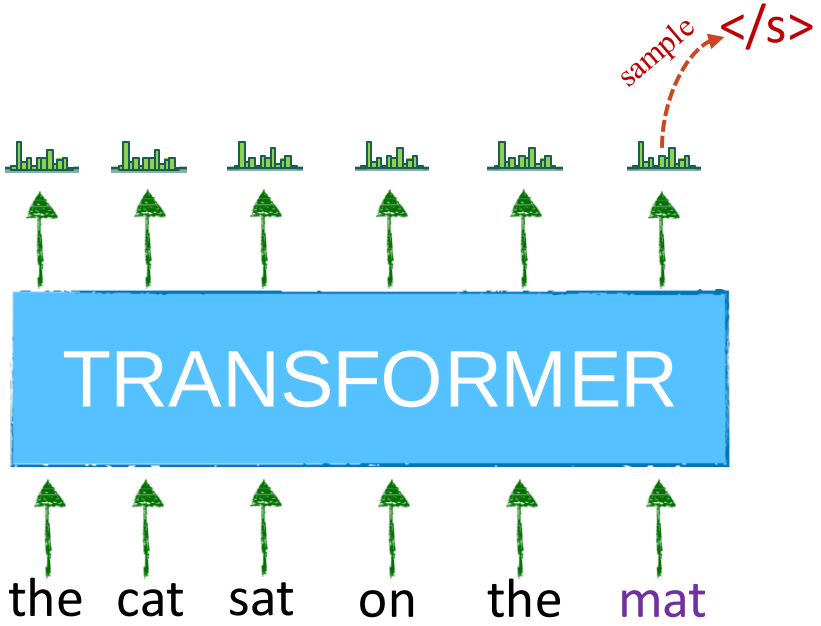
The probabilities get revised upon adding a new token to the input.



How to use the model to generate text?

- Use the output of previous step as input to the next step repeatedly

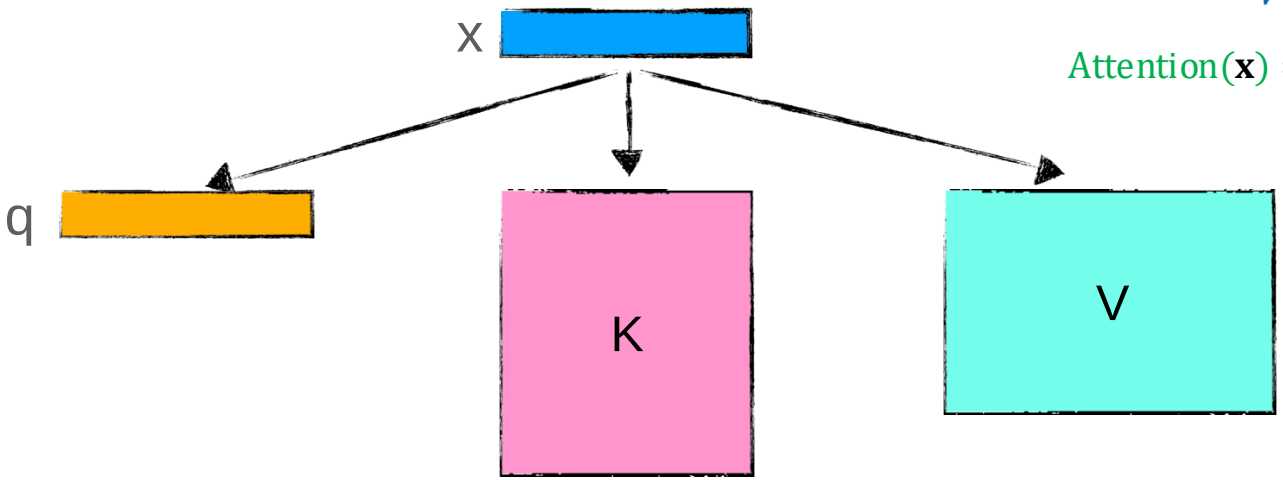
The probabilities get revised upon adding a new token to the input.



An important efficiency consideration about decoding!

Making decoding more efficient

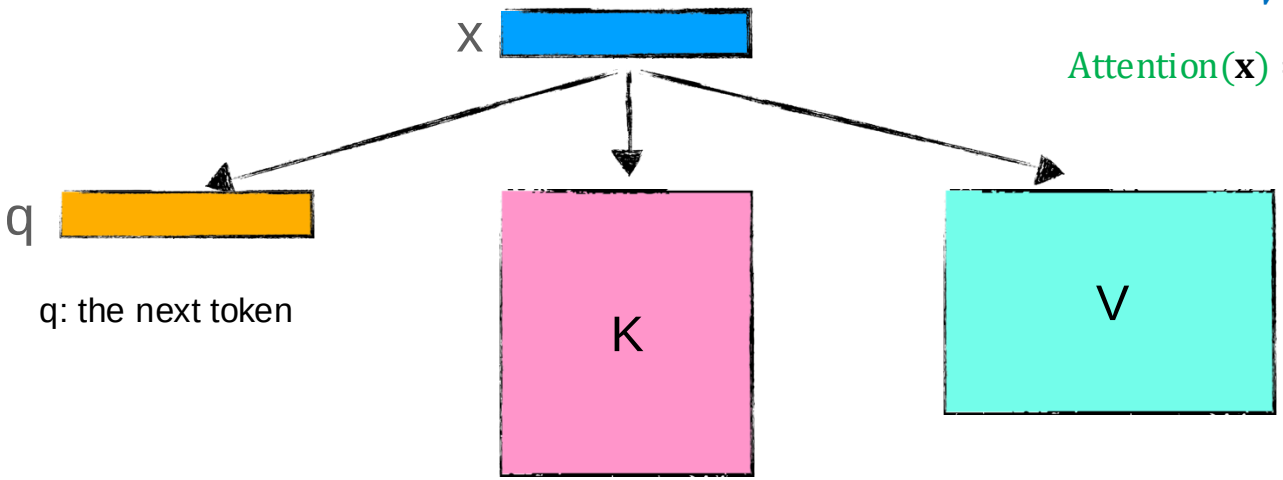
$$Q = \mathbf{W}^q \mathbf{x}$$
$$K = \mathbf{W}^k \mathbf{x}$$
$$V = \mathbf{W}^v \mathbf{x}$$
$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d}}\right)V$$



[Slide credit: Arman Cohan]

Making decoding more efficient

$$Q = \mathbf{W}^q \mathbf{x}$$
$$K = \mathbf{W}^k \mathbf{x}$$
$$V = \mathbf{W}^v \mathbf{x}$$
$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d}}\right)V$$



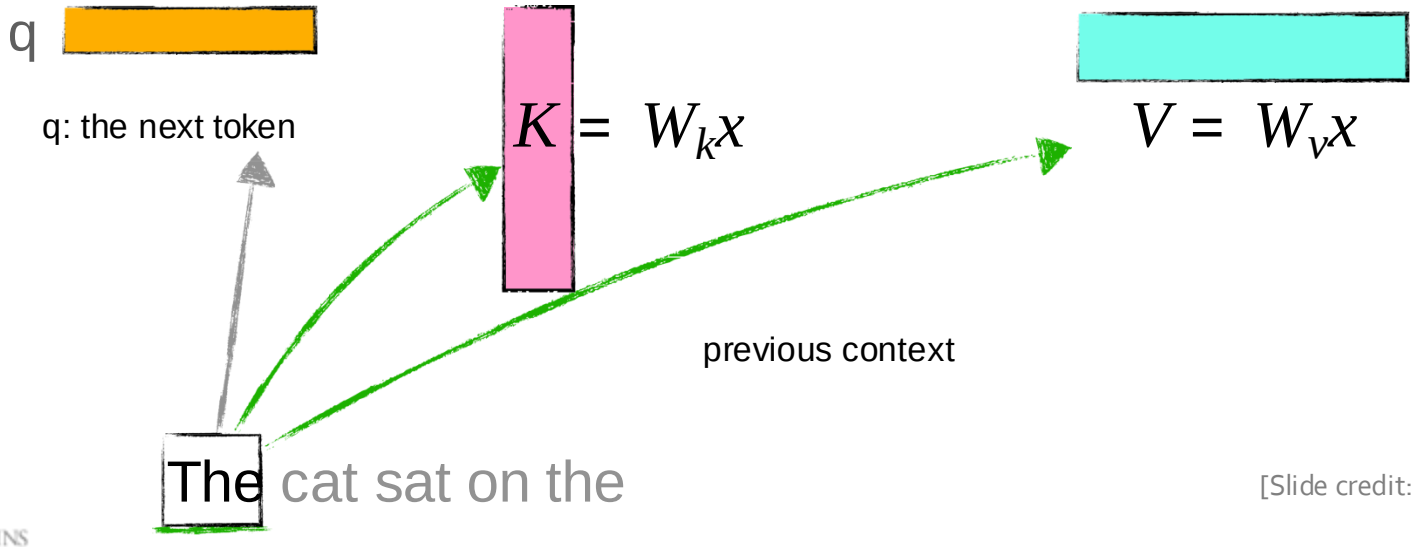
q: the next token

previous context

[Slide credit: Arman Cohan]

Making decoding more efficient

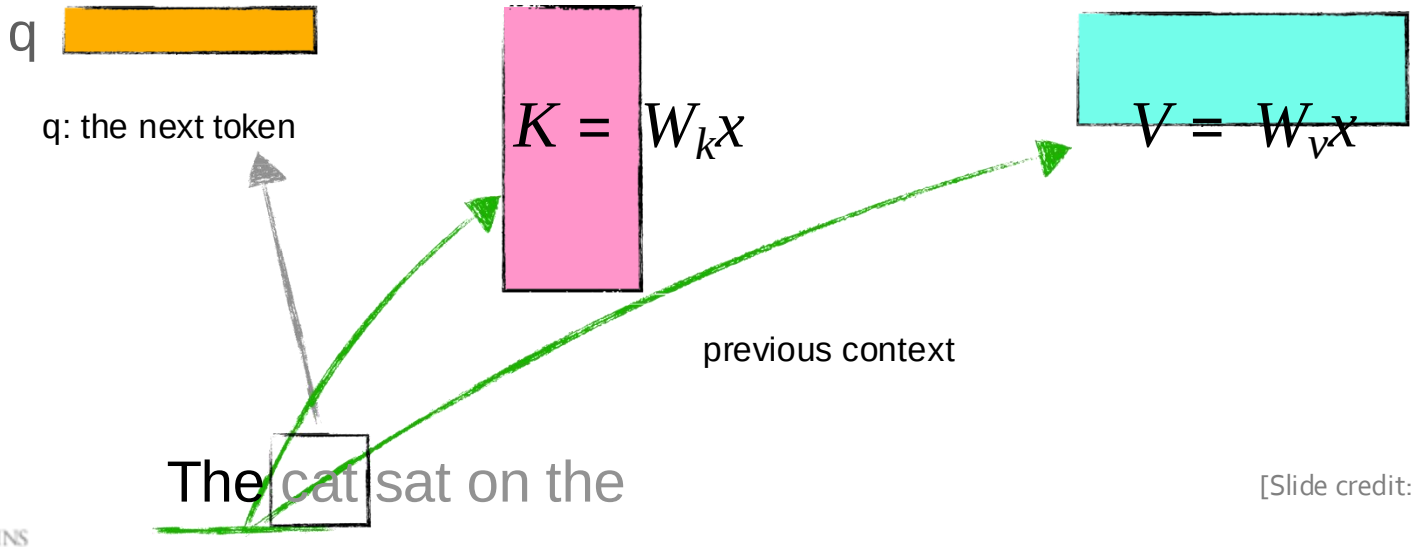
$$Q = \mathbf{W}^q \mathbf{x}$$
$$K = \mathbf{W}^k \mathbf{x}$$
$$V = \mathbf{W}^v \mathbf{x}$$
$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d}}\right)V$$



[Slide credit: Arman Cohan]

Making decoding more efficient

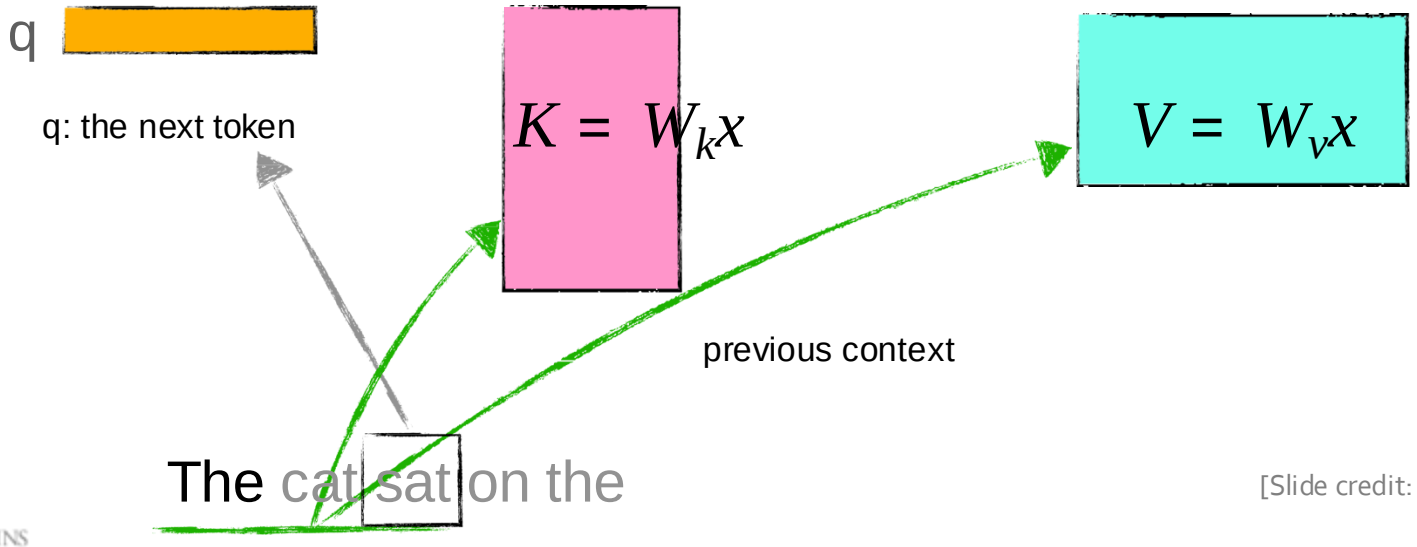
$$Q = \mathbf{W}^q \mathbf{x}$$
$$K = \mathbf{W}^k \mathbf{x}$$
$$V = \mathbf{W}^v \mathbf{x}$$
$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d}}\right)V$$



[Slide credit: Arman Cohan]

Making decoding more efficient

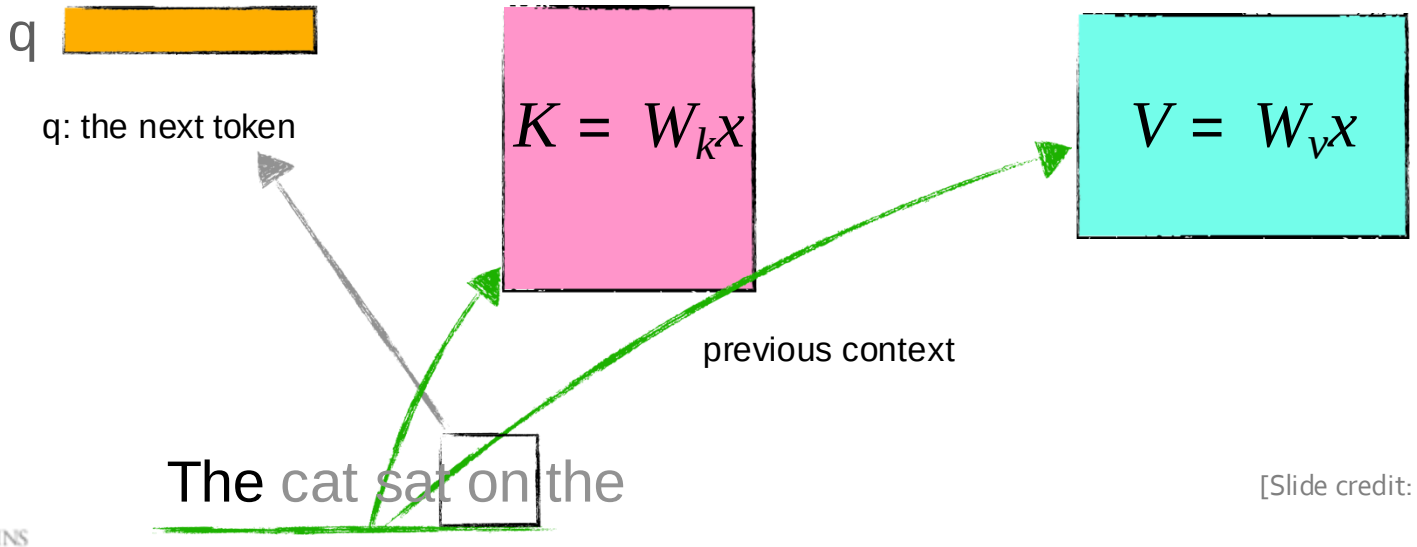
$$Q = \mathbf{W}^q \mathbf{x}$$
$$K = \mathbf{W}^k \mathbf{x}$$
$$V = \mathbf{W}^v \mathbf{x}$$
$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d}}\right)V$$



[Slide credit: Arman Cohan]

