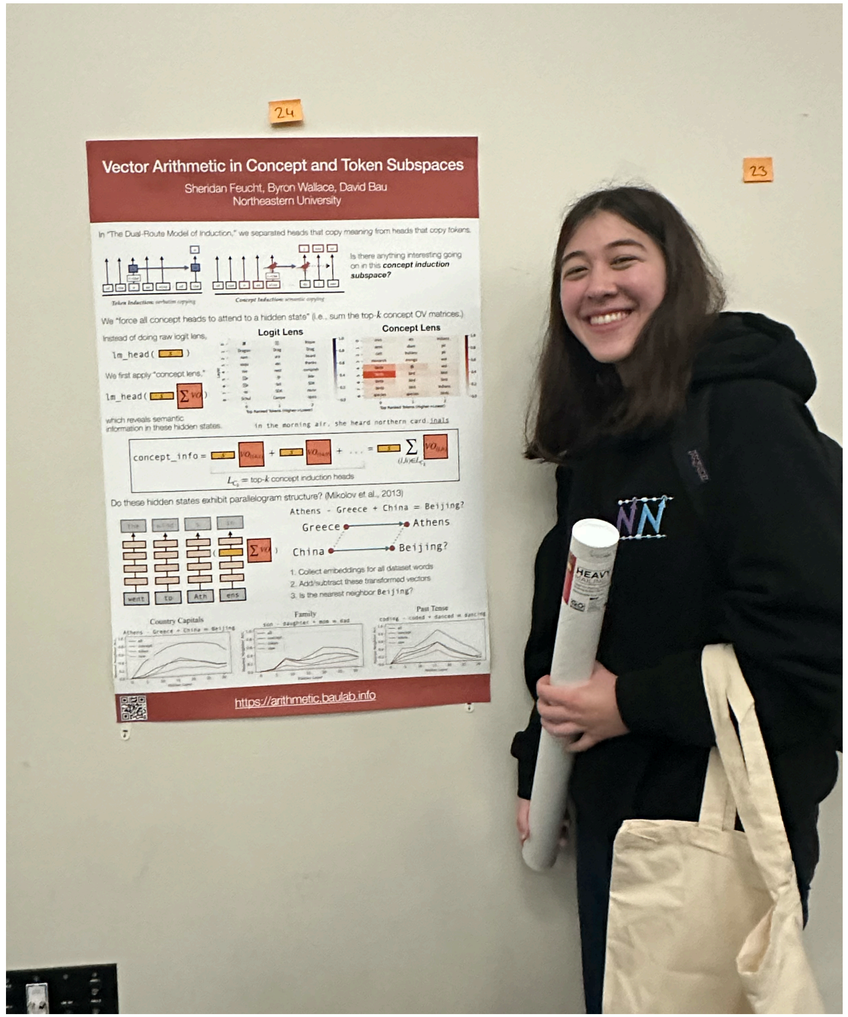
Concept Heads for Vector Arithmetic

January 21, 2026 - Sheridan Feucht

Simplex Group Meeting

Personal Introduction

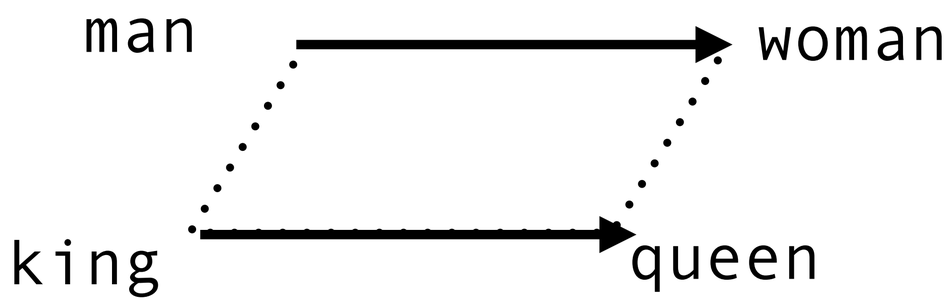


- Sheridan F. (they/them)
- 3rd-year PhD student at Northeastern, advised by David Bau and Byron Wallace
- Visiting Goodfire as a Research Fellow for 3 months
- General interests: how do LLMs represent language and linguistic “concepts”?
- Please interrupt with comments/questions! 🗣️

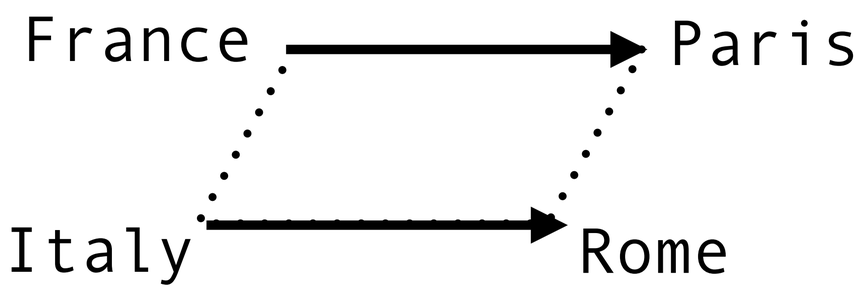
word2vec Arithmetic

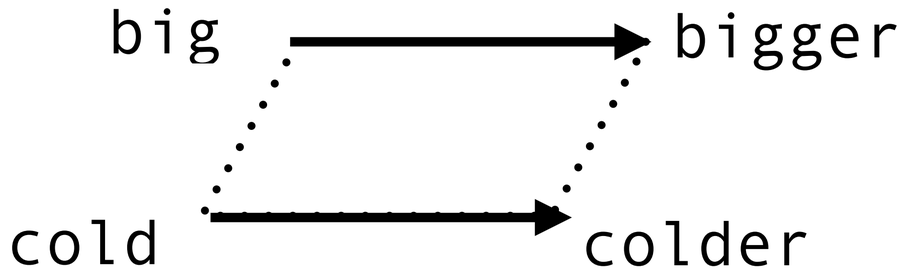
Mikolov et al. (2013)

```
woman - man + king = queen
```



Is this true for LLM hidden states?





