

A slight-less-magical perspective into autoregressive language modeling

Count, compress and prune!

Kyunghyun Cho. NYU & Genentech.

Autoregressive language models

- A language model computes $\log p(s)$ given a sequence $s = (w_1, w_2, \dots, w_T)$
- An autoregressive language model computed it as

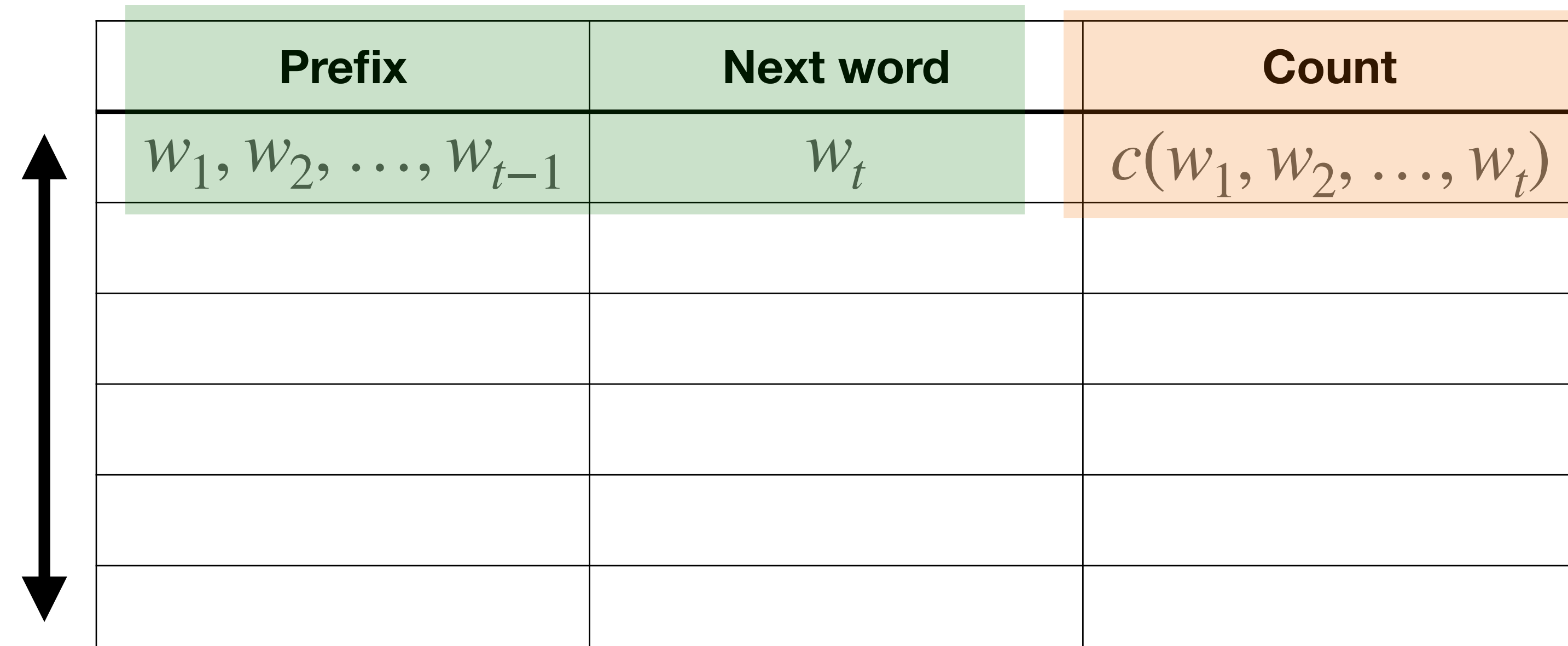
$$\log p(s) = \sum_{t=1}^T \log p(w_t | w_1, \dots, w_{t-1})$$

- This is often referred to as next-word prediction
 - Given a prefix $(w_1, w_2, \dots, w_{t-1})$, compute the probability of the next word w_t

Autoregressive language models

Count!

- Build a table of all possible **prefixes** and their **counts** from the training corpus



The diagram shows a table with three columns: 'Prefix', 'Next word', and 'Count'. The first row is highlighted with green for the first two columns and orange for the third. The first row contains the mathematical expressions w_1, w_2, \dots, w_{t-1} , w_t , and $c(w_1, w_2, \dots, w_t)$ respectively. To the left of the table, a vertical double-headed arrow indicates the height of the table, with the text $O(|D|)$ and $|D| = \text{corpus size}$ next to it.

Prefix	Next word	Count
w_1, w_2, \dots, w_{t-1}	w_t	$c(w_1, w_2, \dots, w_t)$

$O(|D|)$
 $|D| = \text{corpus size}$

Autoregressive language models


Retrieve!