Making decoding more efficient

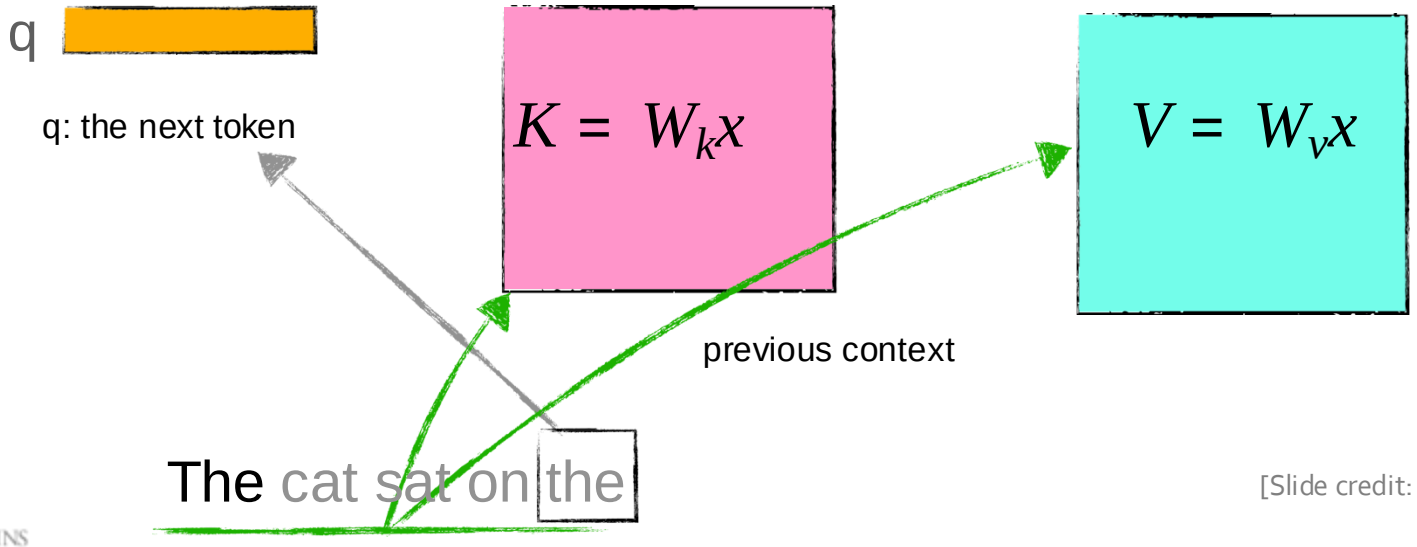
$$Q = \mathbf{W}^q \mathbf{x}$$
$$K = \mathbf{W}^k \mathbf{x}$$
$$V = \mathbf{W}^v \mathbf{x}$$
$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d}}\right)V$$



[Slide credit: Arman Cohan]

Making decoding more efficient

$$Q = \mathbf{W}^q \mathbf{x}$$
$$K = \mathbf{W}^k \mathbf{x}$$
$$V = \mathbf{W}^v \mathbf{x}$$
$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d}}\right)V$$



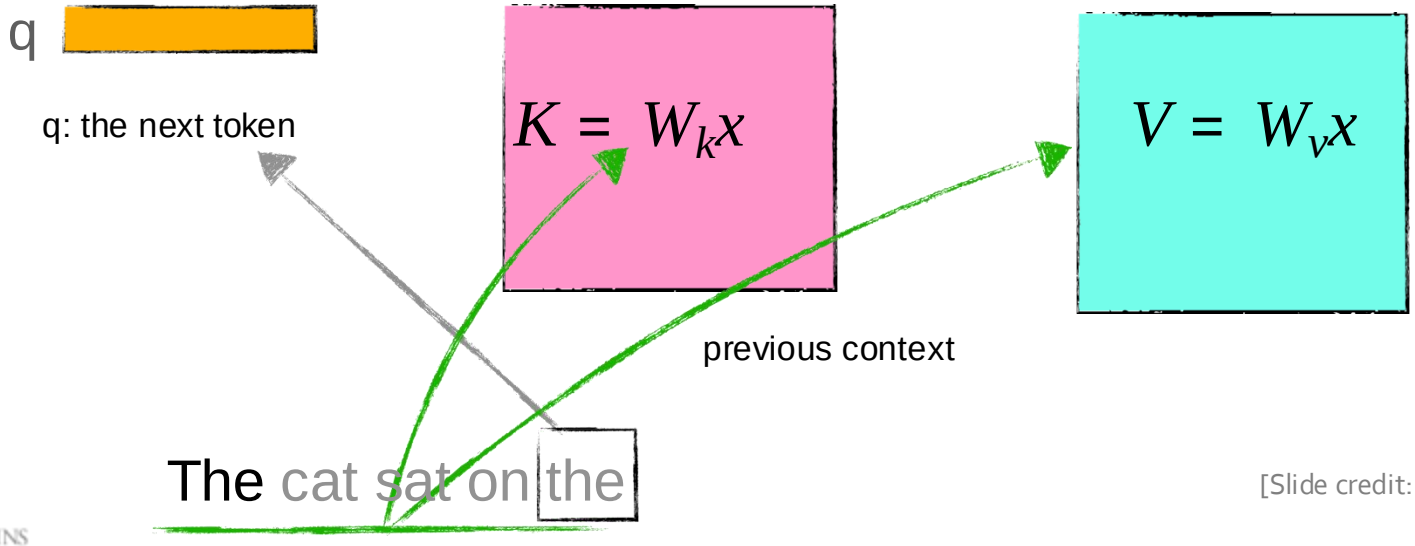
[Slide credit: Arman Cohan]

Making decoding more efficient

$$Q = \mathbf{W}^q \mathbf{x}$$
$$K = \mathbf{W}^k \mathbf{x}$$
$$V = \mathbf{W}^v \mathbf{x}$$

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d}}\right)V$$

- We are computing the Keys and Values many times!
  - Let's reduce redundancy! 😵



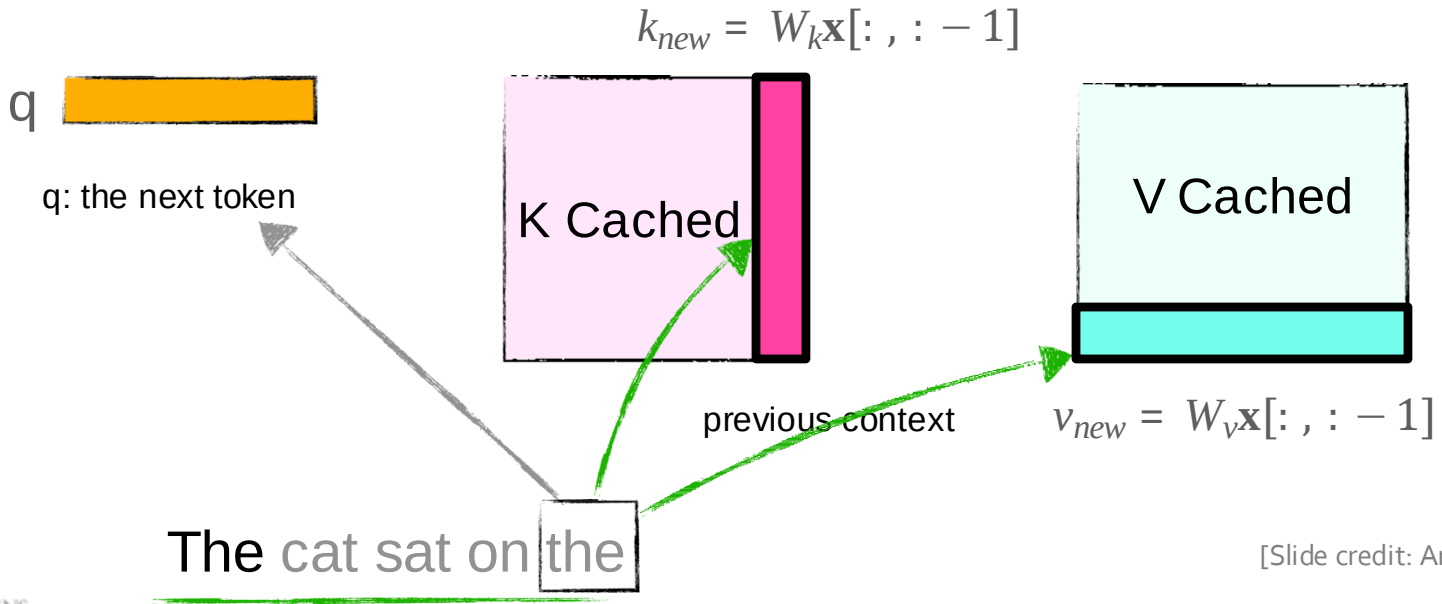
[Slide credit: Arman Cohan]

Making decoding more efficient

$$Q = \mathbf{W}^q\mathbf{x}$$
$$K = \mathbf{W}^k\mathbf{x}$$
$$V = \mathbf{W}^v\mathbf{x}$$

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d}}\right)V$$

- We are computing the Keys and Values many times!
  - Let's reduce redundancy! 😵

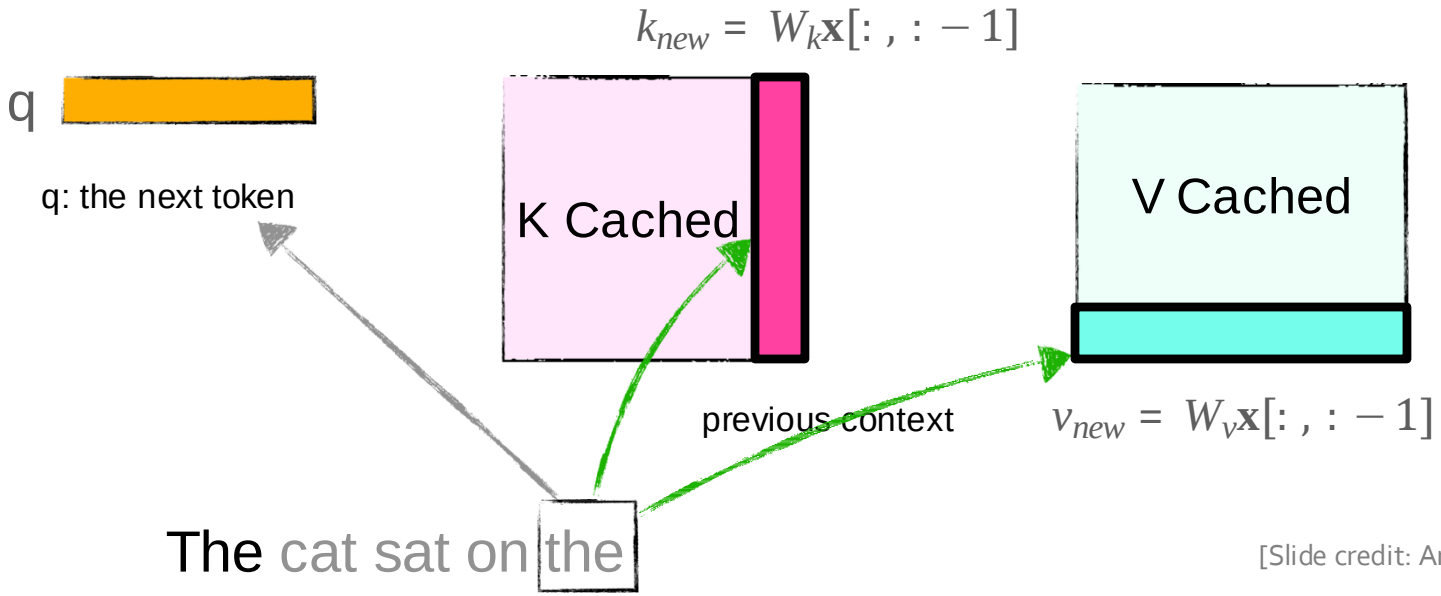


[Slide credit: Arman Cohan]

Making decoding more efficient

$$Q = \mathbf{W}^q\mathbf{x}$$
$$K = \mathbf{W}^k\mathbf{x}$$
$$V = \mathbf{W}^v\mathbf{x}$$
$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d}}\right)V$$

- **Question:** How much memory does this K, V cache require?



[Slide credit: Arman Cohan]

Summary

- This is a very generic Transformer!
- We will implement this in HW5 to build a simple Transformer Language Model!!

- **Next:**
  - Architectural variants
  - Efficiency issues.
  - …

Transformer Architectural Variants

Encoder-decoder

- It is possible to have two stacks of transformer layers
- The encoder is as we've seen
- We can also add a decoder layer that is identical to the encoder but we give it the ability to also attend to the input

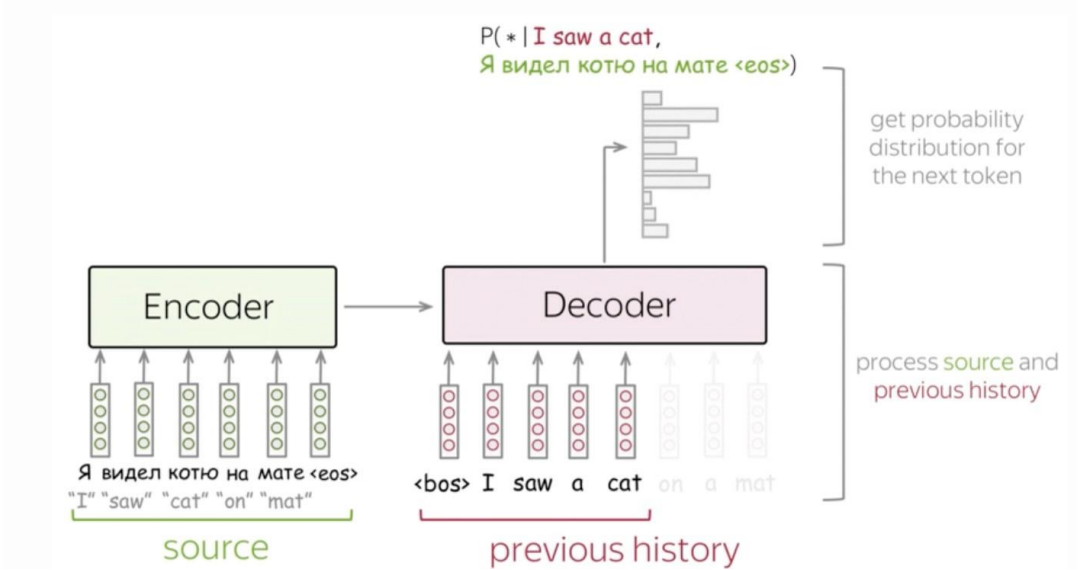
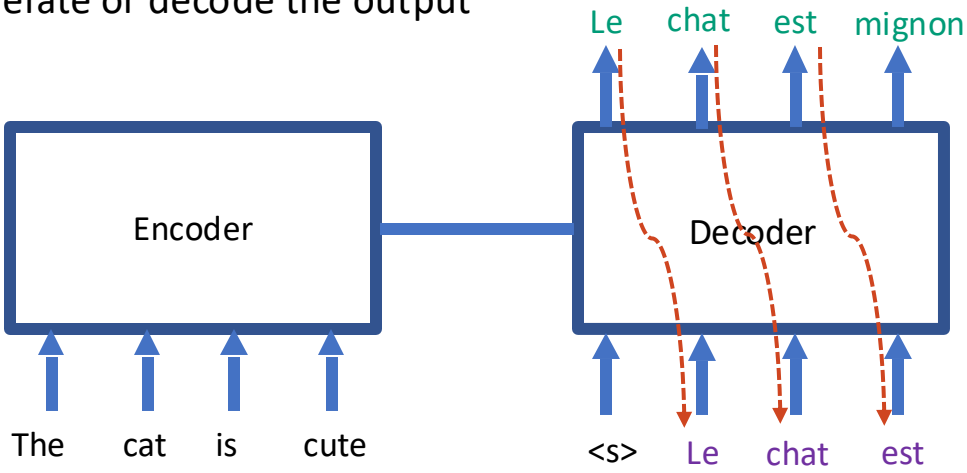


Fig from: https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

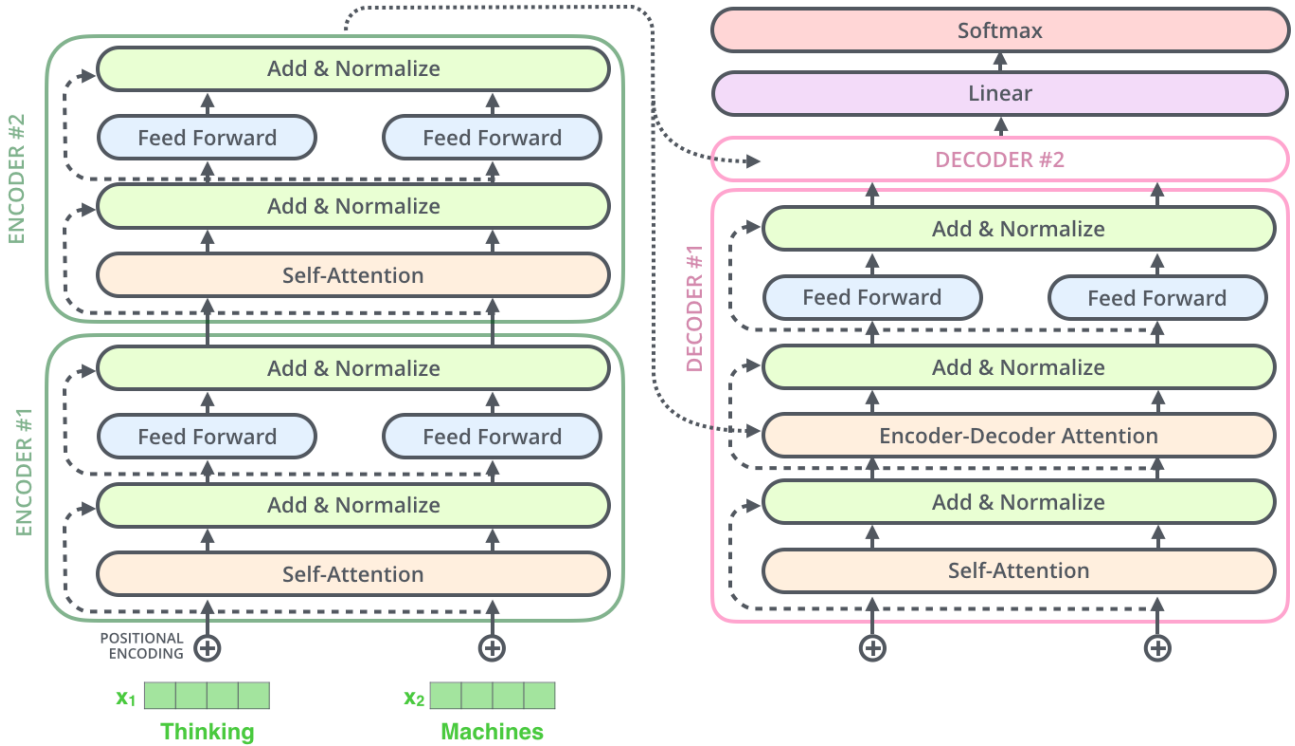
Encoder-decoder models

- Encoder = read or encode the input,
- Decoder = generate or decode the output



Transformer [Vaswani et al. 2017]

- An encoder-decoder architecture built with attention modules.