Use Token Embeddings?

Could just use the input token embeddings…

```
(France - Paris) + Rome = Italy
```

We can't *just* look at token embeddings, since so many words are composed of multiple tokens!

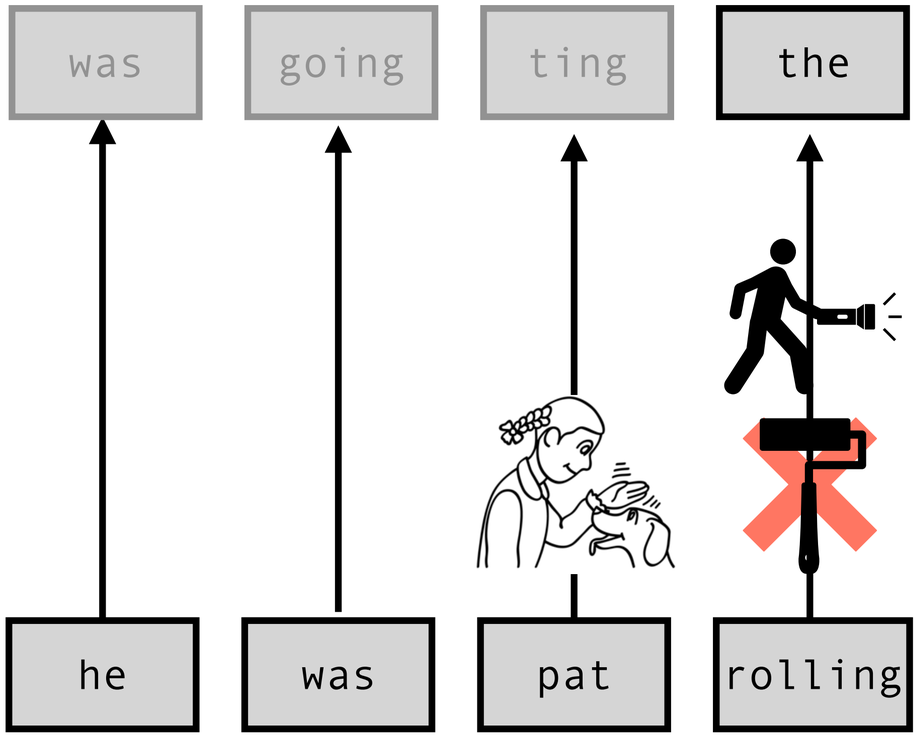
```
'northeastern' —>  ['n', ' ort', ' he', 'astern']

'patrolling' —>    ['pat', 'rolling']
```

Even if we had a tokenizer with every possible word, we'd still have multi-*word* concepts.

```
'New York City' —>  ['New,' 'York', 'City']
```

Mapping Tokens to Words

Detokenization - “What words am I working with here?”



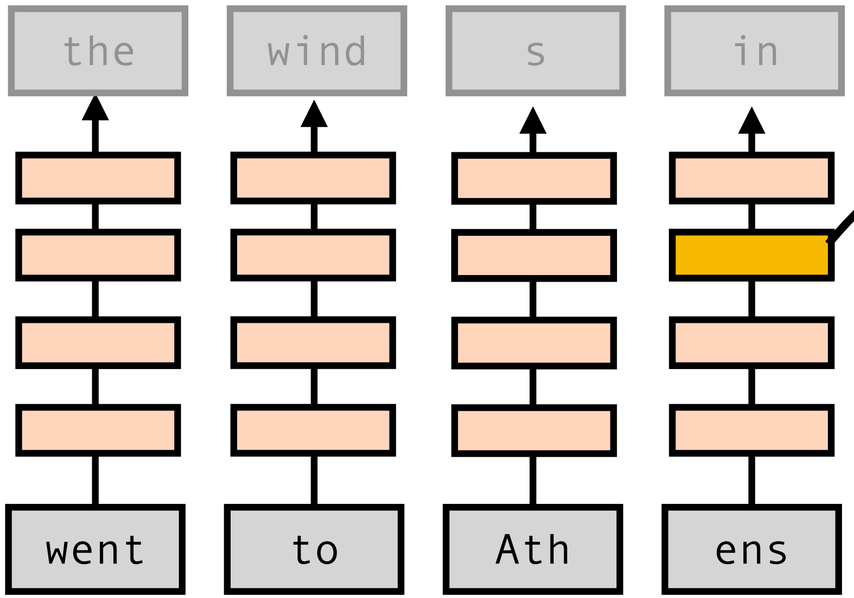
Models seem to “write in” semantic information about multi-token chunks at the **last token position** of that word.

Can we look there for “word embeddings?”