- How to compute $p(w_t | w_1, w_2, \dots, w_{t-1})$:

- Retrieve all rows that match the prefix

- Divide the count of $(w_1, w_2, \dots, w_{t-1}, w_t)$ by the sum of counts

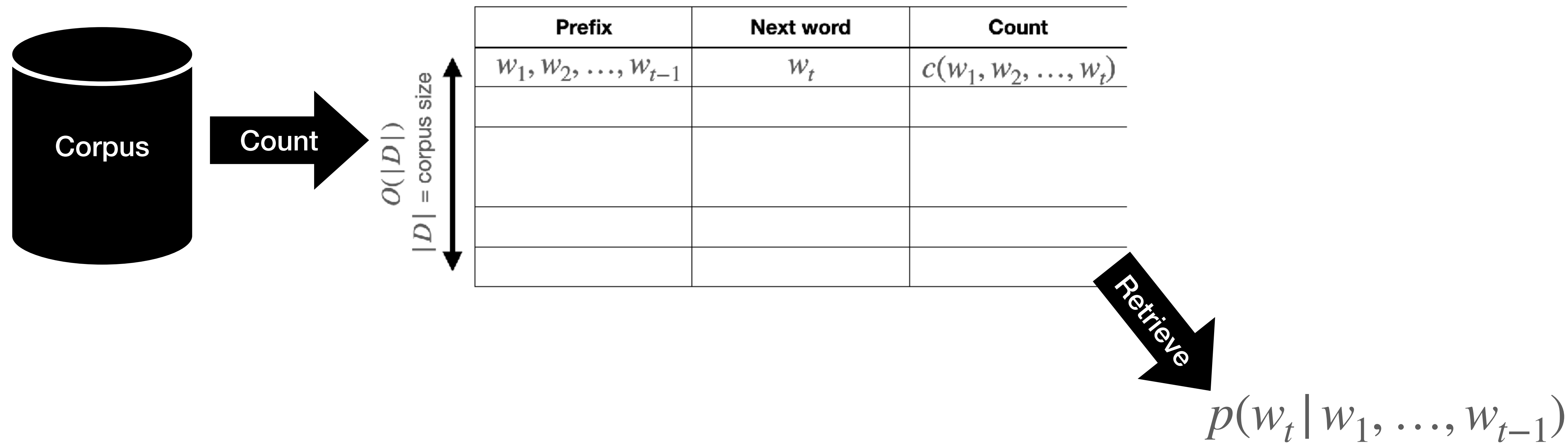
$$p(w_t | w_1, w_2, \dots, w_{t-1}) \approx \frac{c(w_1, w_2, \dots, w_{t-1}, w_t)}{\sum_{w \in V} c(w_1, w_2, \dots, w_{t-1}, w)}$$



Prefix	Next word	Count
w_1, w_2, \dots, w_{t-1}	w_t	$c(w_1, w_2, \dots, w_t)$

Language modeling in one picture

Count and retrieve!



Shannon's language models

- Known ever since 1950



2. ENTROPY CALCULATION FROM THE STATISTICS OF ENGLISH

One method of calculating the entropy H is by a series of approximations F_1, F_2, \dots , which successively take more and more of the statistics of the language into account and approach H as a limit. F_N may be called the N -gram entropy; it measures the amount of information or entropy due to statistics extending over N adjacent letters of text. F_N is given by¹

$$\begin{aligned} F_N &= - \sum_{i,j} p(b_i, j) \log_2 p_{b_i}(j) \\ &= - \sum_{i,j} p(b_i, j) \log_2 p(b_i, j) + \sum_i p(b_i) \log p(b_i) \end{aligned} \quad (1)$$

in which: b_i is a block of $N-1$ letters $[(N-1)\text{-gram}]$

j is an arbitrary letter following b_i

$p(b_i, j)$ is the probability of the N -gram b_i, j

$p_{b_i}(j)$ is the conditional probability of letter j after the block b_i ,

and is given by $p(b_i, j)/p(b_i)$.

The equation (1) can be interpreted as measuring the average uncertainty (conditional entropy) of the next letter j when the preceding $N-1$ letters are known. As N is increased, F_N includes longer and longer range statistics and the entropy, H , is given by the limiting value of F_N as $N \rightarrow \infty$:

$$H = \lim_{N \rightarrow \infty} F_N. \quad (2)$$

The N -gram entropies F_N for small values of N can be calculated from standard tables of letter, digram and trigram frequencies.² If spaces and punctuation are ignored we have a twenty-six letter alphabet and F_0 may be taken (by definition) to be $\log_2 26$, or 4.7 bits per letter. F_1 involves letter frequencies and is given by

$$[Shannon, 1950] \quad F_1 = - \sum_{i=1}^{26} p(i) \log_2 p(i) = 4.14 \text{ bits per letter.} \quad (3)$$