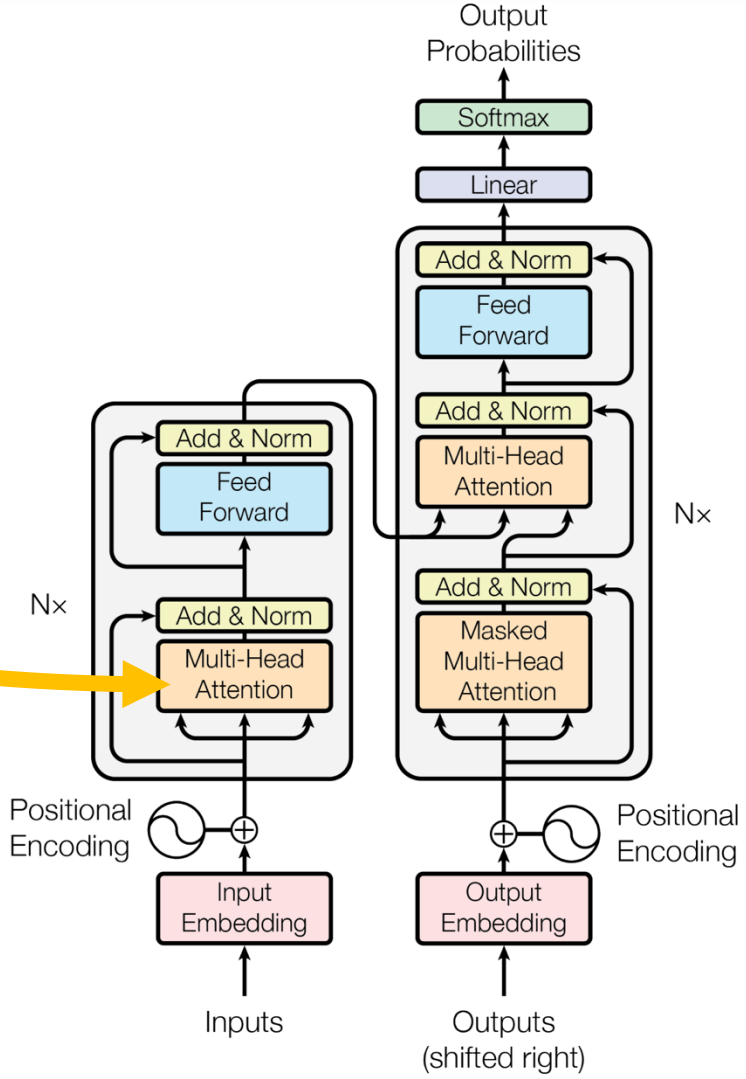


139

Transformer [Vaswani et al. 2017]

- Computation of **encoder** attends to both sides.

Encoder Self-Attention

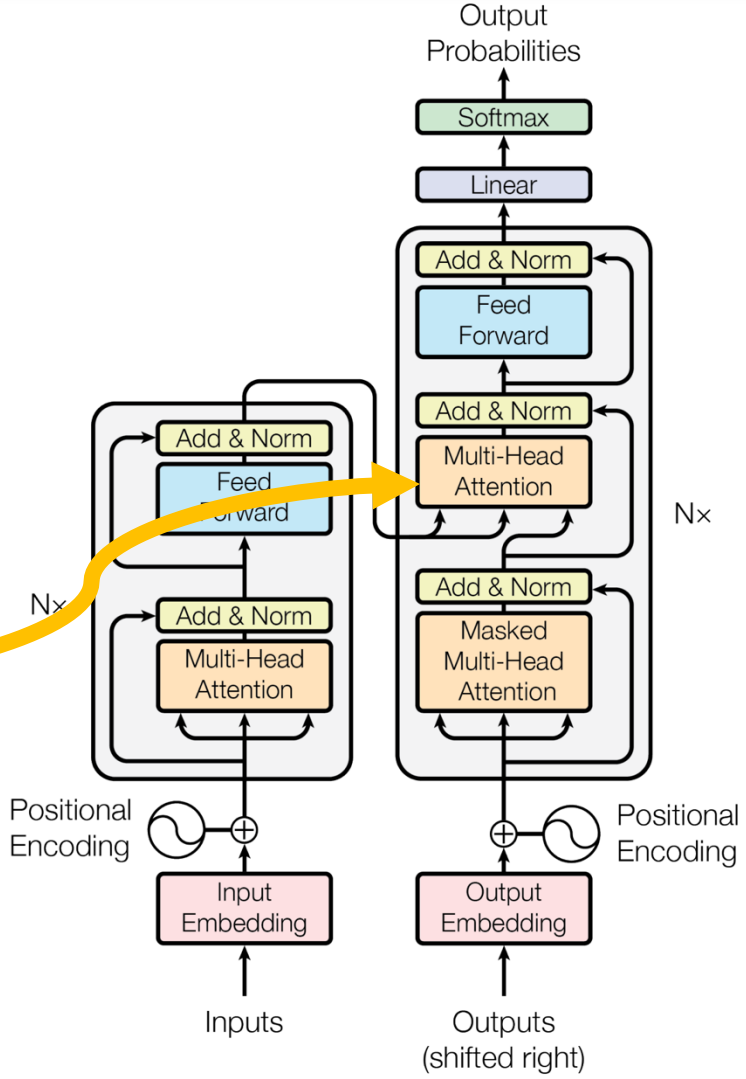


[Attention Is All You Need, Vaswani et al. 2017]

Transformer [Vaswani et al. 2017]

- At any step of **decoder**, it attends to previous computation of **encoder**

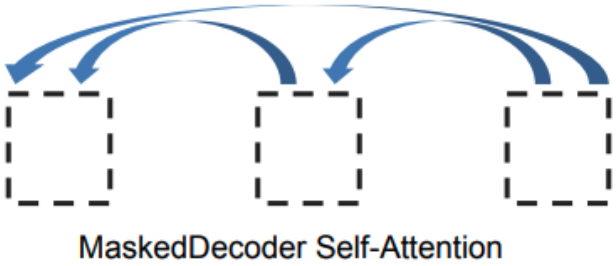
Encoder-Decoder Attention

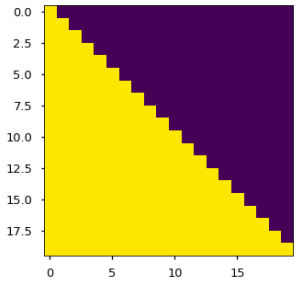


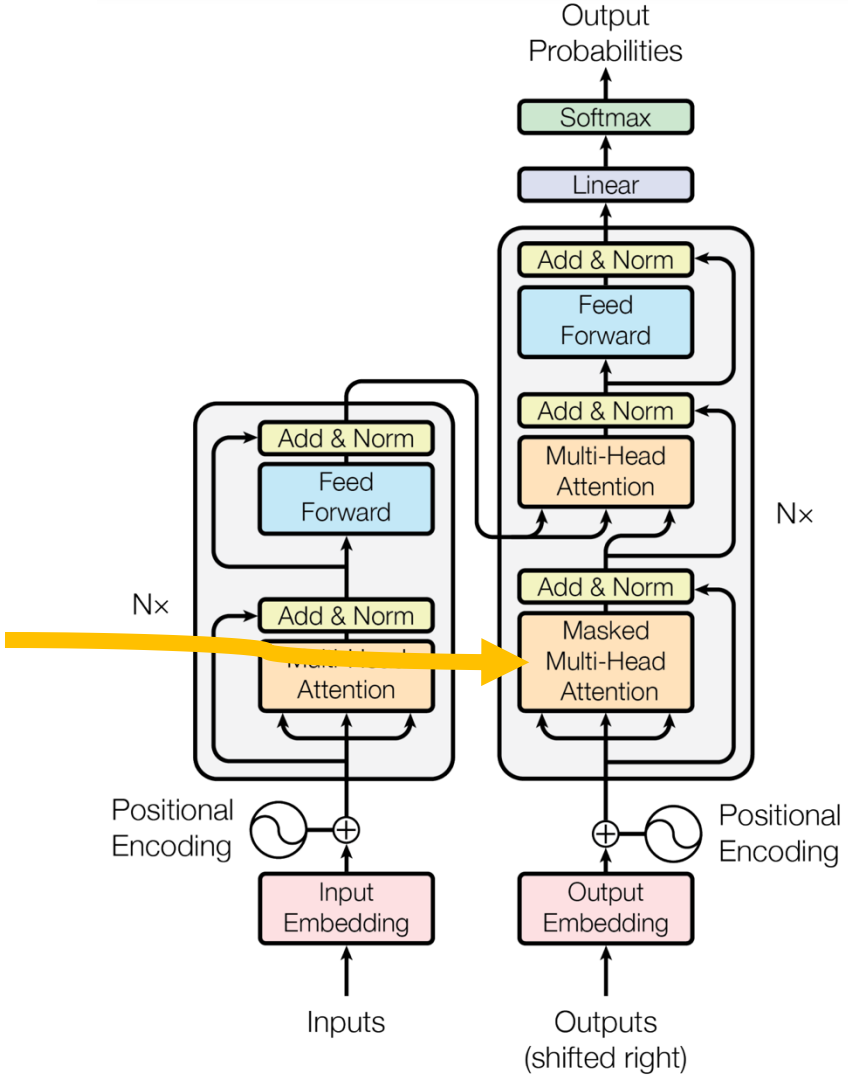
[Attention Is All You Need, Vaswani et al. 2017]

Transformer [Vaswani et al. 2017]

- At any step of **decoder**, it attends to previous computation of **encoder** as well as **decoder's** own generations





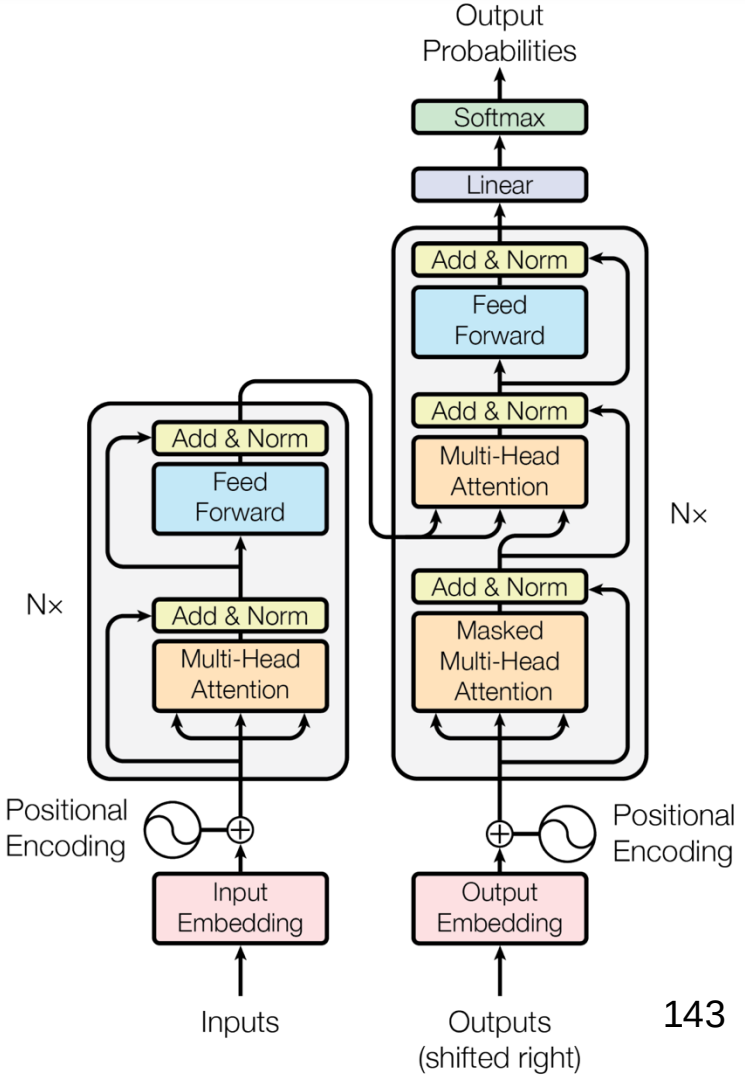


[Attention Is All You Need, Vaswani et al. 2017]

Transformer [Vaswani et al. 2017]

- At any step of **decoder**, it attends to previous computation of **encoder** as well as **decoder's** own generations

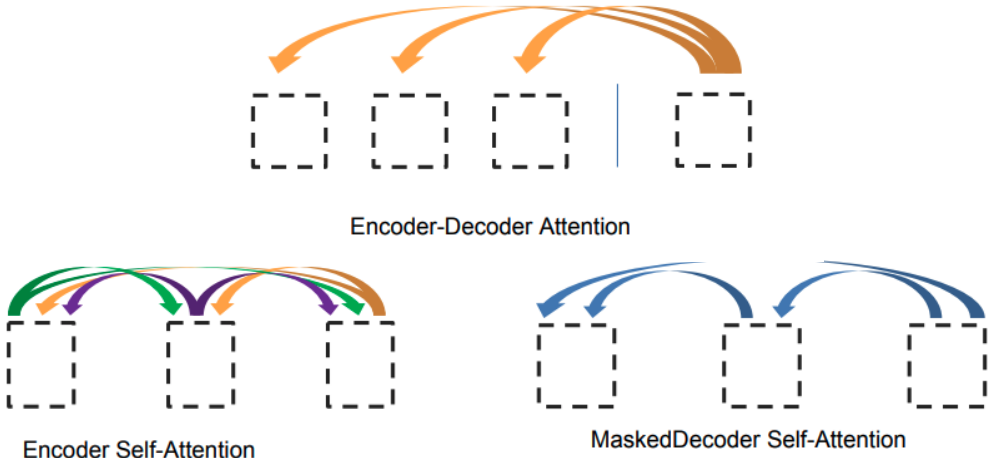
- At any step of decoder, **re-use** previous computation of encoder.
- Computation of decoder is **linear**, instead of quadratic.