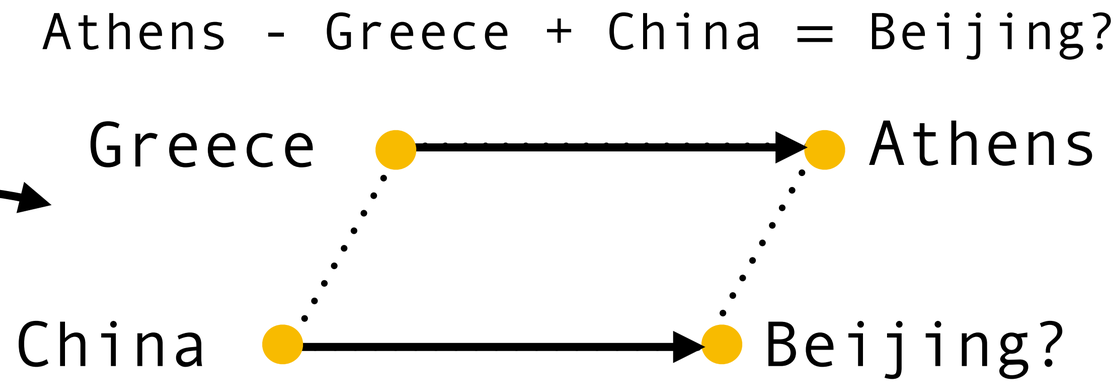
Gurnee et al. (2023) - Finding Neurons in a Haystack: Case Studies with Sparse Probing

Feucht et al. (2024) - Token Erasure as a Footprint of Lexical Items in LLMs

First Attempt



(1) Run all words through some template and extract raw hidden states

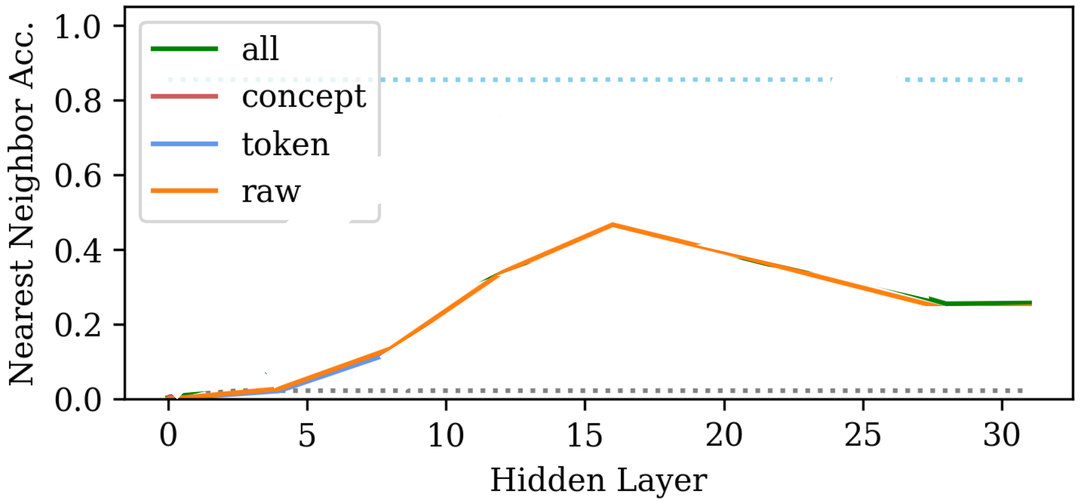


(2) Add/subtract these raw hidden state vectors

(3) Check if the nearest neighbor (among all the countries/capitals) is correct.