Learning is counting

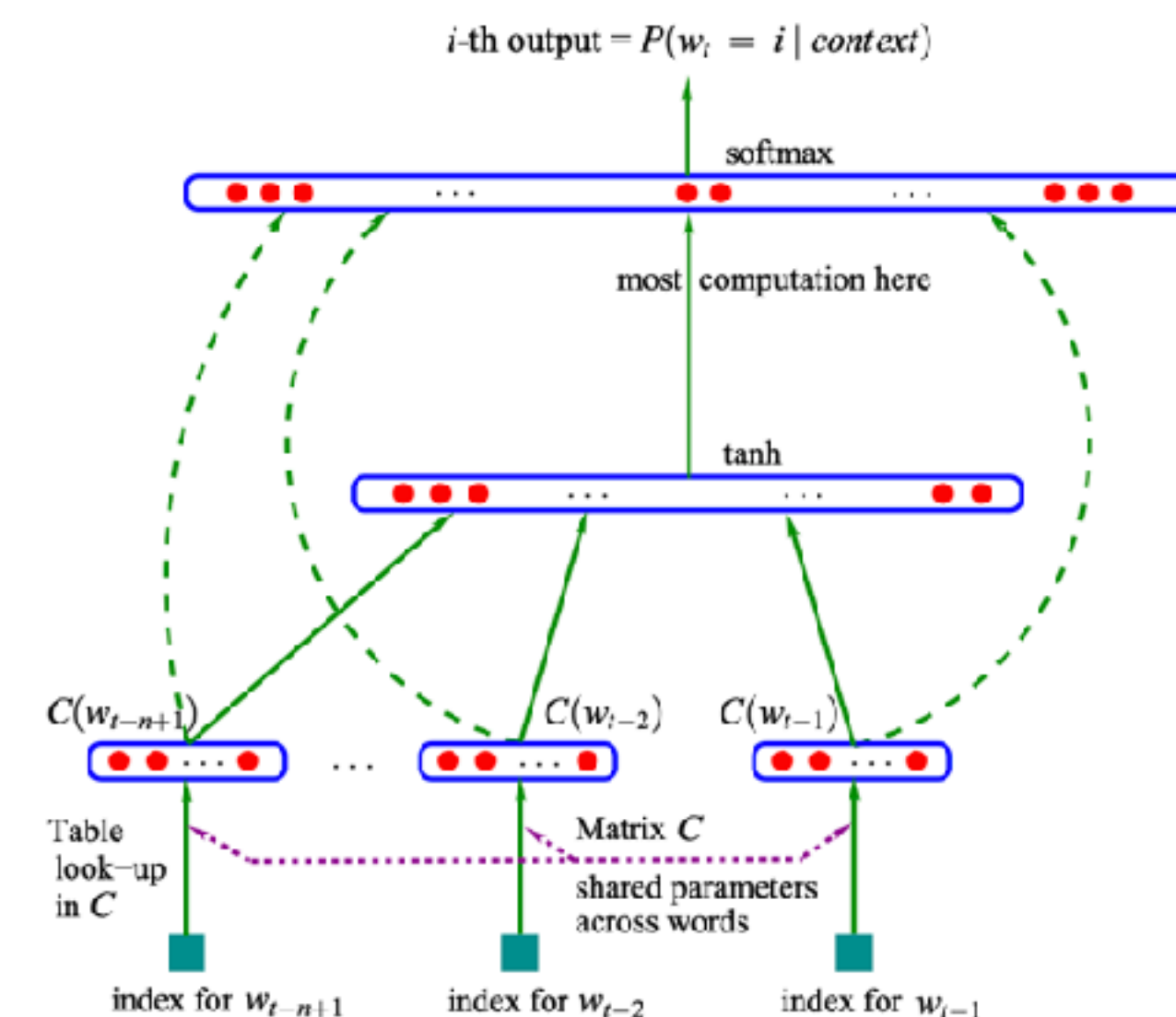
Bengio et al. (1999 & 2001)

- Instead of building a table explicitly, replace/approximate it with a neural network classifier by minimizing

$$-\mathbb{E}_{s \sim p_{\text{true}}(s)} \sum_{t=1}^{|s|} \mathbb{E}_{w_t \sim w_{1:t-1}} \log p_{\text{NN}}(w_t | w_{1:t-1})$$

- This is equivalent to minimizing the cross-entropy over all examples:

$$-\frac{1}{N} \sum_{s^n \in D} \sum_{t=1}^{|s^n|} \log p_{\text{NN}}(w_t^n | w_{1:t-1}^n)$$



Learning is counting

Bengio et al. (1999 & 2001)

- We replace/approximate a count table with a neural network by minimizing

$$-\mathbb{E}_{s \sim p_{\text{true}}(s)} \sum_{t=1}^{|s|} \mathbb{E}_{w_t \sim w_{1:t-1}} \log p_{\text{NN}}(w_t | w_{1:t-1})$$

- In other words, a neural language model *learns to count*
 - Nothing magical here ... but ...

Learning is *not* counting

Bengio et al. (1999 & 2001)

- Cross-entropy is upper-bound to entropy
- A neural language model puts high probabilities to sequences that are highly *improbable* under the count-based distribution.
- This enables generalization to unseen and rare prefixes (?)

$$-\sum_{y \in V} p^*(y|x) \log p(y|x; \theta) \geq -\sum_{y \in V} p^*(y|x) \log p^*(y|x),$$

because

$$\begin{aligned} \sum_{y \in V} p^*(y|x) \log p(y|x; \theta) &= \sum_{y \in V} p^*(y|x) \log \frac{p^*(y|x)}{p^*(y|x)} p(y|x; \theta) \\ &= \sum_{y \in V} p^*(y|x) \left(\log p^*(y|x) + \log \frac{p(y|x; \theta)}{p^*(y|x)} \right) \\ &= \sum_{y \in V} p^*(y|x) \log p^*(y|x) + \underbrace{\sum_{y \in V} p^*(y|x) \log \frac{p(y|x; \theta)}{p^*(y|x)}}_{=-\text{KL}(p^* \| p^\theta) \geq 0} \\ &\leq \sum_{y \in V} p^*(y|x) \log p^*(y|x). \end{aligned}$$

Learning is *not* counting

Count-based language models *cannot* generalize

- “a lion is chasing a llama” is impossible, because it does not appear in the corpus::

$$p(\text{"a lion is chasing a llama"}) = \left\{ \begin{array}{l} p(\text{"a"}) \times \\ p(\text{"lion"} \mid \text{"a"}) \times \\ p(\text{"is"} \mid \text{"a lion"}) \times \\ p(\text{"chasing"} \mid \text{"a lion is"}) \times \\ p(\text{"a"} \mid \text{"a lion is chasing"}) \times \\ \underbrace{p(\text{"llama"} \mid \text{"a lion is chasing a"})}_{=0} \end{array} \right. = 0$$