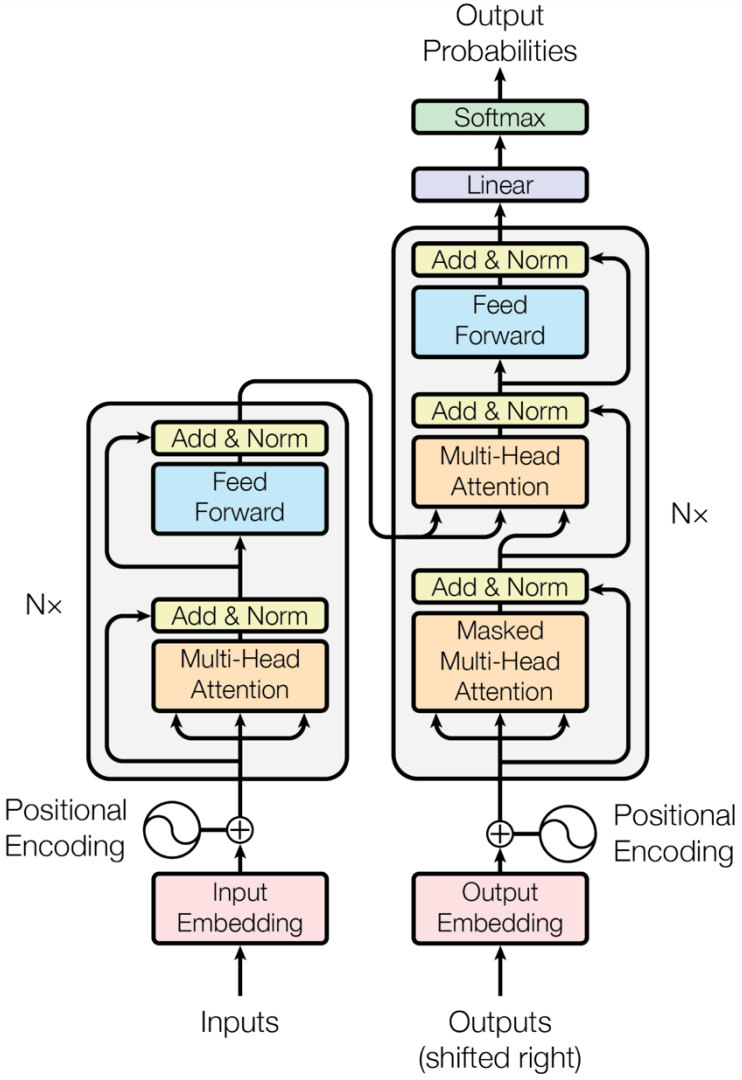
[Attention Is All You Need, Vaswani et al. 2017]

Recap: Transformer

- Yaaay we know Transformers now!
- An encoder-decoder architecture
- 3 forms of attention



[Attention Is All You Need, Vaswani et al. 2017]



Quiz: Enc-Dec Cost

- Source data (large!):
  - The references for a Wikipedia article.
  - Web search using article section titles, ~ 10 web pages per query.

- For a passage of length N and a summary of length M, the complexity of the attention is:
  - $O(N) + O(M)$
  - $O(N) + O(M) + O(NM)$
  - $O(N^2) + O(M^2) + O(NM)$
  - $O(N^2) + O(M^2)$

No, self attention is all-to-all and so quadratic.

[Slide: John Canny]

Quiz: Enc-Dec Cost

- Source data (large!):
  - The references for a Wikipedia article.
  - Web search using article section titles, ~ 10 web pages per query.

- For a passage of length N and a summary of length M, the complexity of the attention is:
  - $O(N) + O(M)$
  - $O(N) + O(M) + O(NM)$
  - $O(N^2) + O(M^2) + O(NM)$
  - $O(N^2) + O(M^2)$

No, self attention is all-to-all and so quadratic in M and N.

[Slide: John Canny]

Quiz: Enc-Dec Cost

- Source data (large!):
  - The references for a Wikipedia article.
  - Web search using article section titles, ~ 10 web pages per query.

- For a passage of length N and a summary of length M, the complexity of the attention is:
  - $O(N) + O(M)$
  - $O(N) + O(M) + O(NM)$
  - $O(N^2) + O(M^2) + O(NM)$
  - $O(N^2) + O(M^2)$

No, self attention is all-to-all and so quadratic in M and N.

[Slide: John Canny]

Quiz: Enc-Dec Cost

- Source data (large!):
  - The references for a Wikipedia article.
  - Web search using article section titles, ~ 10 web pages per query.

- For a passage of length N and a summary of length M, the complexity of the attention is:
  - $O(N) + O(M)$
  - $O(N) + O(M) + O(NM)$
  - $O(N^2) + O(M^2) + O(NM)$
  - $O(N^2) + O(M^2)$

No, cross attention is missing.

[Slide: John Canny]

Quiz: Enc-Dec Cost

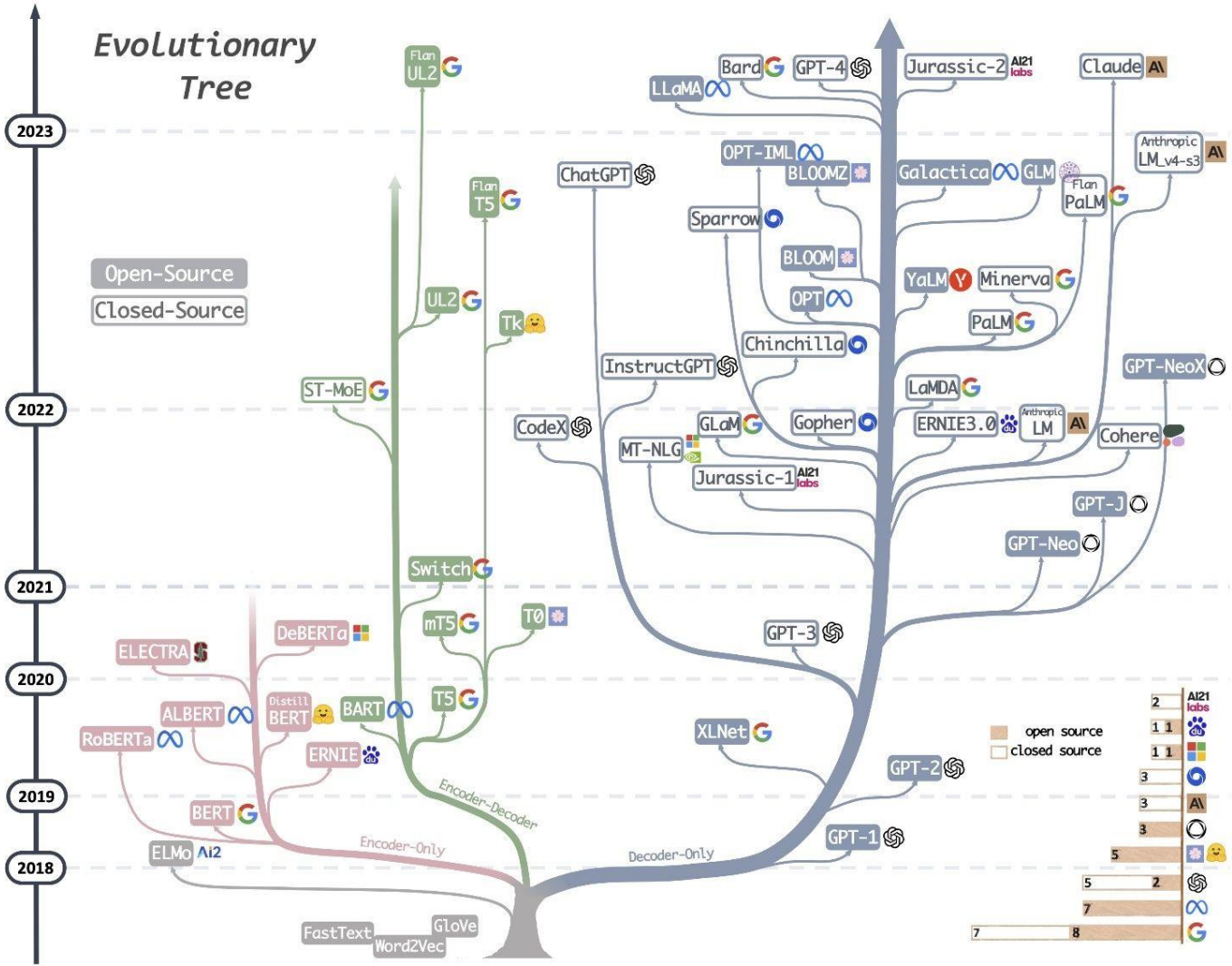
- Source data (large!):
  - The references for a Wikipedia article.
  - Web search using article section titles, ~ 10 web pages per query.

- For a passage of length N and a summary of length M, the complexity of the attention is:
  - $O(N) + O(M)$
  - $O(N) + O(M) + O(NM)$
  - $O(N^2) + O(M^2) + O(NM)$
  - $O(N^2) + O(M^2)$

Yes. The three terms are respectively the Encoder self-attention, Decoder self-attention, and Cross attention.

[Slide: John Canny]

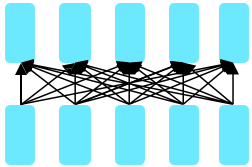
After Transformer ...



Yang et al. Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond, 2023

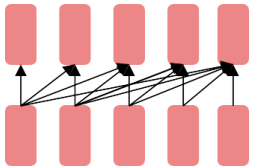
Impact of Transformers

- A building block for a variety of LMs



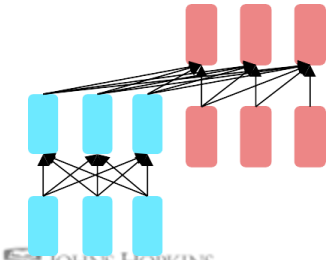
Encoders

- 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

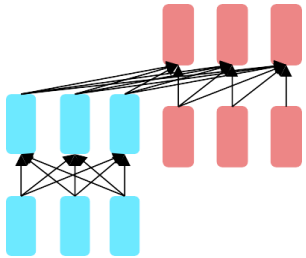


Encoder-Decoders

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

Transformer Language Model Families

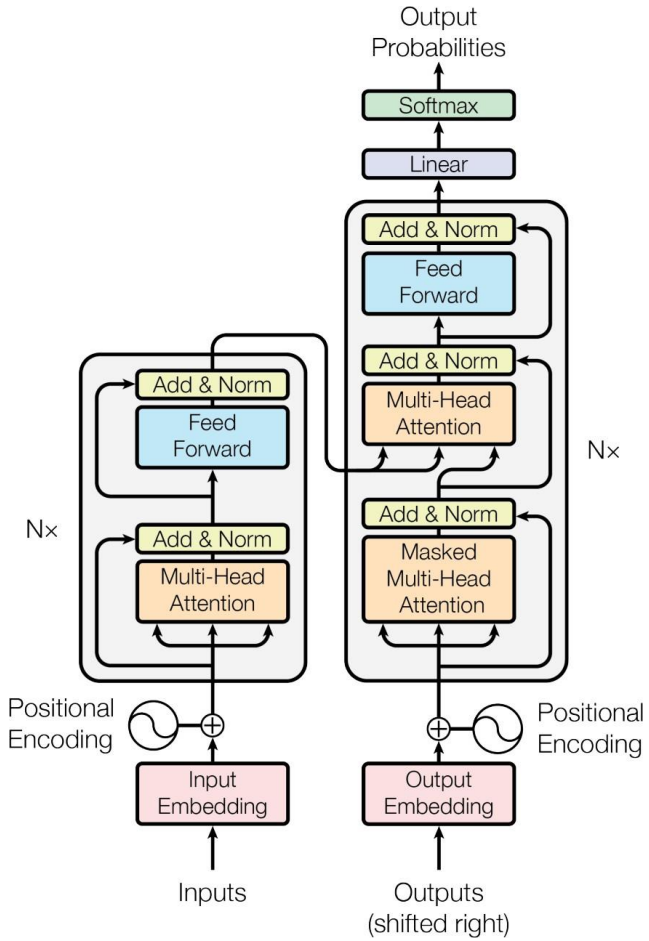
Encoder-Decoder Family of Transformers



Encoder-Decoders

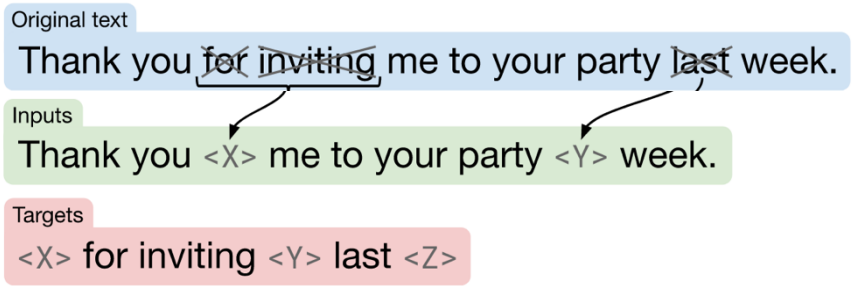
Encoder-decoder Models

- The original transformer architecture was encoder decoder
- Encoder-decoder models are flexible in both generation and classification tasks
- How can we pretrain an encoder-decoder model like BERT to be a good general language pretrained LM?



T5: Text-To-Text Transfer Transformer

- An encoder-decoder architecture
- Pre-training objective:
  corrupt and reconstruct objective



Model	Parameters	No. of layers	d_{model}	d_{ff}	d_{kv}	No. of heads
Small	60M	6	512	2048	64	8
Base	220M	12	768	3072	64	12
Large	770M	24	1024	4096	64	16
3B	3B	24	1024	16384	128	32
11B	11B	24	1024	65536	128	128

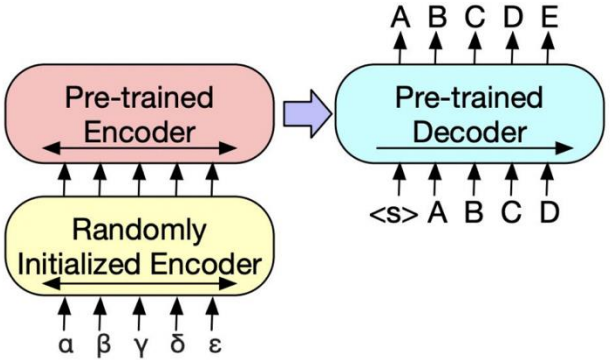
- The original paper is an excellent set of in-depth analysis of various parameters of model design. We discuss some of these results in other places.

`https://huggingface.co/t5-base`

Exploring the limits of transfer learning with text-to-text transfer transformers, 2020

BART (Lewis et al. 2020)

- Similar Architecture as T5.
  - Corrupt the input -> ask the model to reconstruct the original input
  - Outperformed existing methods on generative tasks (question answering, and summarization).



BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension

Mike Lewis*, Yinhan Liu*, Naman Goyal*, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, Luke Zettlemoyer
Facebook AI
{mikelewis,yinhanliu,naman}@fb.com

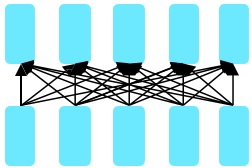
BART

The code might be outdated, but the logic is the same …

```
from transformers import BartTokenizer, BartForConditionalGeneration

tokenizer = BartTokenizer.from_pretrained("facebook/bart-large")
model = BartForConditionalGeneration.from_pretrained("facebook/bart-large")

TXT = "The sun is <mask> ."
input_ids = tokenizer([TXT], return_tensors="pt")["input_ids"]
logits = model(input_ids).logits

masked_index = (input_ids[0] == tokenizer.mask_token_id).nonzero().item()
probs = logits[0, masked_index].softmax(dim=0)
values, predictions = probs.topk(5)

tokenizer.decode(predictions).split()
```

Result: `['located', 'at', 'approximately', 'also', 'about']`

Encoder-only Family of Transformers



BERT

Bidirectional Encoder Representations from Transformers

BERT

Bidirectional Encoder Representations from Transformers

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

BERT

Bidirectional Encoder Representations from Transformers

Let's only use Transformer Encoders, no Decoders

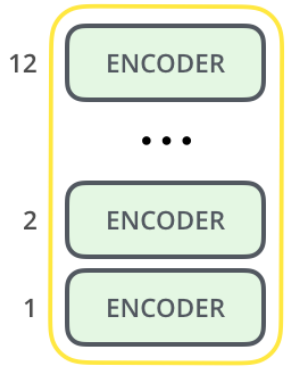
BERT

Bidirectional Encoder Representations from Transformers

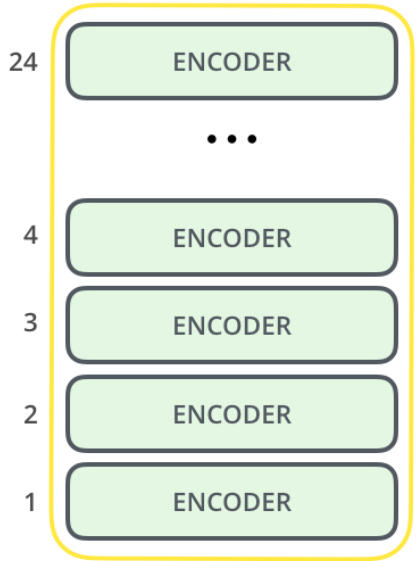
It's a language model that builds rich representations
via self-supervised learning (pre-training)