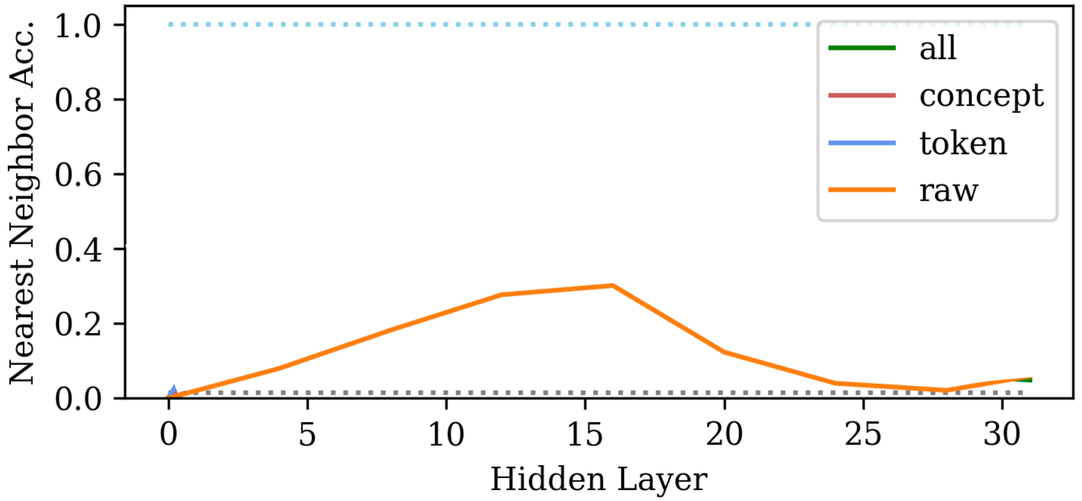
Country Capitals

`Athens - Greece + China = Beijing`



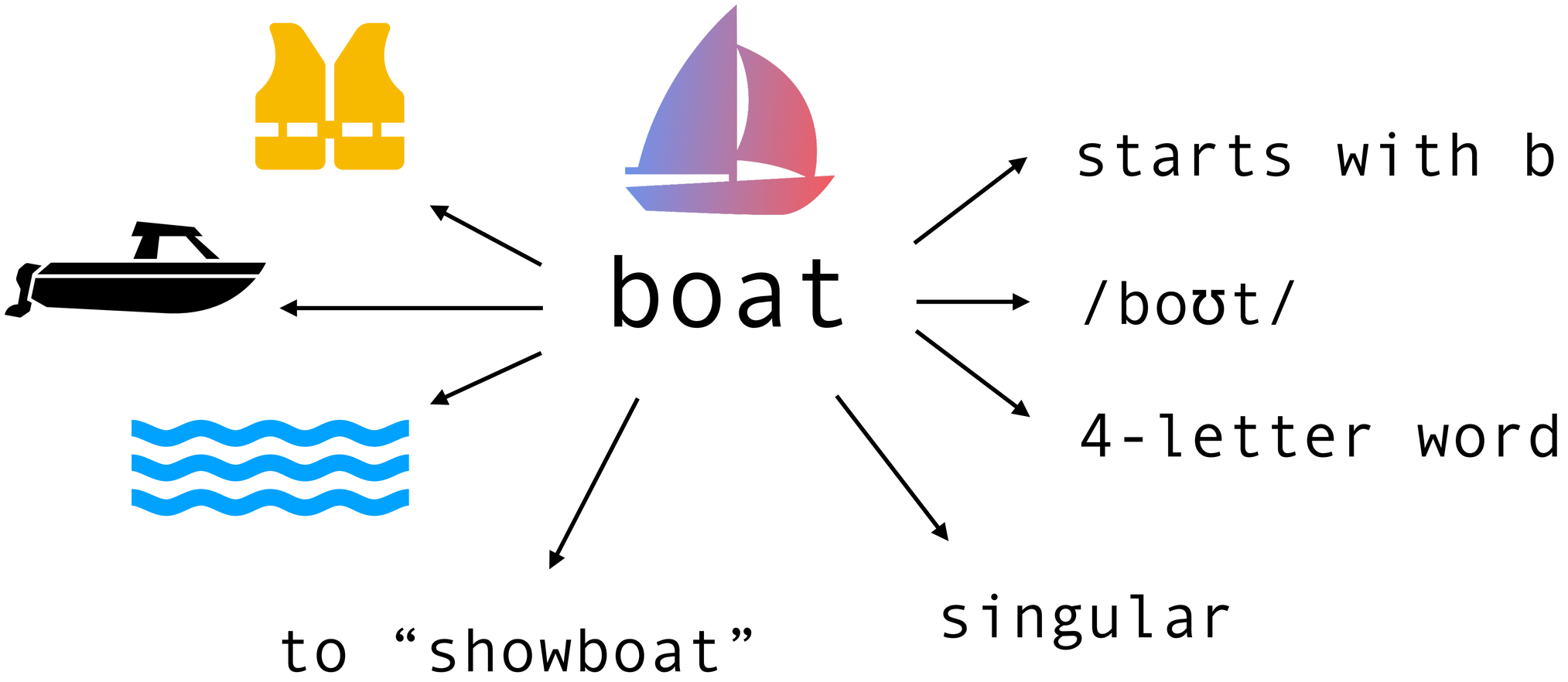
Present Participle

`code - coding + dancing = dance`



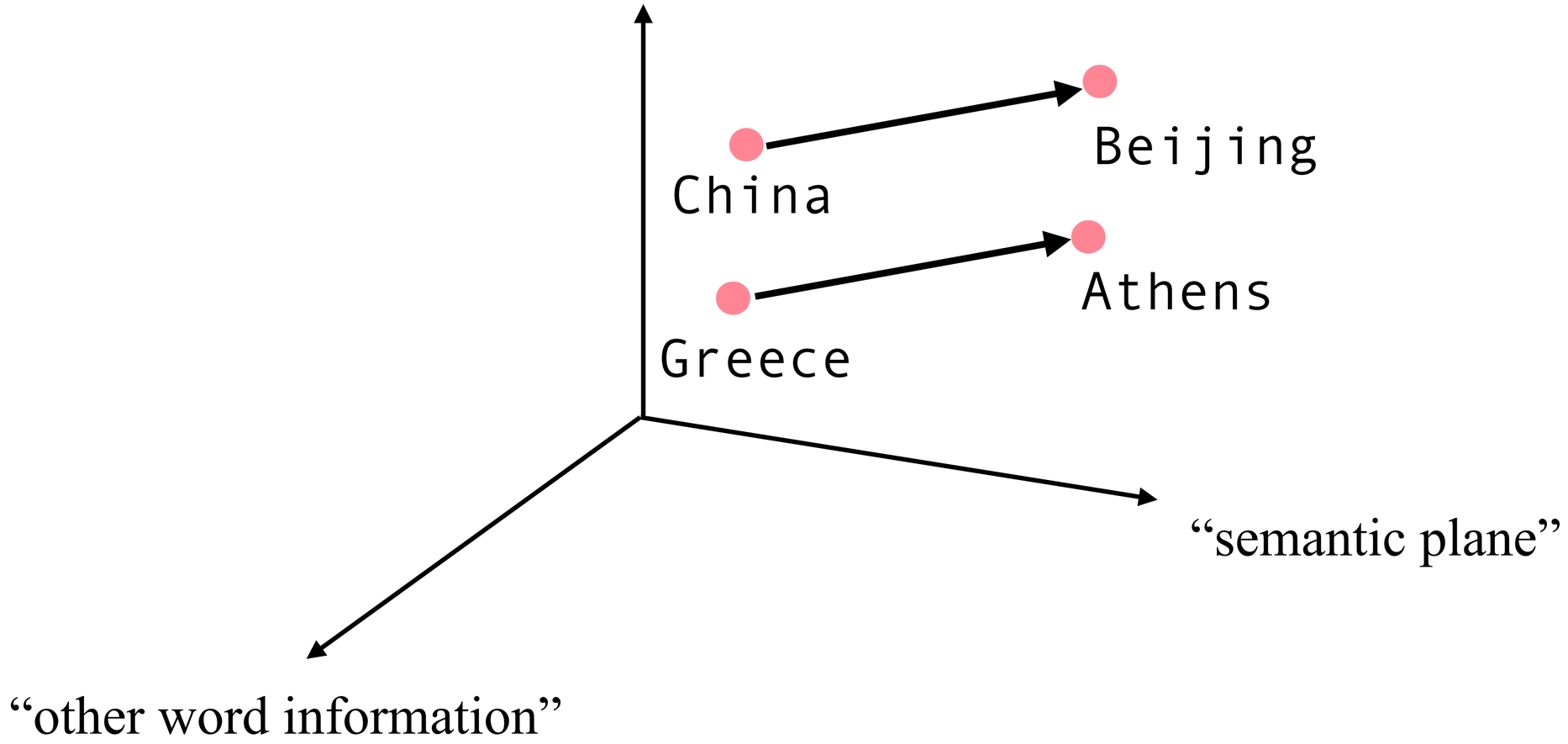
This works pretty okay actually, but can we do better?