Learning is *not* counting

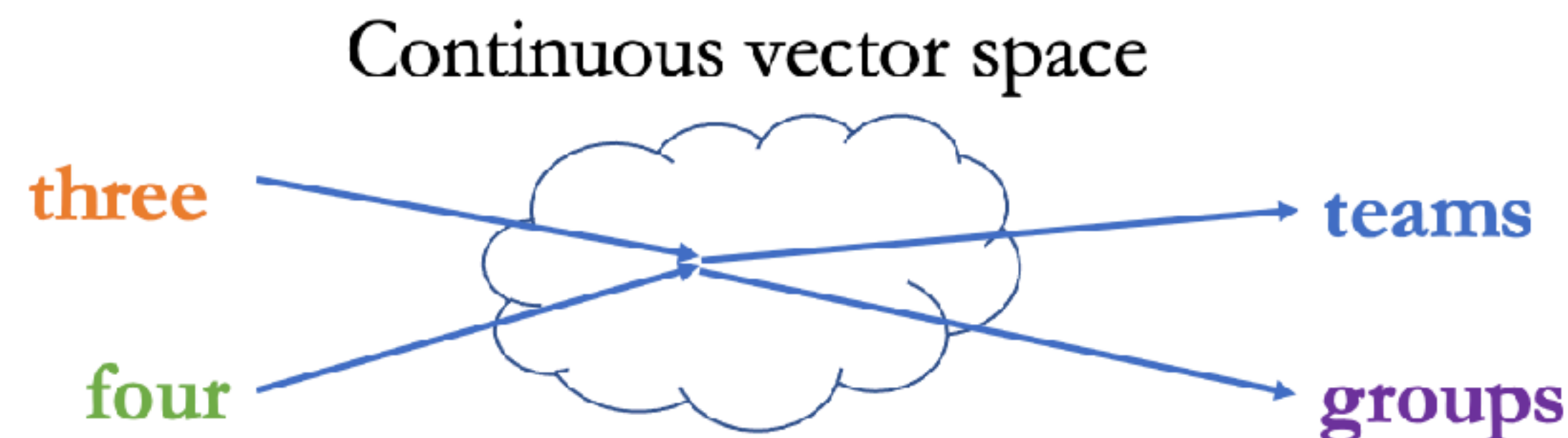
Neural language models *can* generalize

- We train a neural net with SGD and a lot of regularization techniques:
 - Neural nets memorize but also *compress*.
 - Similar inputs are clustered to save # of bits
 - Similar to brown clustering, latent semantic analysis, etc.
 - Similarity is implicitly imposed by similarity in the context (next words)

Learning is counting *and* compressing

Neural language models generalize by compressing *the # of rows*

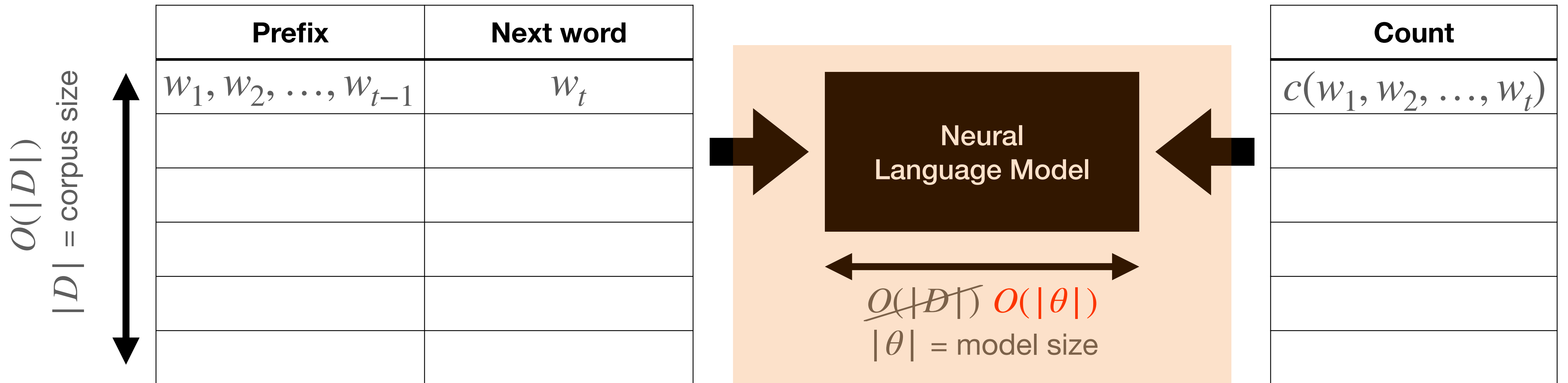
- A neural language model memorizes and compresses:
 - Training examples
 - there are **three** **teams** left for qualification.
 - **four** **teams** have passed the first round.
 - **four** **groups** are playing in the field.
 - Q: how likely is “groups” followed by “three”?



Learning is counting *and* compressing

Neural language models generalize by compressing *the # of rows*

- A neural language model *counts* and *compresses*:



- Still nothing seems too magical here ... **Is this where magic happens?**

How does learning count and compress?

Magic is nothing but a set of questions we haven't answered yet

- Two sides of magic
 1. **Compression:** assigning no probability to probable instances
 2. **Generalization:** assigning a probability to impossible instances
- **Probability recovered from data** transfers to unseen instances

Question 1: what do LM's choose to ignore?

- Compression often implies losing information (probability mass here)*
- Which subsets of training examples are lost in the process of compression?
- Compression in a learned system is known to *forget about non-canonical examples*.
- What does our language model choose to ignore and forget?