BERT: Architecture

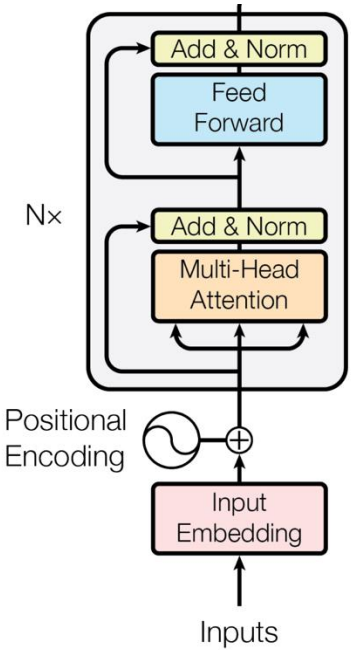
- Stacks of Transformer encoders



$BERT_{BASE}$



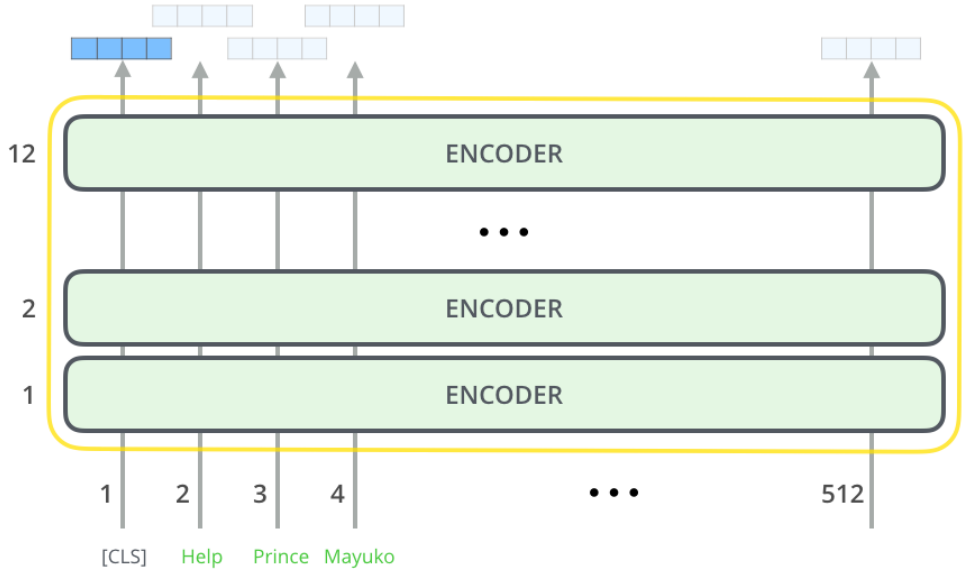
$BERT_{LARGE}$

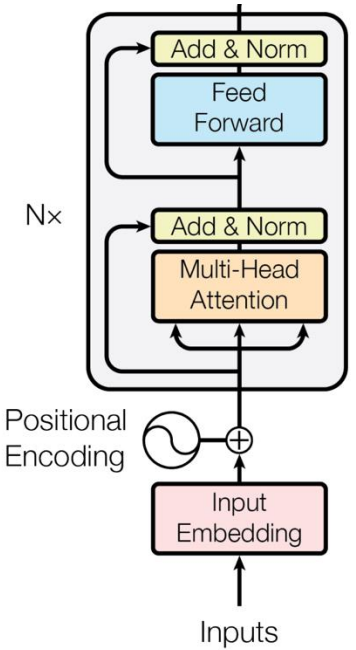


[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]

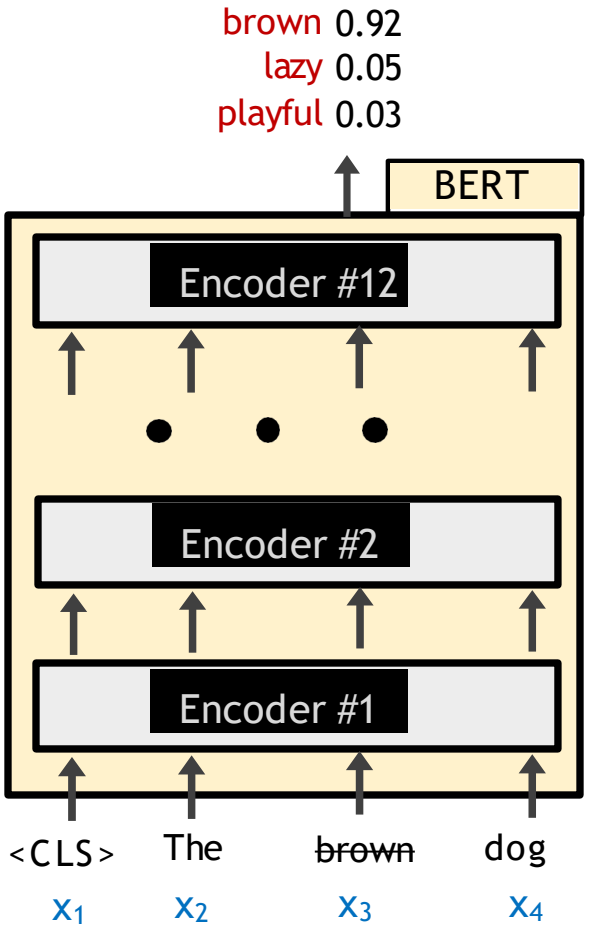
BERT: Architecture

- Model output dimension: 512





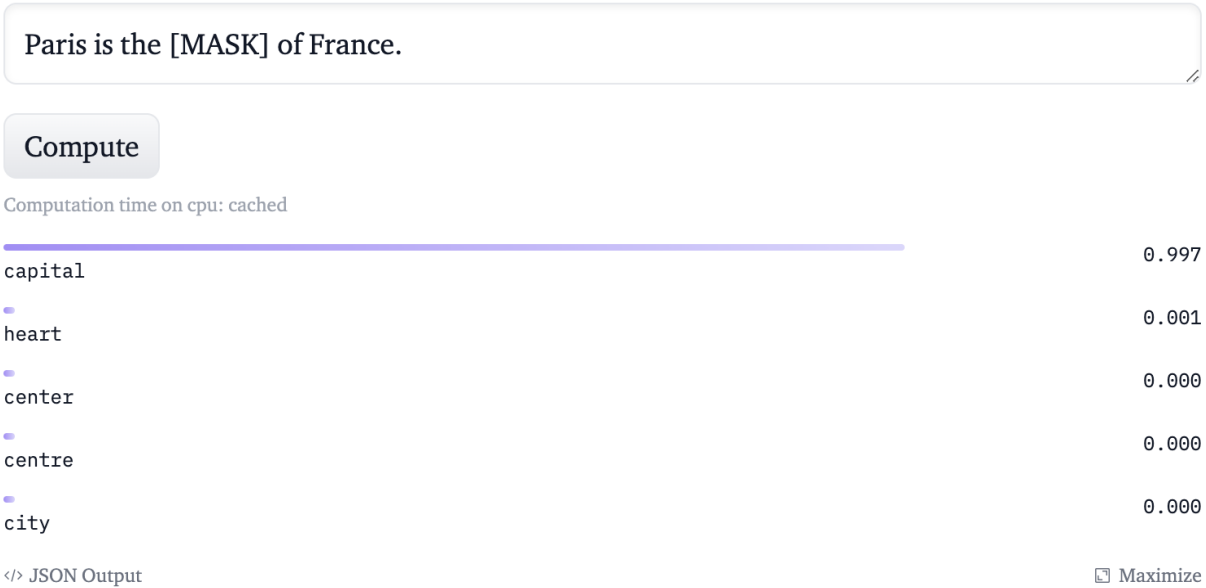
[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]



BERT is trained to uncover masked tokens.

Probing BERT Masked LM

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

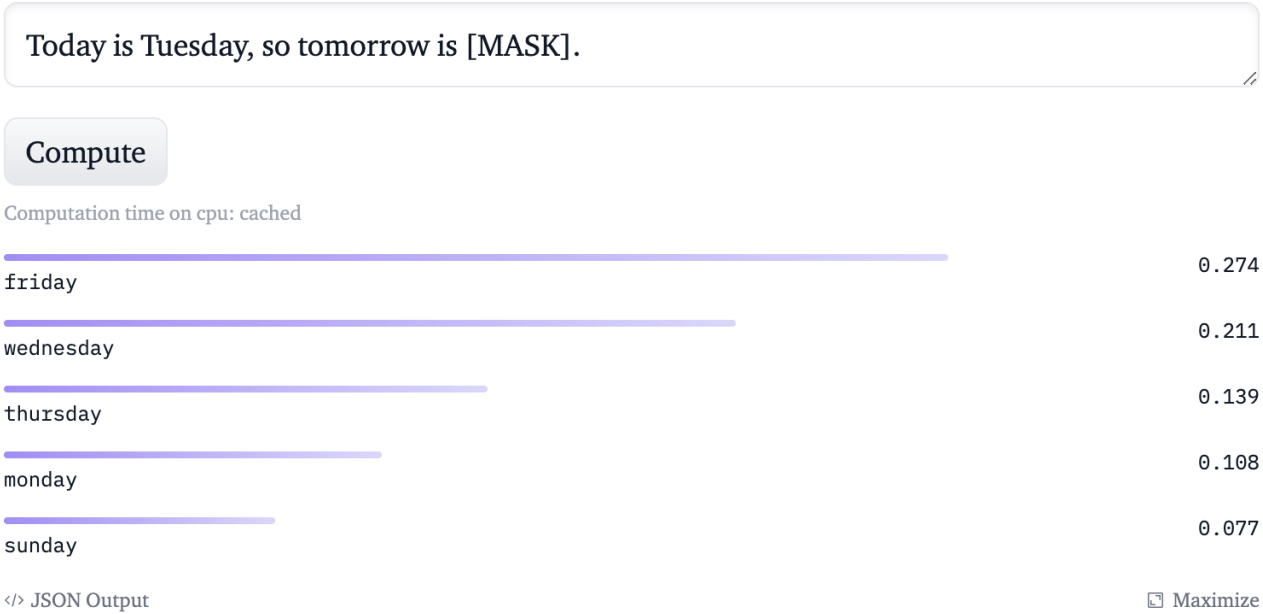


https://huggingface.co/bert-base-uncased

168

Probing BERT Masked LM

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

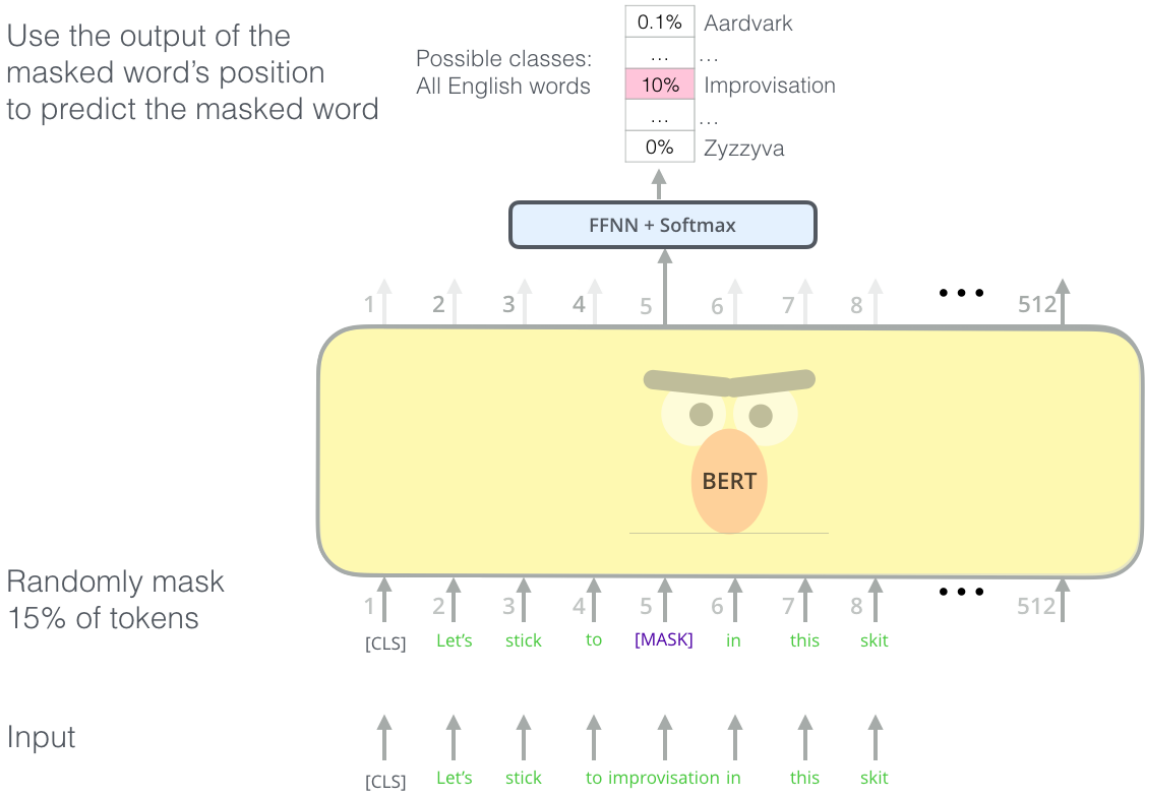


https://huggingface.co/bert-base-uncased

169

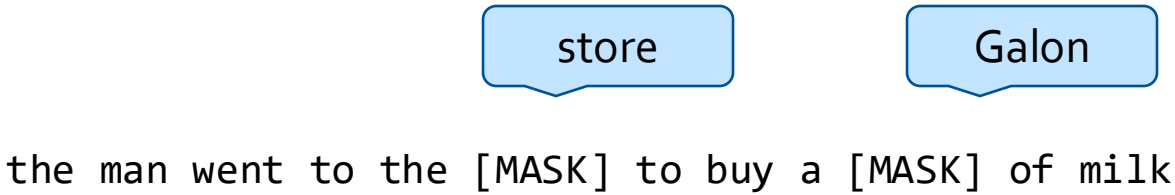
BERT: Pre-training Objective (1): Masked Tokens

- 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



- Too little masking: Too expensive to train
- Too much masking: Underdefined
  - (not enough info for the model to recover the masked tokens)

Later work shows that more principled masking (instead of uniformly random) could benefit downstream task performance and result in faster training.

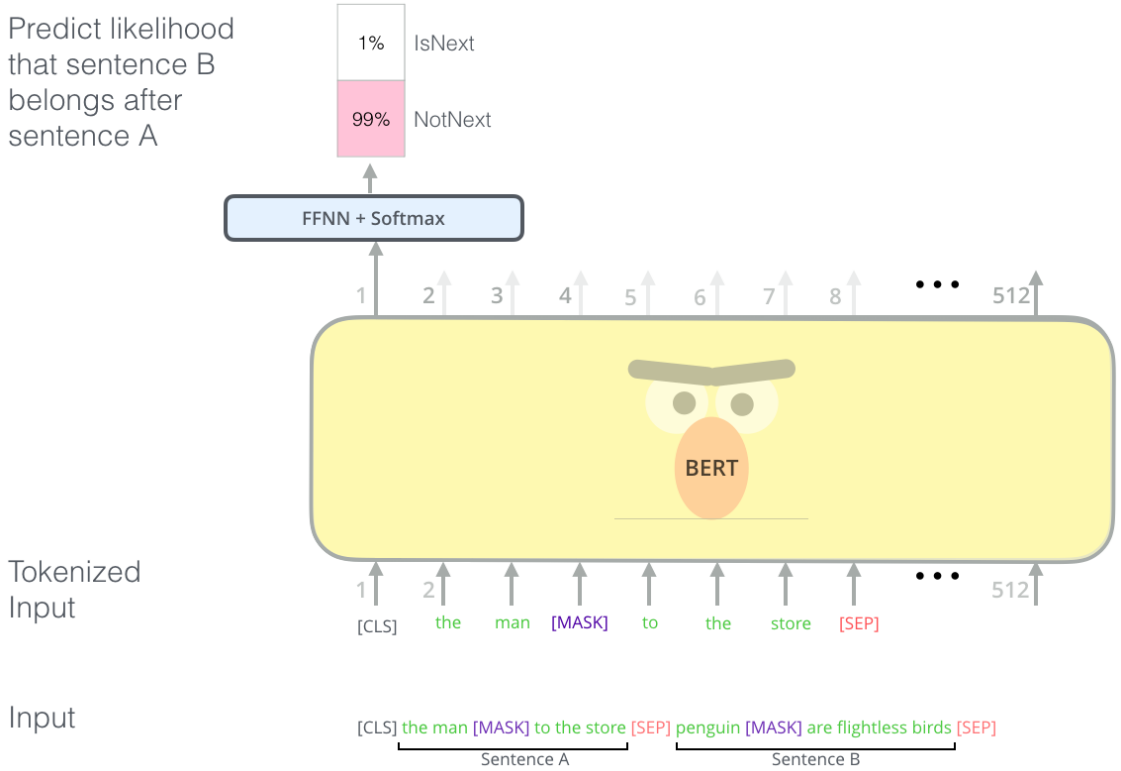
PMI Masking (Levine et al., 2021) https://arxiv.org/pdf/2010.01825.pdf
SpanBERT (Joshi et al., 2020) https://arxiv.org/pdf/1907.10529.pdf

[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

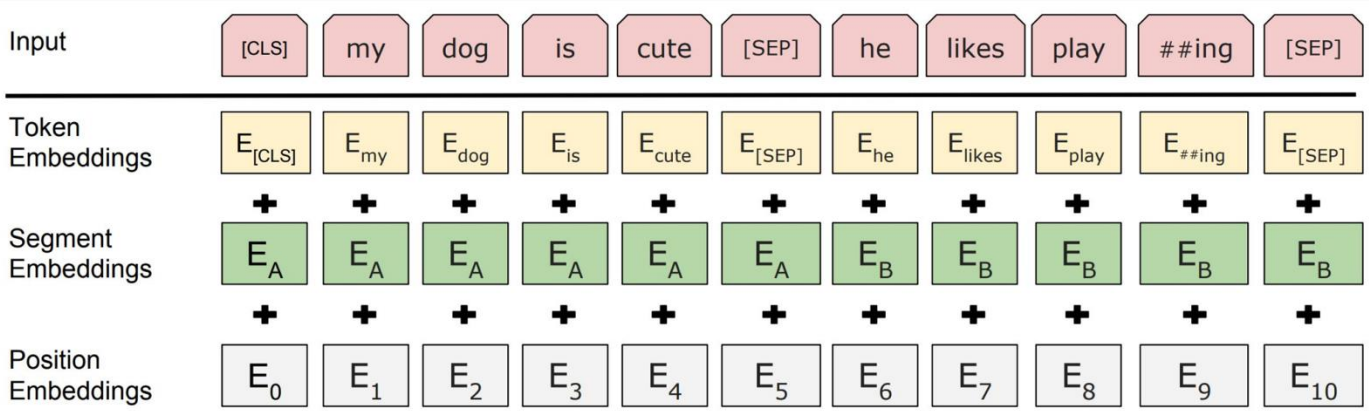
Sentence A = `The man went to the store.`
Sentence B = `He bought a gallon of milk.`
Label = `IsNextSentence`