Word Representations are Rich

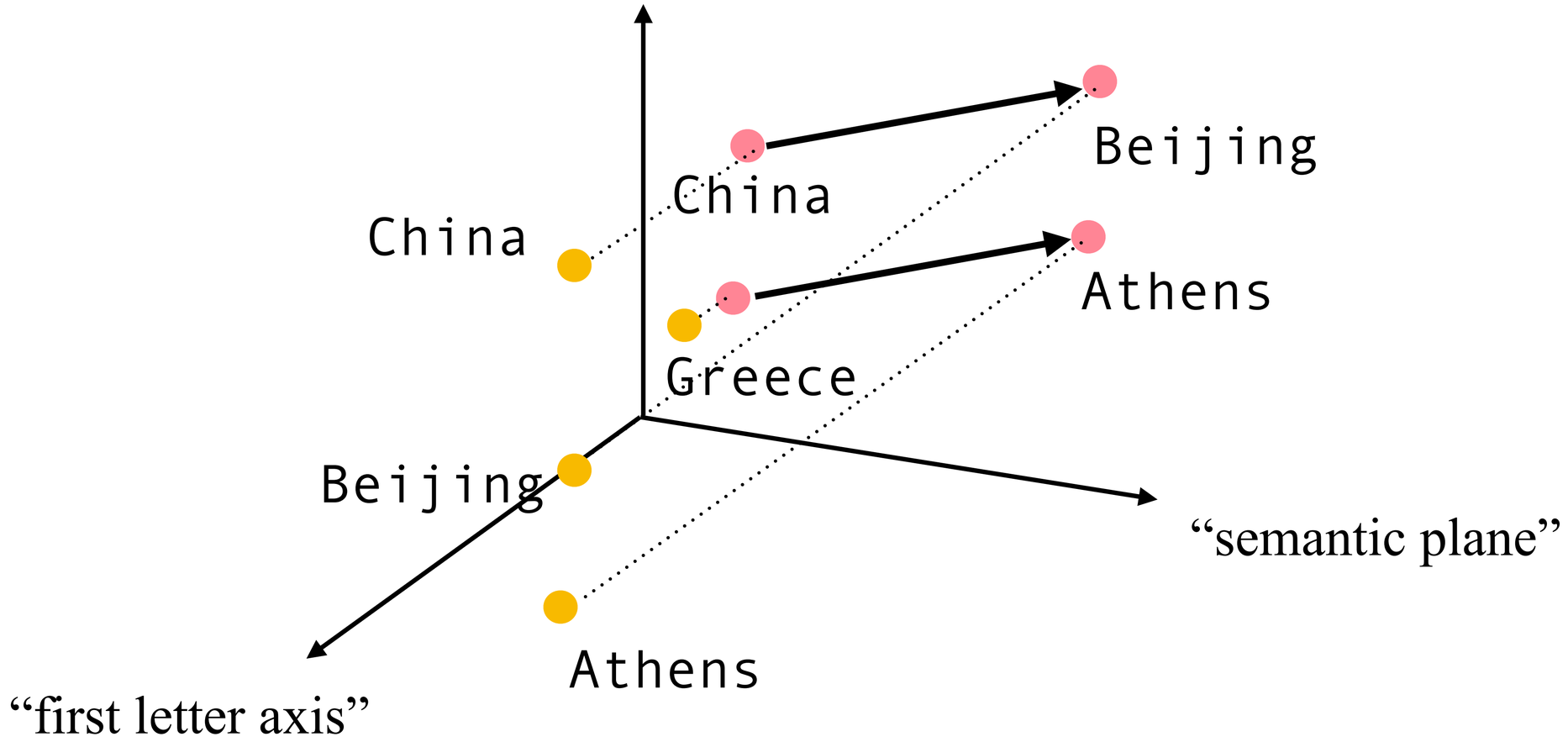


McLaughlin et al. (2025) - I Have No Mouth, and I Must Rhyme: Uncovering Internal Phonetic Representations in Llama 3.2