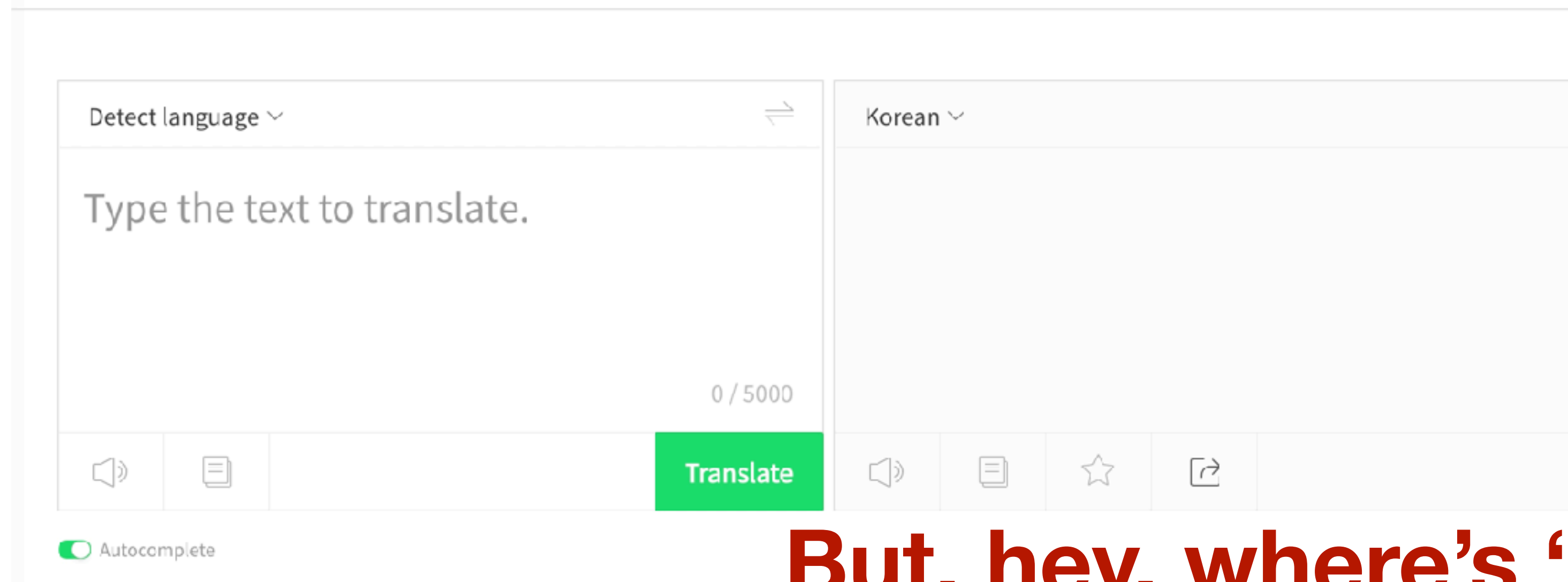
Hooker et al. (2021)

* lossy-less compression is definitely possible, and neural nets can be used for lossy-less compression, although our practice in training a language model does not guarantee it.

Question 2: what do LM's choose to prefer?

A very difficult question to even ask correctly

- Instead, we ask if there is any degeneracy in LM's generalization?
 - Yes, $p(\emptyset | X) \gg 0$!? [Kulikov et al. 2022]
 - Yes, $p(s | X) \gg p(s' | X)$ for a very unlikely X : *hallucination*



The image shows a screenshot of a web-based translation interface. On the left, there is a text input area with the placeholder text "Type the text to translate." and a character count "0 / 5000". Above the input area is a "Detect language" dropdown menu. To the right of the input area is a "Korean" dropdown menu. Below the input area is a green "Translate" button. At the bottom left, there is an "Autocomplete" toggle switch.

But, hey, where's "generation"?

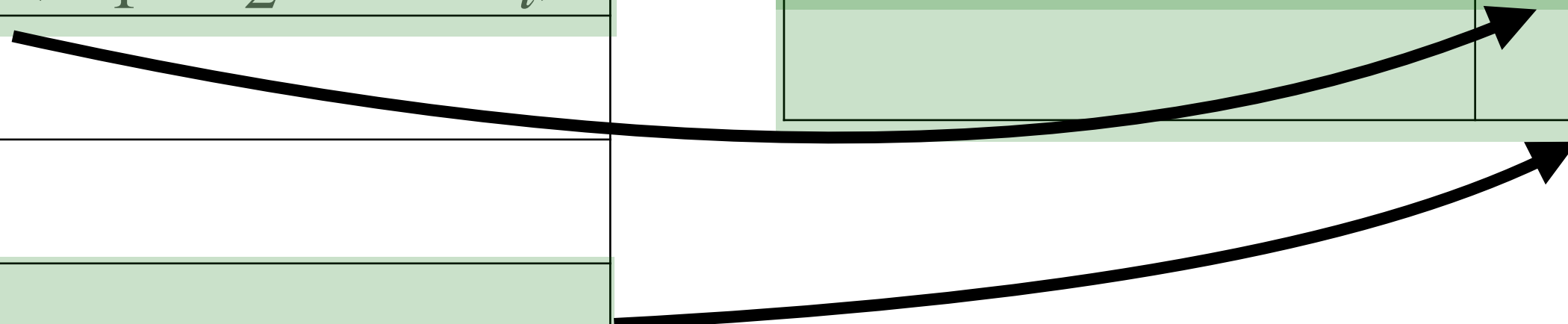
Generation

Language models are cool, because they produce realistic text

- Prompting, $p(w_{T+1}, w_{T+2}, \dots, w_{T'} | w_1, w_2, \dots, w_T)$, is equivalent to selecting a subset of rows from the original table.