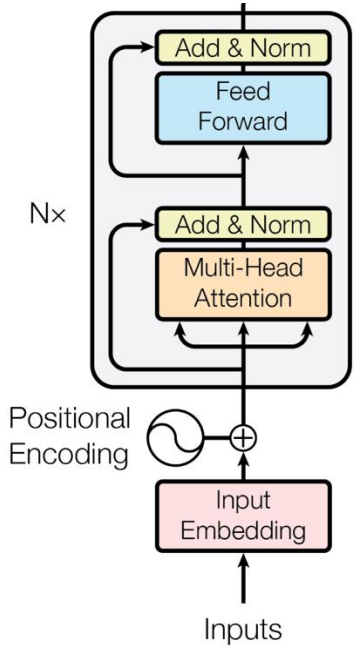
Sentence A = `The man went to the store.`
Sentence B = `Penguins are flightless.`
Label = `NotNextSentence`

[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

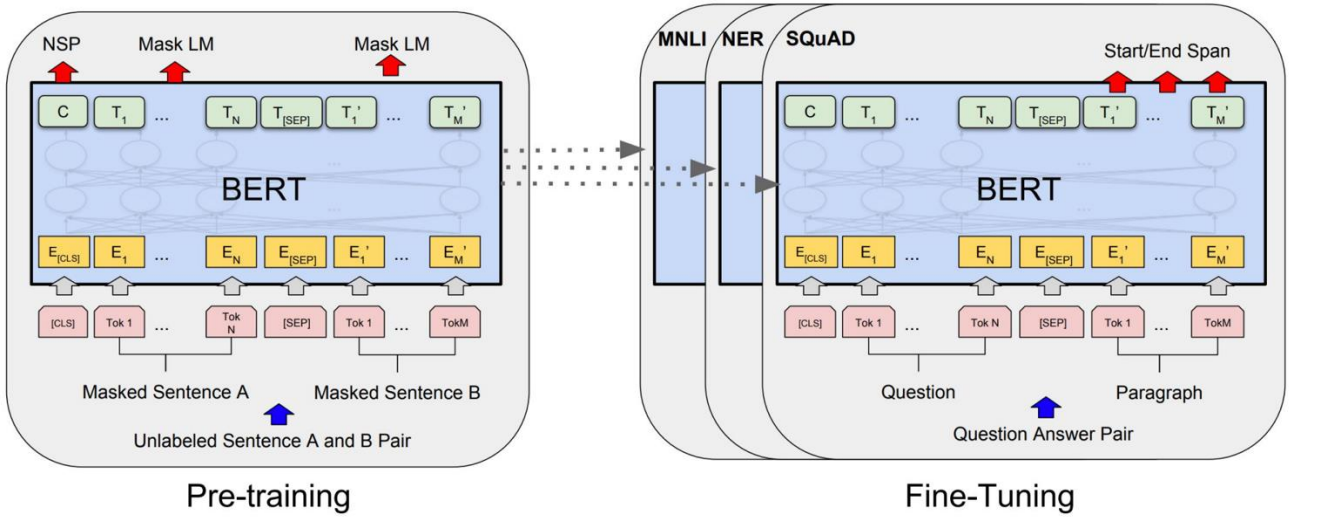
- Trains model on unlabeled data over different pre-training tasks (self-supervised learning)
- **Data:** Wikipedia (2.5B words) + BookCorpus (0.8B words)
- **Training Time:** 1M steps (~40 epochs)
- **Optimizer:** AdamW, 1e-4 learning rate, linear decay
- **BERT-Base:** 12-layer, 768-hidden, 12-head, sequence length of 512
- **BERT-Large:** 24-layer, 1024-hidden, 16-head, sequence length of 512
- Trained on 4x4 and 8x8 TPUs for 4 days (cost today using cloud TPU: $1.3K and $5K)

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

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]

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

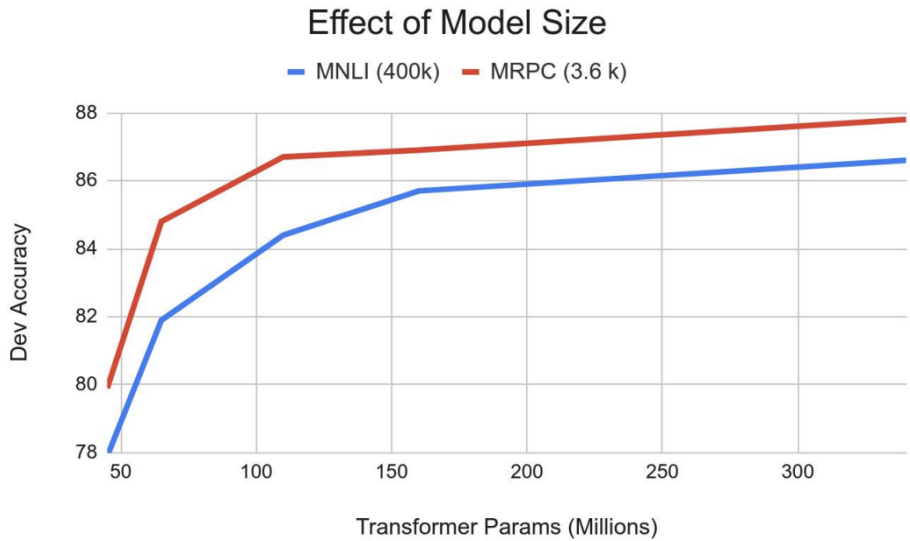
Human Performance (88.00%)
Running Best
Submissions

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 Size



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

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

Impact of BERT

- In order to have state-of-the-art performance on different tasks, there is no need for coming up with a novel model architecture
  - End of task-specific model architecture engineering

- An early sign that larger scales and self-supervised learning (language modeling) are the key for future performance improvements

Why did no one think of this before?

- 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)
  - Exact same architecture as BERT
  - 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)

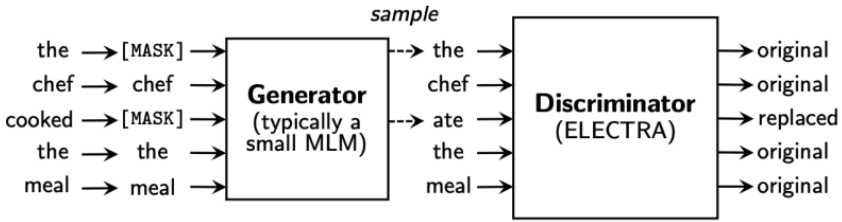
Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$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

What Happened After BERT?

- RoBERTa (Liu et al., 2019)
  - Exact same architecture as BERT
  - 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)

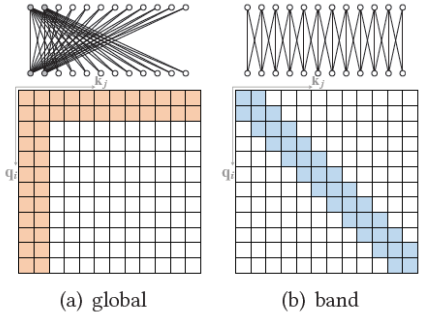
- ALBERT (Lan et al., 2020)
  - Increasing model sizes by sharing model parameters across layers
  - Less storage, much stronger performance but runs slower..

- ELECTRA (Clark et al., 2020)
  - Pre-training objective: replaced-token detection
  - Two models generator and discriminator (GAN-like)
  - It provides a more efficient training method

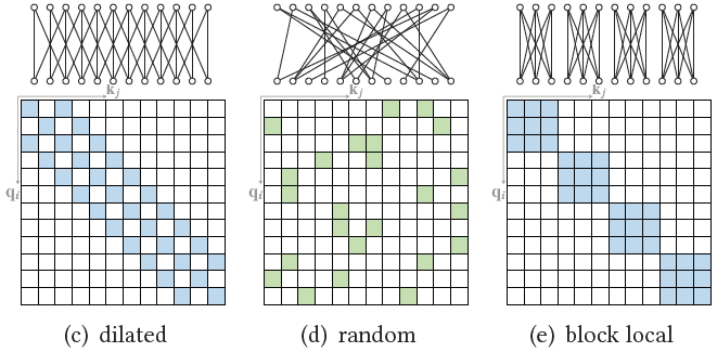


What Happened After BERT?

- Models that handle long contexts
  - Longformer, Big Bird, …
- Multilingual BERT
  - Trained single model on 104 languages from Wikipedia.
- BERT extended to different domains
  - SciBERT, BioBERT, FinBERT, ClinicalBERT, …
- Making BERT smaller to use
  - DistillBERT, TinyBERT, …



(a) global (b) band



(c) dilated (d) random (e) block local

Text generation using BERT

- Does not support generation or sequence-to-sequence tasks
  - Summarization, Translation, Text simplification, etc

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 .

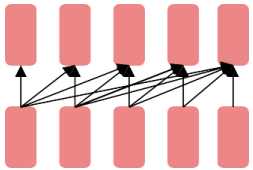
Leveraging Pre-trained Checkpoints for Sequence Generation Tasks

Sascha Rothe, Shashi Narayan, Aliaksei Severyn

Summary Thus Far

- BERT and the family
- An encoder; Transformer-based networks trained on massive piles of data.
- Incredible for learning contextualized embeddings of words
- It's very useful to pre-train a large unsupervised/self-supervised LM then fine-tune on your particular task (replace the top layer, so that it can work)
- However, they were not designed to generate text.

Decoder-only Family of Transformers

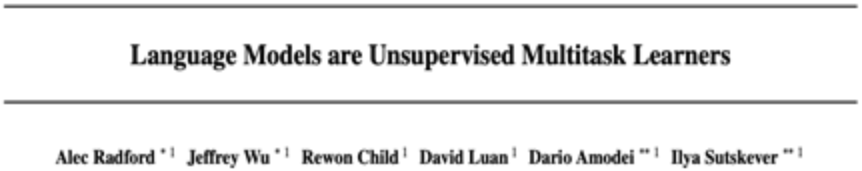


Decoders

GPT

Generative Pre-trained Transformer

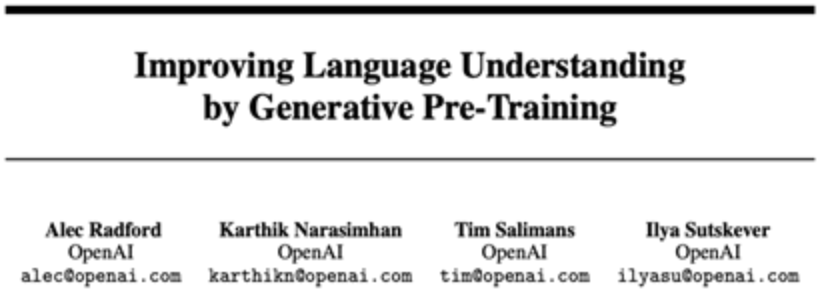
GPT-2: A Big Language Model (2019)



Language Models are Unsupervised Multitask Learners

Alec Radford [*1] Jeffrey Wu [*1] Rewon Child [1] David Luan [1] Dario Amodei [**1] Ilya Sutskever [**1]

GPT: An Auto-Regressive LM (2018)



Improving Language Understanding by Generative Pre-Training

Alec Radford
OpenAI
alec@openai.com

Karthik Narasimhan
OpenAI
karthikn@openai.com

Tim Salimans
OpenAI
tim@openai.com

Ilya Sutskever
OpenAI
ilyasu@openai.com

GPT-2

- GPT-2 uses only Transformer Decoders (no Encoders) to generate new sequences from scratch or from a starting sequence
- As it processes each subword, it masks the "future" words and conditions on and attends to the previous words

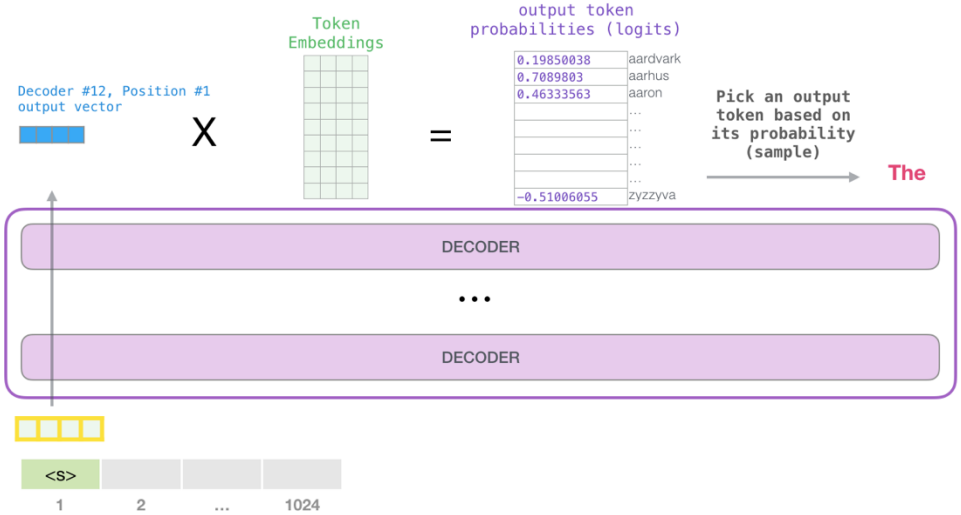
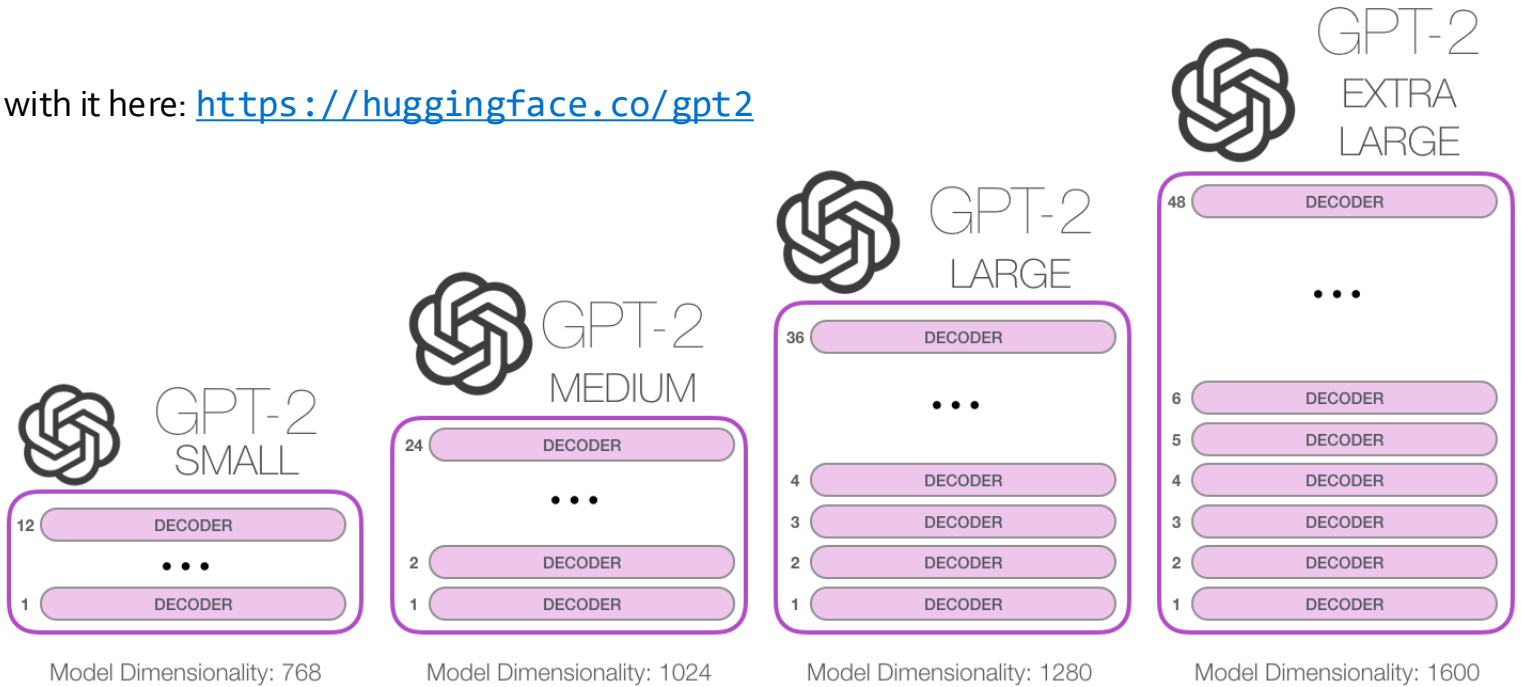


Image by http://jalammar.github.io/illustrated-gpt2/

GPT2: Model Sizes

Play with it here: https://huggingface.co/gpt2



117M parameters 345M 762M 1542M

[Image by http://jalammar.github.io/illustrated-gpt2/]

GPT2: Some Results

Language Models are Unsupervised Multitask Learners

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	56.25	85.7	82.3	39.14	46.54	0.99	1.08	18.3	**21.8**
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**	65.85	1.16	1.17	37.50	75.20
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**	47.33	1.01	**1.06**	26.37	55.72
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**	**40.31**	**0.97**	**1.02**	22.05	44.575
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**	**35.76**	**0.93**	**0.98**	**17.48**	42.16

Table 3. Zero-shot results on many datasets. No training or fine-tuning was performed for any of these results. PTB and WikiText-2 results are from (Gong et al., 2018). Other language model results are from (Dai et al., 2019).