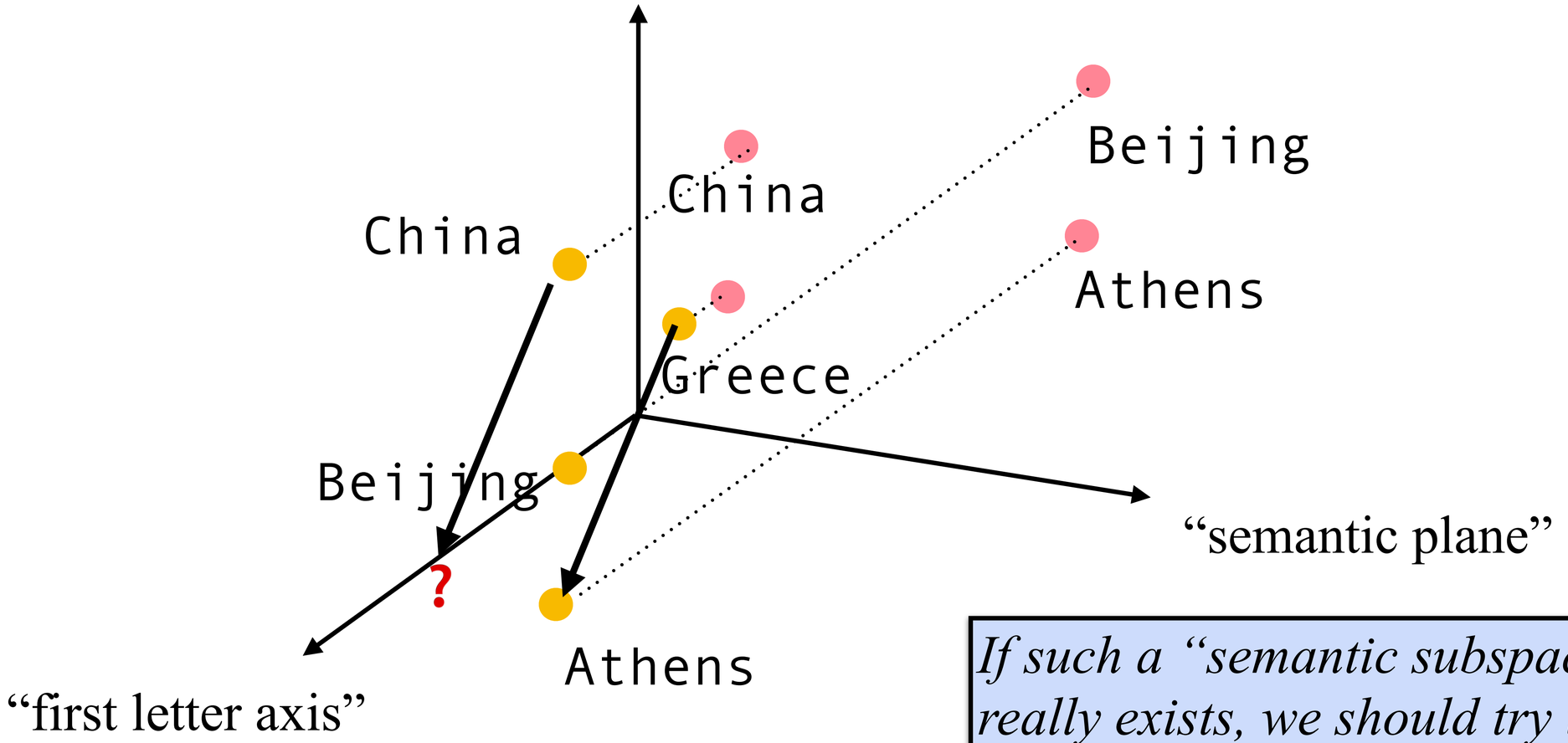
Word Representations are Rich



Word Representations are Rich



Word Representations are Rich



If such a "semantic subspace" really exists, we should try to project onto that space first.