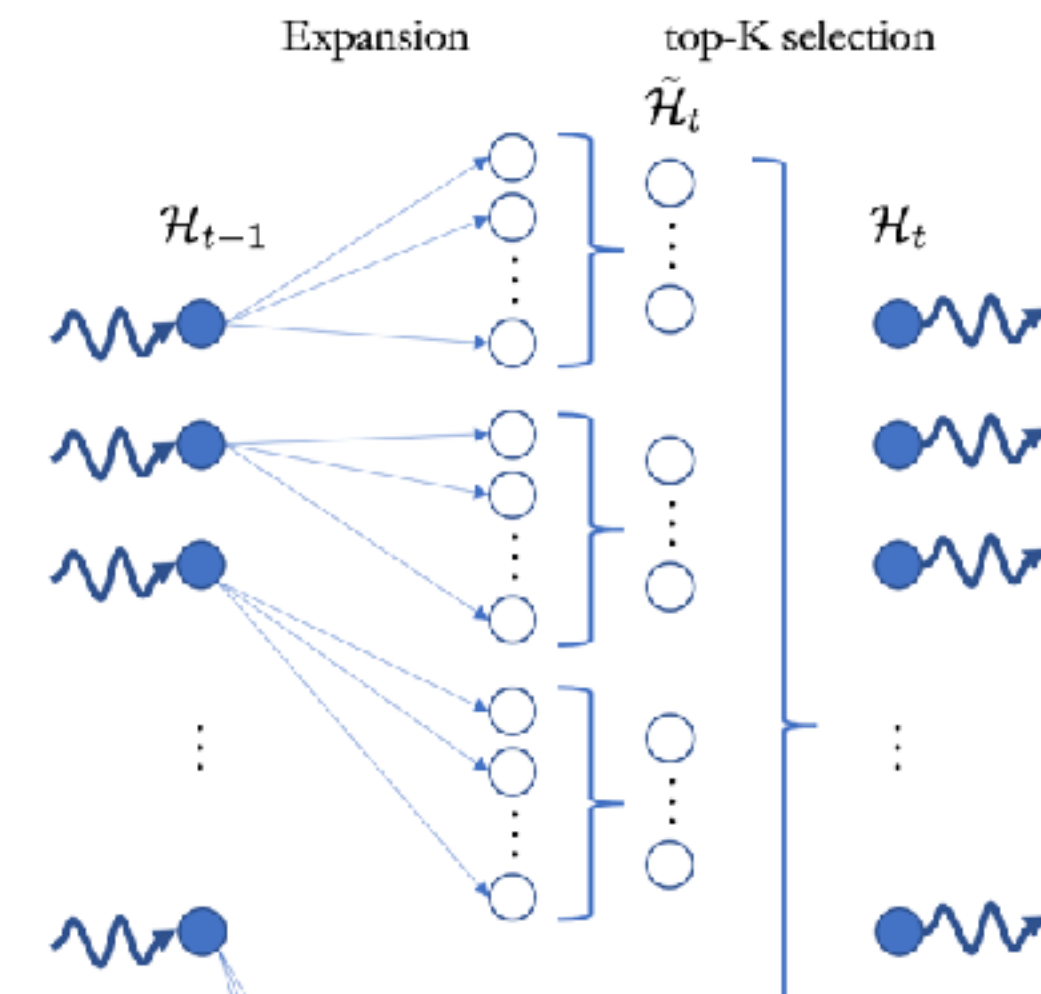
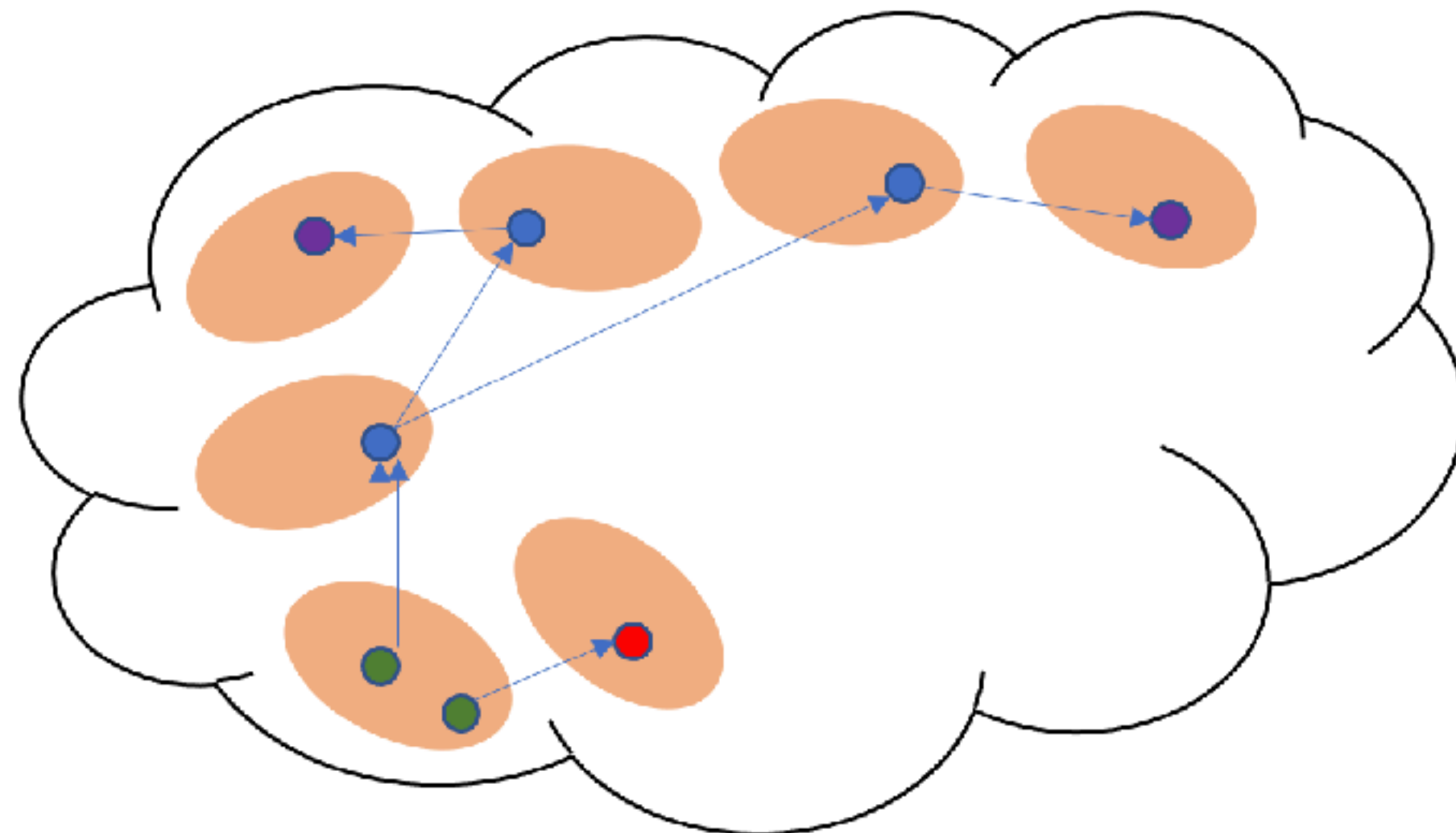
Prefix	Next word	Count
w_1, w_2, \dots, w_T	w_t	$c(w_1, w_2, \dots, w_t)$