Article: Prehistoric man sketched an incredible array of prehistoric beasts on the rough limestone walls of a cave in modern day France 36,000 years ago.
Now, with the help of cutting-edge technology, those works of art in the Chauvet-Pont-d'Arc Cave have been reproduced to create the biggest replica cave in the world.
The manmade cavern named the Caverne du Pont-d'Arc has been built a few miles from the original site in Vallon-Pont-D'arc in Southern France and contains 1,000 painstakingly-reproduced drawings as well as around 450 bones and other features...
Cavemen and women sketched an incredible array of prehistoric beasts on the rough limestone walls of a cave 36,000 years ago and now a replica has been created (pictured)
...

GPT-2: The original site in Vallon-Pont-D'arc in Southern France is a Unesco World Heritage site and is the oldest known and the best preserved cave decorated by man. The replica cave was built a few miles from the original site in Vallon-Pont-D'Arc in Southern France. The cave contains images of 14 different species of animals including woolly rhinoceros, mammoths, and big cats.	**Reference:** Cave mimics famous Caverne du Pont-d'Arc in France, the oldest cave decorated by man and the best preserved. The replica contains all 1,000 paintings which include 425 such as a woolly rhinoceros and mammoths. Minute details were copied using 3D modelling and anamorphic techniques, often used to shoot widescreen images. The modern cave also includes replica paw prints of bears, bones and details preserved in the original cave.

Impact of GPT2

- Zero-shot learning (no use of task-specific supervision) increasingly become a reality.

NMT: "Translate to french," <English text>, <French text>.

QA: "Answer the question," <Document>, <Question>, <Answer>.

SUMM: <Document> "TL; DR:" <Summarization>

GPT-3: A Very Large Language Model (2020)

- More layers & parameters
- Bigger dataset
- Longer training
- Larger embedding/hidden dimension
- Larger context window

[Slide credit: Sbhya Chhabria & Michael Tang]

Impact of GPT3

- Moving away from the fine-tuning paradigm
  - Zero/Few-shot learning and in-context learning
- Massive LM scale makes high zero/few-shot performance possible
- Start of closed source models
  - Not too many details about their model
  - No released code / model checkpoint
- Also revitalized ppen source efforts:
  - OPT, LLaMA by Meta, BLOOM by Huggingface, etc.

GPT4

Model	Usage
davinci-002	$0.0020 / 1K tokens

Model	Input	Output
gpt-4	$0.03 / 1K tokens	$0.06 / 1K tokens

- Transformer-based
  - The rest is …. mystery! ☺
  - If we're going based on costs, GPT4 is ~15-30 times costlier than GPT3. That should give you an idea how its likely size!

- Note, these language models involve more than just pre-training.
  - Pre-training provides the foundation based on which we build the model.
  - We will discuss the later stages (post hoc alignment) in a 2-3 weeks.

https://openai.com/pricing

Accessing API Models

```
import openai
openai.api_key = ("sk-                                                    ")
my_prompt = '''The sun is [MASK].

    Replace [MASK] with the most probable 5 words to replace, and give me their probabilities.'''
# Here set parameters as you like
response = openai.Completion.create(
  engine="text-davinci-002",
  prompt=my_prompt,
  temperature=0,
  max_tokens=100,
)

print(response['choices'][0]['text'])
```

Other Available [Decoder] LMs

EleutherAI: GPT-Neo (6.7B), GPT-J (6B), GPT-NeoX (20B)

https://huggingface.co/EleutherAI

https://6b.eleuther.ai/

LLaMA, 65B: https://github.com/facebookresearch/llama

Mistral and Mixtral:

https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2

https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1

Training Transformer LMs: Empirical Considerations

Pre-training Transformer LMs

- You have learned about the basics of pre-training Transformer language models.
- There is so much empirical knowledge/experiences that goes into training these models.
- Various empirical issues about:
  - Preparation/pre-processing data
  - Efficient training of models
  - …

C4: The Data

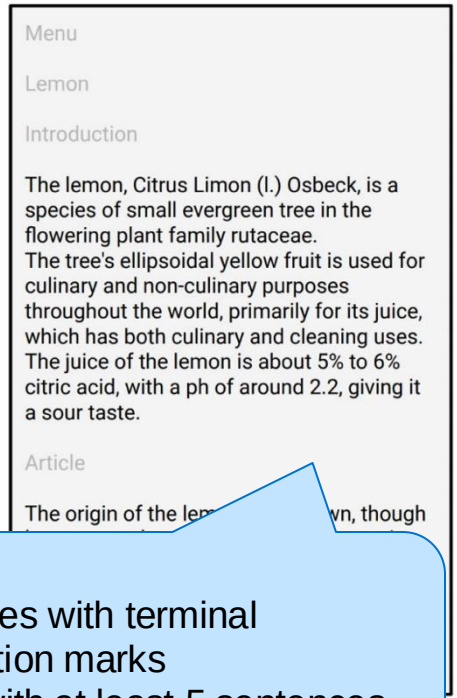
- C4: Colossal Clean Crawled Corpus
  - Web-extracted text
  - English language only
  - 750GB

Data set	Size
★ C4	745GB
C4, unfiltered	6.1TB

Play with the data: https://c4-search.apps.allenai.org/

C4: The Data

Remove any:
- References to Javascript
- "Lorem ipsum" text — placeholder text commonly used to demonstrate the visual form of a document



Menu

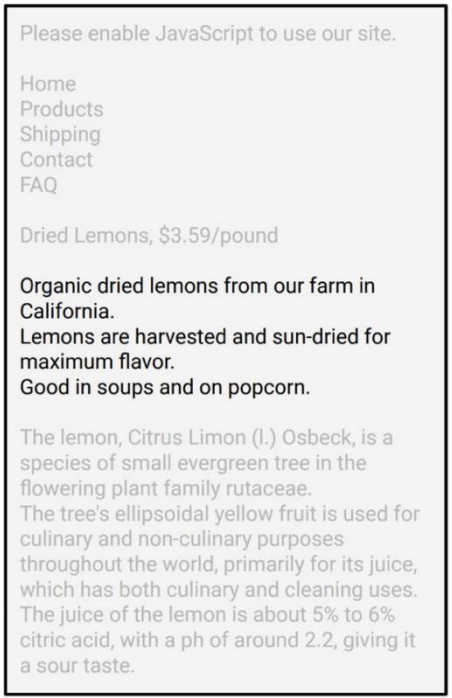
Lemon

Introduction

The lemon, Citrus Limon (l.) Osbeck, is a species of small evergreen tree in the flowering plant family rutaceae.
The tree's ellipsoidal yellow fruit is used for culinary and non-culinary purposes throughout the world, primarily for its juice, which has both culinary and cleaning uses. The juice of the lemon is about 5% to 6% citric acid, with a ph of around 2.2, giving it a sour taste.

Article

The origin of the lem wn, though



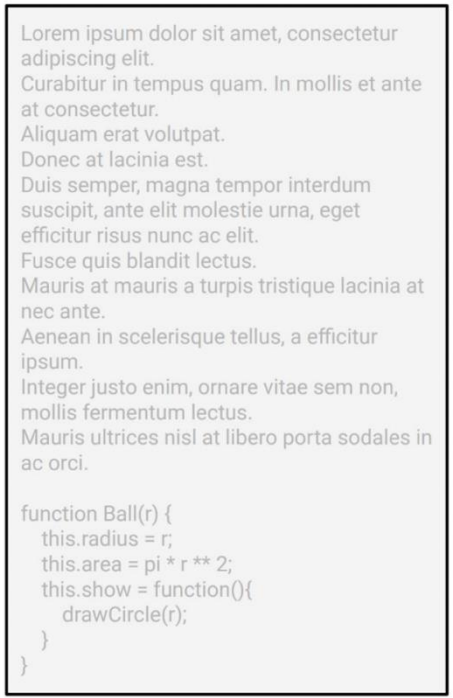
Please enable JavaScript to use our site.

Home
Products
Shipping
Contact
FAQ

Dried Lemons, $3.59/pound

Organic dried lemons from our farm in California.
Lemons are harvested and sun-dried for maximum flavor.
Good in soups and on popcorn.

The lemon, Citrus Limon (l.) Osbeck, is a species of small evergreen tree in the flowering plant family rutaceae.
The tree's ellipsoidal yellow fruit is used for culinary and non-culinary purposes throughout the world, primarily for its juice, which has both culinary and cleaning uses. The juice of the lemon is about 5% to 6% citric acid, with a ph of around 2.2, giving it a sour taste.



Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Curabitur in tempus quam. In mollis et ante at consectetur.
Aliquam erat volutpat.
Donec at lacinia est.
Duis semper, magna tempor interdum suscipit, ante elit molestie urna, eget efficitur risus nunc ac elit.
Fusce quis blandit lectus.
Mauris at mauris a turpis tristique lacinia at nec ante.
Aenean in scelerisque tellus, a efficitur ipsum.
Integer justo enim, ornare vitae sem non, mollis fermentum lectus.
Mauris ultrices nisl at libero porta sodales in ac orci.

```
function Ball(r) {
  this.radius = r;
  this.area = pi * r ** 2;
  this.show = function(){
    drawCircle(r);
  }
}
```

Retain:
- Sentences with terminal punctuation marks
- Pages with at least 5 sentences, sentences with at least 3 words

Slide adapted from Colin Raffel

Pre-training Data: Experiment

- Takeaway:
  - Clean and compact data is better than large, but noisy data.
  - Pre-training on in-domain data helps.

Data set	Size	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ C4	745GB	83.28	**19.24**	80.88	71.36	**26.98**	**39.82**	**27.65**
C4, unfiltered	6.1TB	81.46	19.14	78.78	68.04	26.55	39.34	27.21

Pre-training Data Duplicates

- There is a non-negligible number of duplicates in any pre-training data.

	% train examples with dup in train	% train examples with dup in valid	% valid with dup in train
C4	3.04%	1.59%	4.60%
RealNews	13.63%	1.25%	14.35%
LM1B	4.86%	0.07%	4.92%
Wiki40B	0.39%	0.26%	0.72%

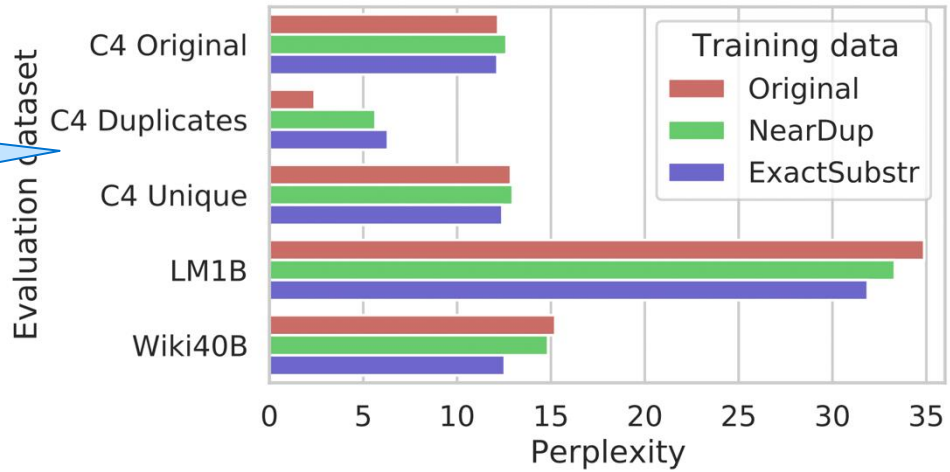
Dataset	Example	Near-Duplicate Example
Wiki-40B	\n_START_ARTICLE_\nHum Award for Most Impactful Character \n_START_SECTION_\nWinners and nominees\n_START_PARAGRAPH_\nIn the list below, winners are listed first in the colored row, followed by the other nominees. [...]	\n_START_ARTICLE_\nHum Award for Best Actor in a Negative Role \n_START_SECTION_\nWinners and nominees\n_START_PARAGRAPH_\nIn the list below, winners are listed first in the colored row, followed by the other nominees. [...]
LM1B	I left for California in 1979 and tracked Cleveland 's changes on trips back to visit my sisters .	I left for California in 1979 , and tracked Cleveland 's changes on trips back to visit my sisters .
C4	Affordable and convenient holiday flights take off from your departure country, "Canada". From May 2019 to October 2019, Condor flights to your dream destination will be roughly 6 a week! Book your Halifax (YHZ) - Basel (BSL) flight now, and look forward to your "Switzerland" destination!	Affordable and convenient holiday flights take off from your departure country, "USA". From April 2019 to October 2019, Condor flights to your dream destination will be roughly 7 a week! Book your Maui Kahului (OGG) - Dubrovnik (DBV) flight now, and look forward to your "Croatia" destination!

Deduplicating Data Improves LMs

- Models: GPT-2-like (1.5B param) models
- On there datasets:
  - C4 : the original training data
  - C4-NearDup: C4 excluding exact duplicates
  - C4-ExactSubs: C4 excluding near-duplicates

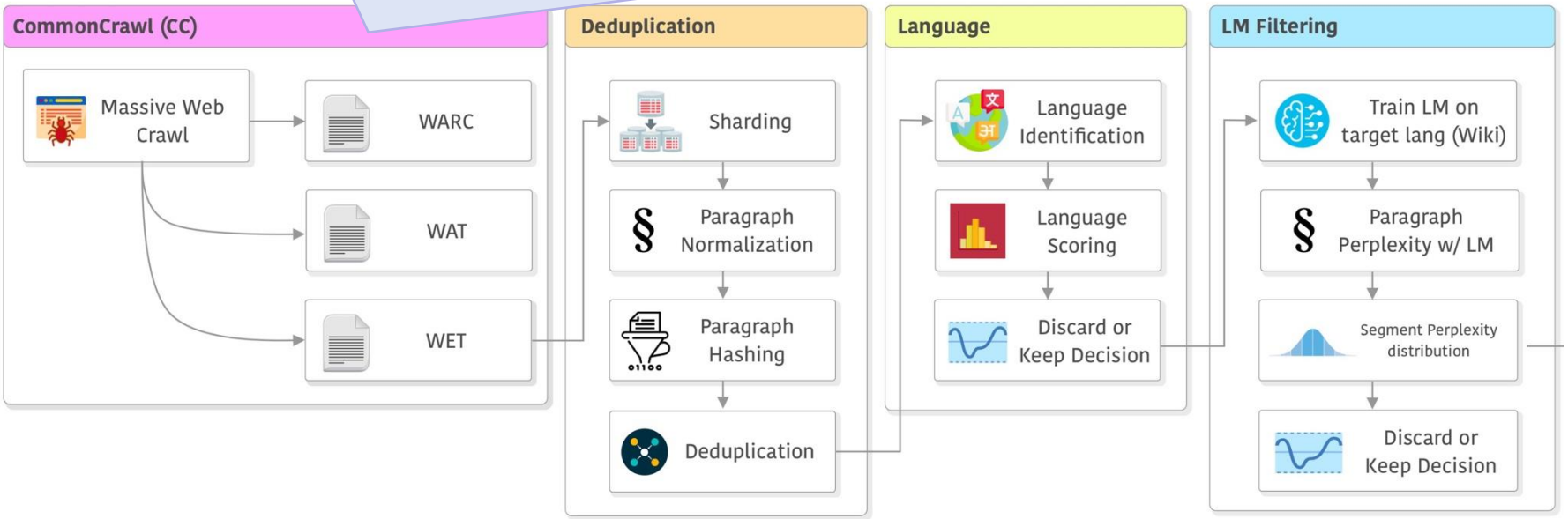
Except when evaluated on duplicate evaluation data!

Training on deduplicated data always leads to lower PPL!