The Dual-Route Model of Induction

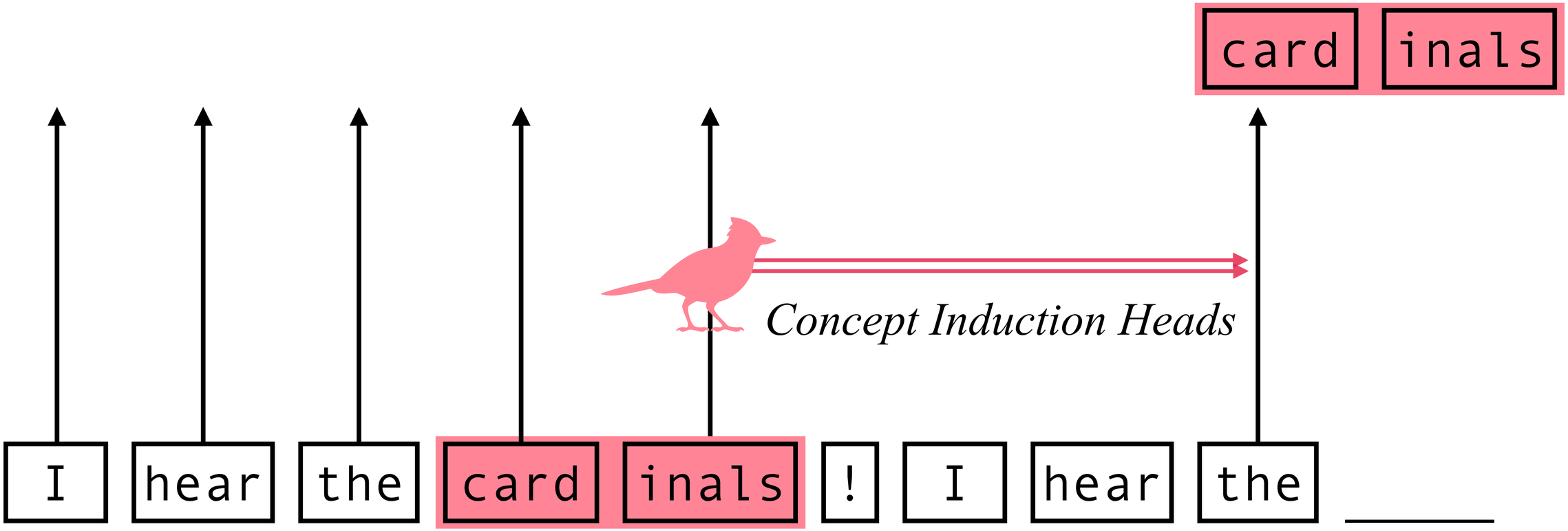
Sheridan Feucht, Eric Todd, Byron Wallace, & David Bau *
Northeastern University
{feucht.s,todd.er,b.wallace,d.bau}@northeastern.edu

Let's try using "concept heads" from previous work!

The Dual-Route Model of Induction

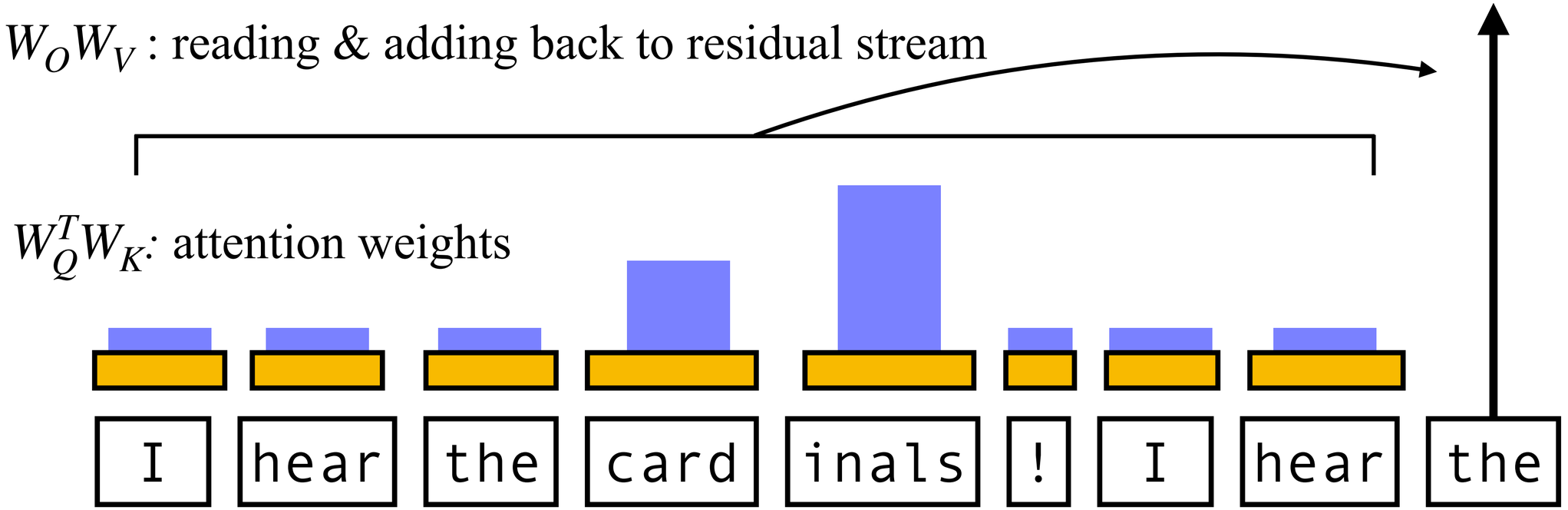
Sheridan Feucht, Eric Todd, Byron Wallace, & David Bau *
Northeastern University
{feucht.s,todd.er,b.wallace,d.bau}@northeastern.edu

Let's try using "concept heads" from previous work!