Prefix	Next word	Count

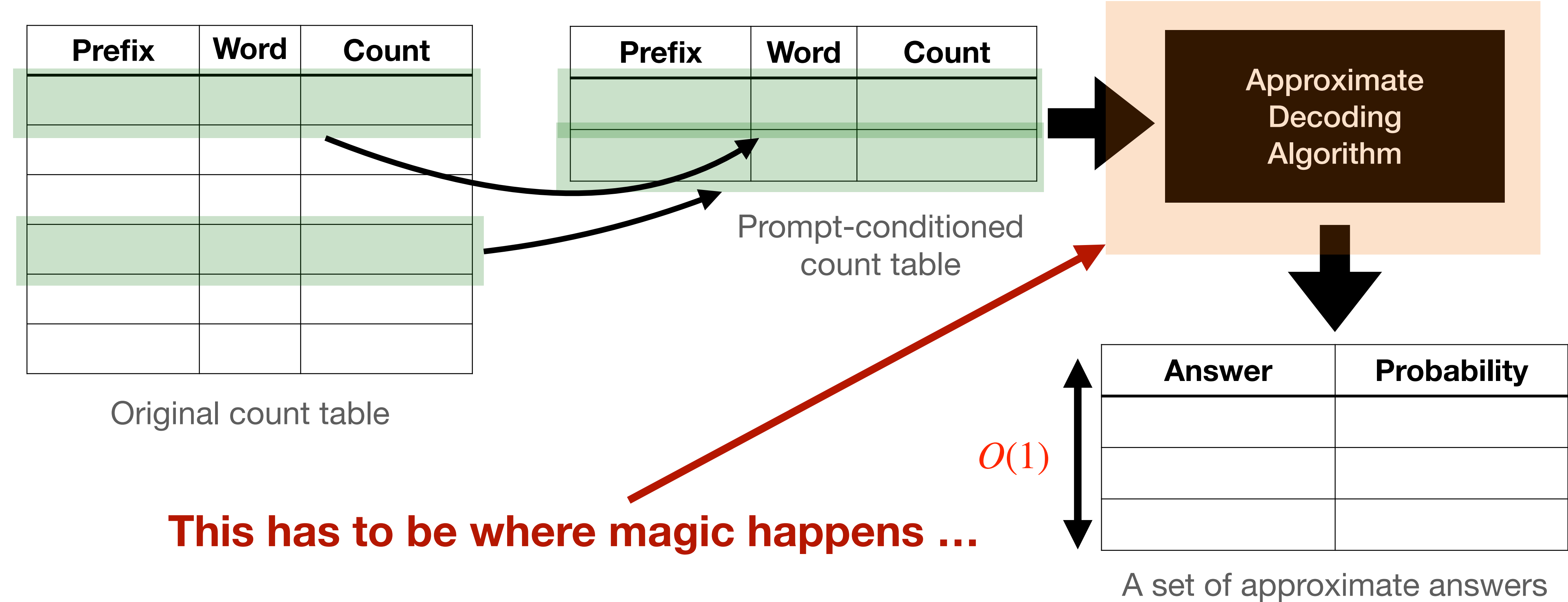


Generation requires pruning

- The number of rows in this sub-table is still too large: $O(e^{T'-T+1})$
- We must dramatically reduce the # of rows: $O(e^{T'-T+1}) \rightarrow O(1)$
 - This allows us to inspect (all) possible answers: necessary for controllability



Generation is pruning



Question 3: what do LM's prefer to generate?