LLaMA's Data Pipeline

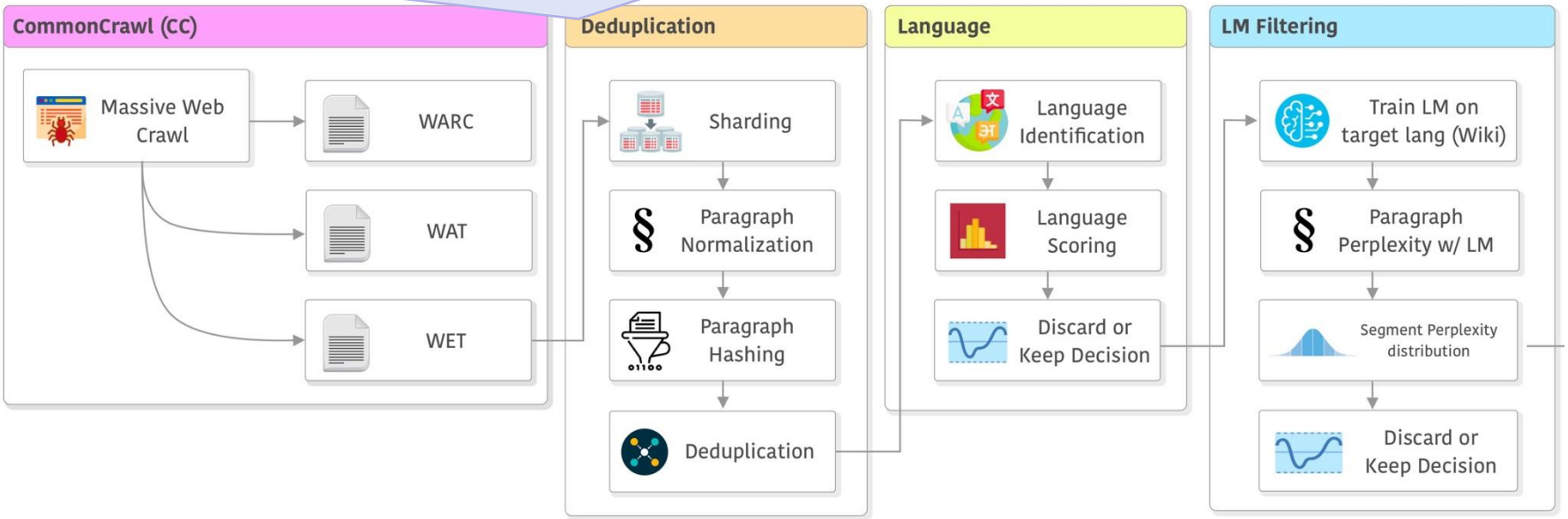
Starts with the massive crawled data by CommonCrawl.
The WET format that contains textual information.
WARC is raw, WAT is metadata, WET is text+some metadata.



https://twitter.com/tarantulae/status/1650170087708454913?t=ncWWY0FI0tYC_dYLs76r5g

LLaMA's Data Pipeline

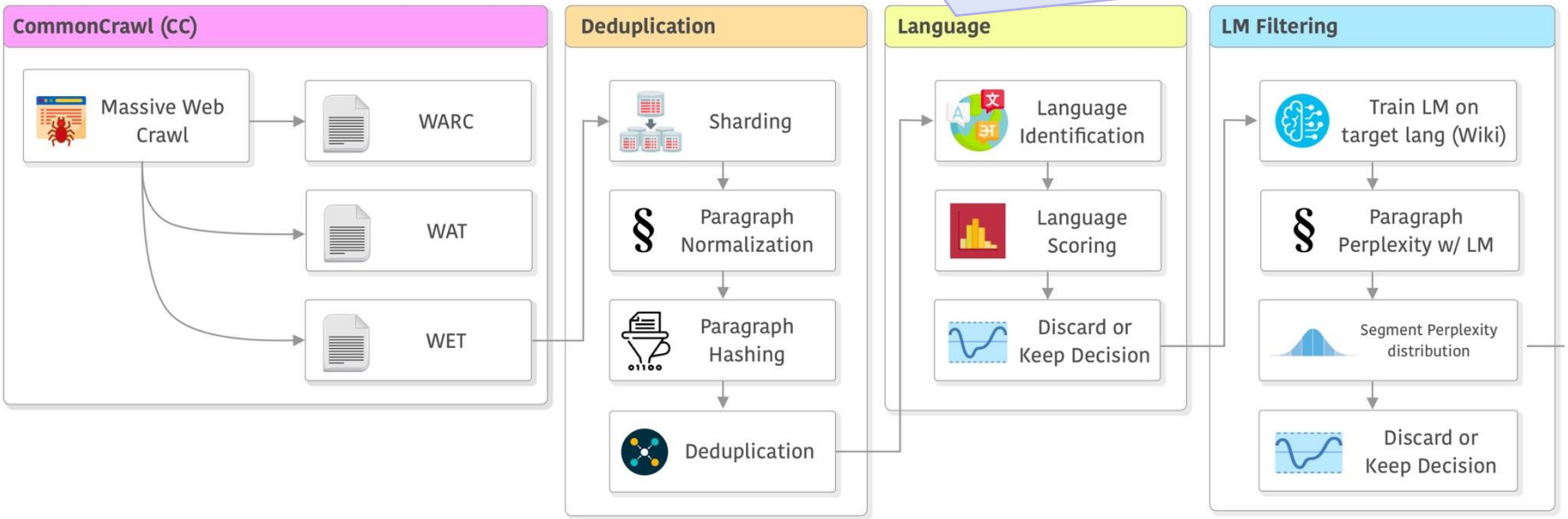
Shard WET content into shards of 5GB each (one CC snapshot can have 30TB). Then you normalize paragraphs (lowercasing, numbers as placeholders, etc), compute per-paragraph hashes and then duplicate them.



https://twitter.com/tarantulae/status/1650170087708454913?t=ncWWY0FI0tYC_dYLs76r5g

LLaMA's Data Pipeline

Perform language identification and decide whether to keep or discard languages.
The order of when you do this in the pipeline can impact the language discrimination quality.

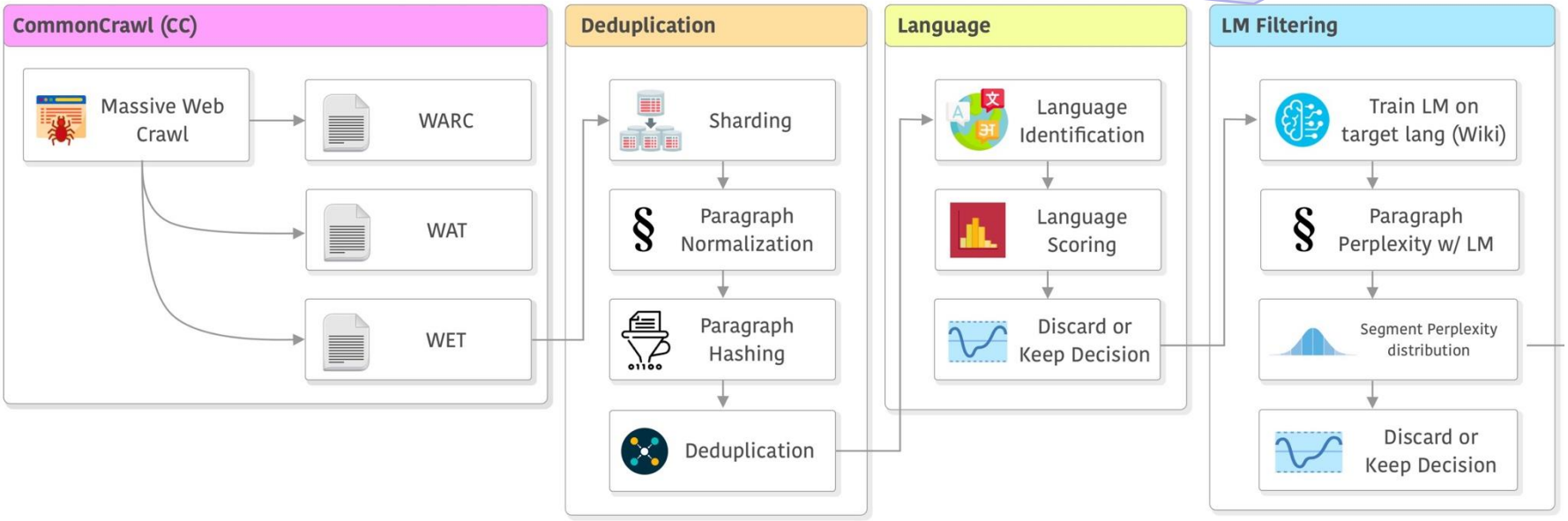


https://twitter.com/tarantulae/status/1650170087708454913?t=ncWWY0FI0tYC_dYLs76r5g

LLaMA's Data Pipeline

Do further quality filtering: Train a simple LM (n-gram) on target languages using Wikipedia, then compute per-paragraph perplexity on the rest of the data:

- Very high PPL: Very different than Wiki and likely low-quality → Drop
- Very low PPL: Very similar or near duplicates to Wiki → Drop



Grouped Query-Attention

- Used for training LLaMA 2.
- One key-value vector for each group of queries — an interpolation between "multi-head" attention and "multi-query" attention.

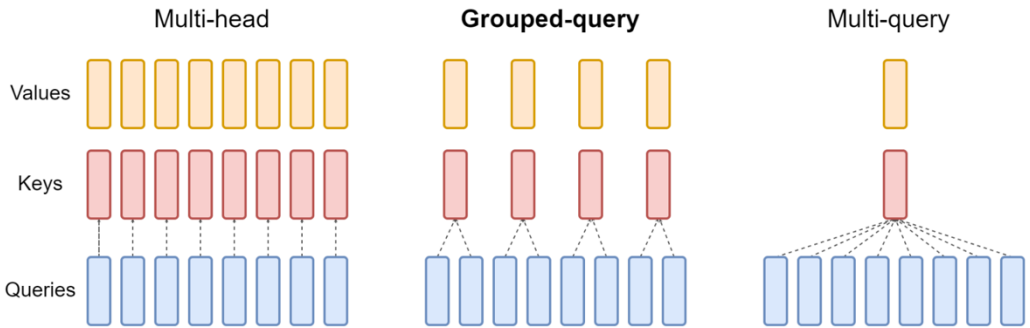


Figure 2: Overview of grouped-query method. Multi-head attention has H query, key, and value heads. Multi-query attention shares single key and value heads across all query heads. Grouped-query attention instead shares single key and value heads for each *group* of query heads, interpolating between multi-head and multi-query attention.

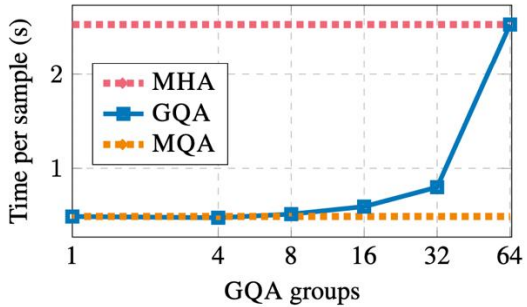
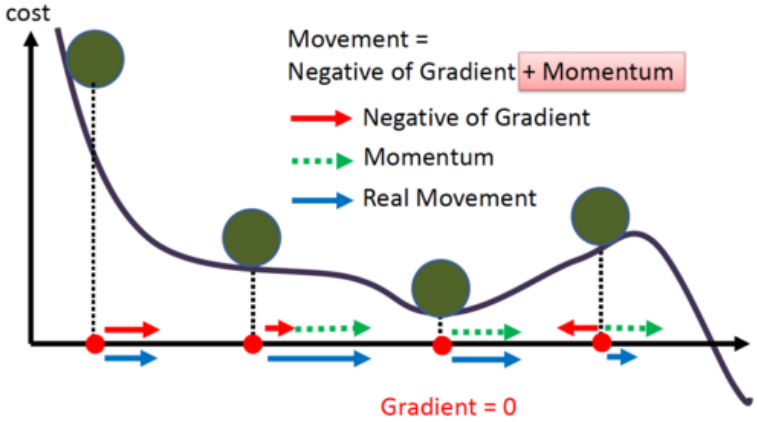


Figure 6: Time per sample for GQA-XXL as a function of the number of GQA groups with input length 2048 and output length 512. Going from 1 (MQA) to 8 groups adds modest inference overhead, with increasing cost to adding more groups.

- Improves inference scalability for our larger models

Optimizers

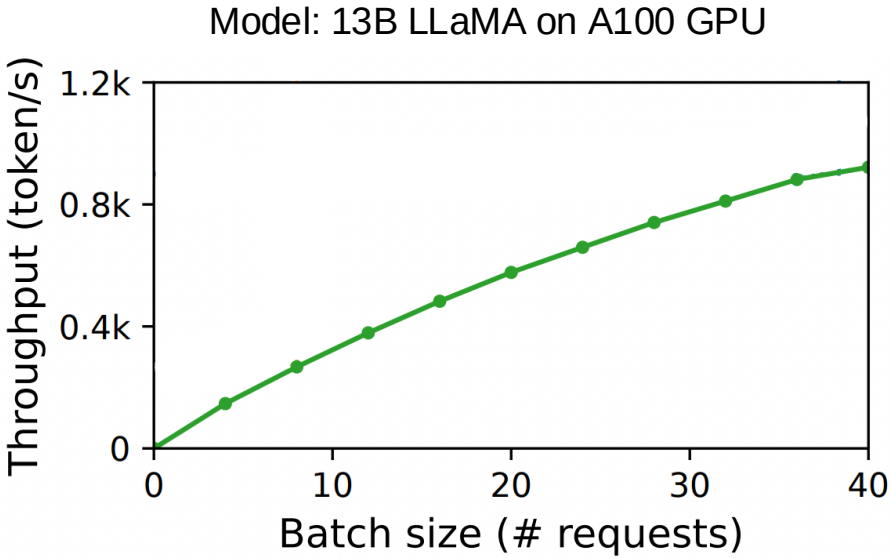
- Most modern models use "AdamW" optimizer (not vanilla Gradient Descent).
  - Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order "momentums".
  - "W" because it decouples "weight decay" from "learning rate". (Details out of scope for us. See the cited paper.)



https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html
https://pytorch.org/docs/stable/generated/torch.optim.Adam.html
[Decoupled Weight Decay Regularization, 2017]

Batching Data

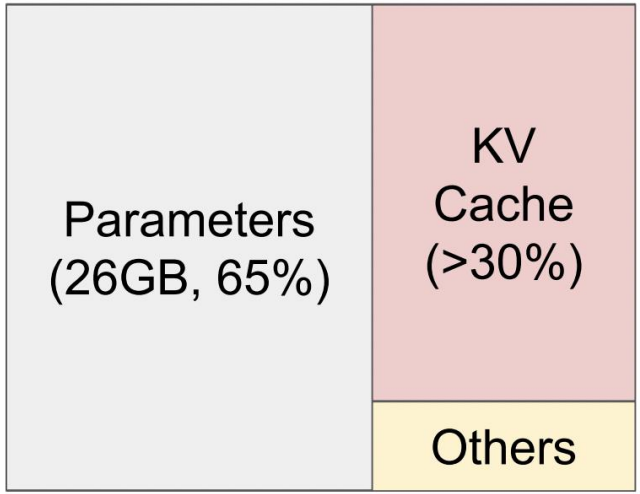
- Previously we talked about the importance of batching data
- GPUs are faster at Tensor operations and hence, we want to do batch processing
- The lager batch of data, the faster they get processed.
- Alas, the speedup is often sub-linear (e.g., 2x larger batch leads to less than 2x speedup).



The Memory Usage

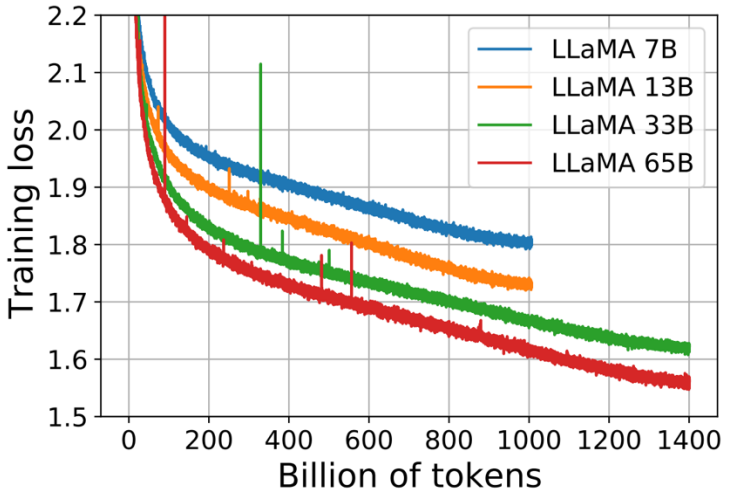
- Here is the memory usage of an NVIDIA A100 when serving (i.e., no training)
  - Model: 13B LLaMA
  - Batch size of 10

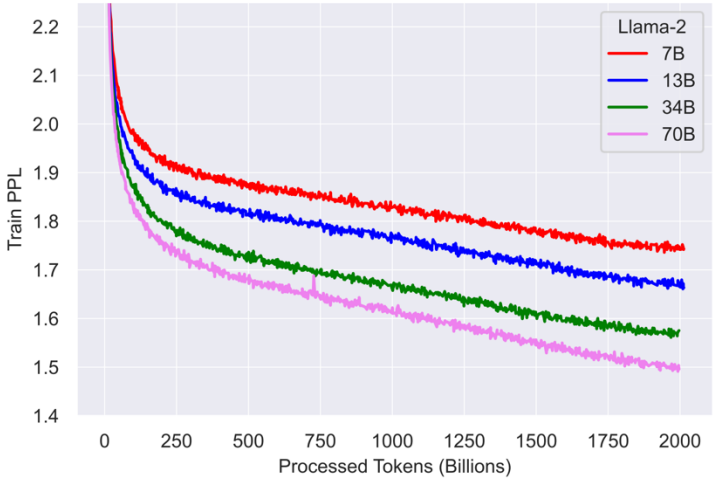
- ~65% of your GPU memory is the model parameters that never change
- ~32% of your memory are KV tensors that change for each input.
  - This KV cache will increase for larger batch sizes.
  - Managing this part of the memory is key for efficient training.



Convergence

- In practice, your model's loss should continue to go down with more training on more data.
- So, the real bottlenecks are:
  - (1) compute
  - (2) data
- Sometimes training diverges (spikes in the loss), at which point practitioners usually restart training from an earlier checkpoint.





Summary

- There is many empirical knowledge that goes into engineering LMs.
- Here we covered a basic topics about data and architecture engineering.
- Various topics are forthcoming: scaling laws, efficient training, etc.