Another very difficult question to even ask correctly

- Instead, we ask (again) if there is any degeneracy in LM's generation?

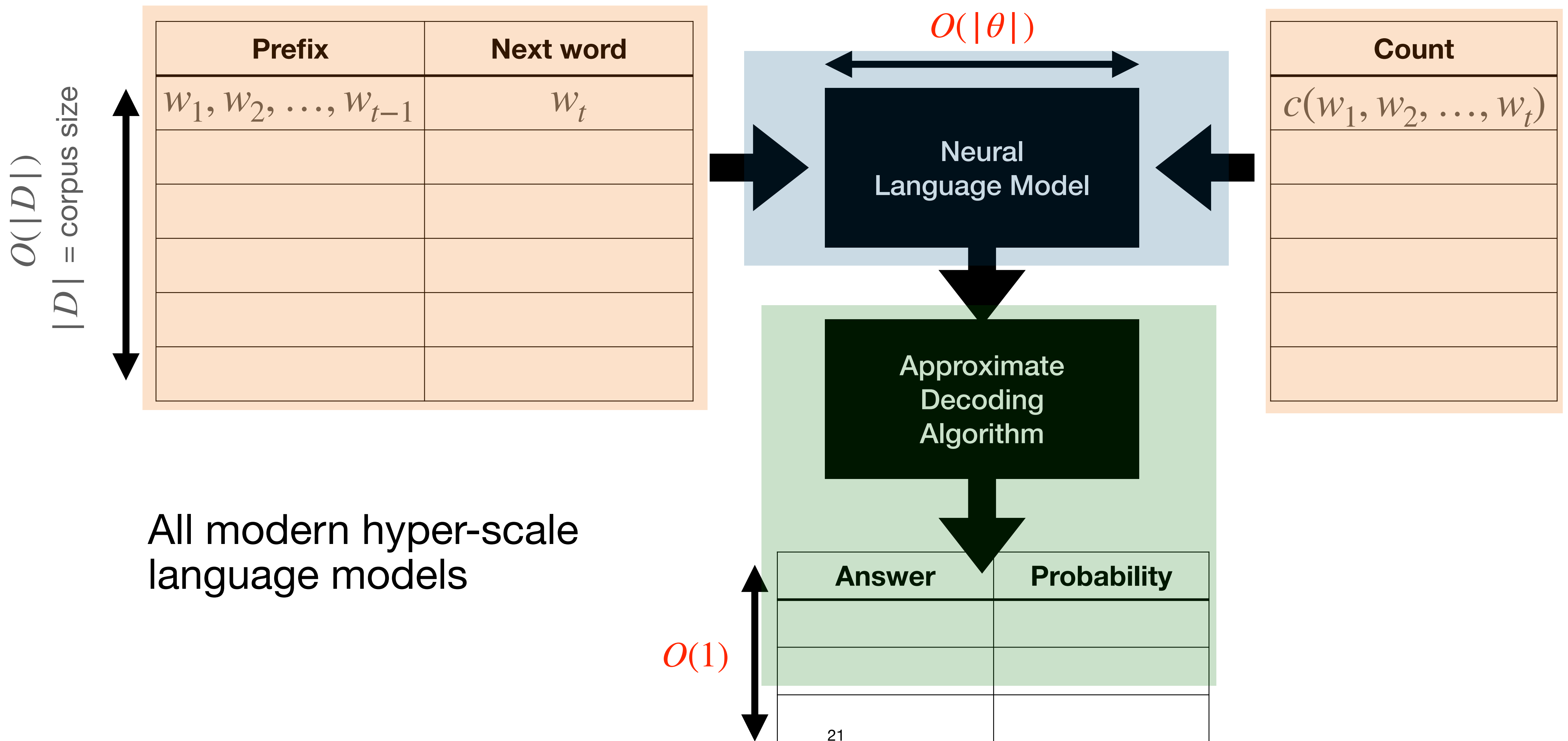
• Yes, $q(\underbrace{w_T, w_{T+1}, \dots}_{\infty} \mid w_1, \dots, w_T) > 0!$?

- See Holzmann et al. [2019] and Welleck et al. [2020, 2021] for more analysis.

```
response = openai.Completion.create(engine="davinci", prompt="no barber can shave their own head. i am a barber. i shave", max_tokens=100, top_p=.5)
print(response["choices"][0]["text"])
```

everybody's head. i have a little mirror on the wall. i look at myself in the mirror. i shave my head. i am a barber. i shave everybody's head. i have a little mirror on the wall. i look at myself in the mirror. i shave my head. i am a barber. i shave everybody's head. i have a little mirror on the wall. i look at myself in the mirror

Count, compress and prune



Count, compress and prune

Modern hyper-scale LM's do wonders but are limited ...

- Simple principles behind modern hyper-scale language models
 - The same set of principles hasn't changed much ever since 50's and 90's
 - A lot of seeming magic happen within deep learning: SGD, attention, etc.
- Obvious questions have not been asked nor answered properly
 - Memorization, hallucination, length bias, inconsistent generation, etc.
- More work, beyond scaling, is needed to move beyond the current paradigm

Count, compress and prune

“The real lesson is that whatever works the best today is beatable by something we have to still invent and to invent it we actually have to try to work on something new.”



Tomas Mikolov (2021, Thesis Review Podcast)