Concept Lens

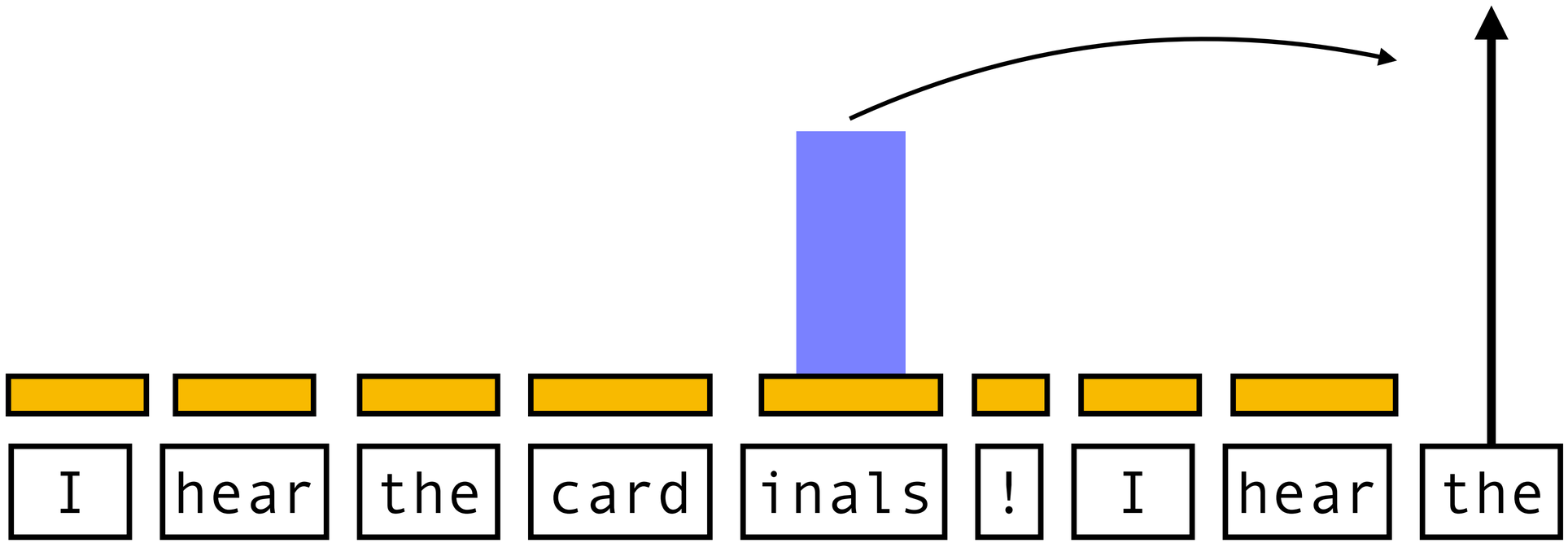
To get a “concept subspace,” we can analyze the OV matrices of concept induction heads.



Elhage et al. (2021). A Mathematical Framework for Transformer Circuits.

Concept Lens

Let’s ignore queries and keys: what if all attention forced onto this word?

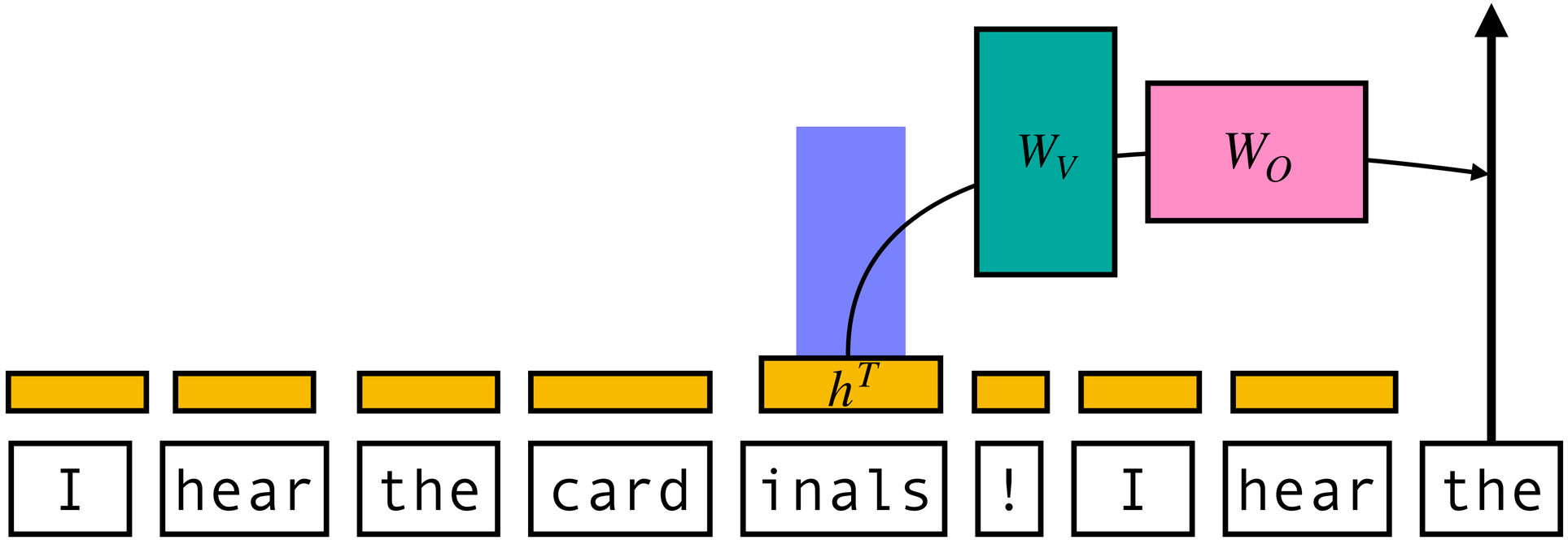


Elhage et al. (2021). A Mathematical Framework for Transformer Circuits.

Concept Lens

If so, we're literally just multiplying this hidden state by W_V and W_O and adding it to the residual stream.

$$h^T W_V W_O$$

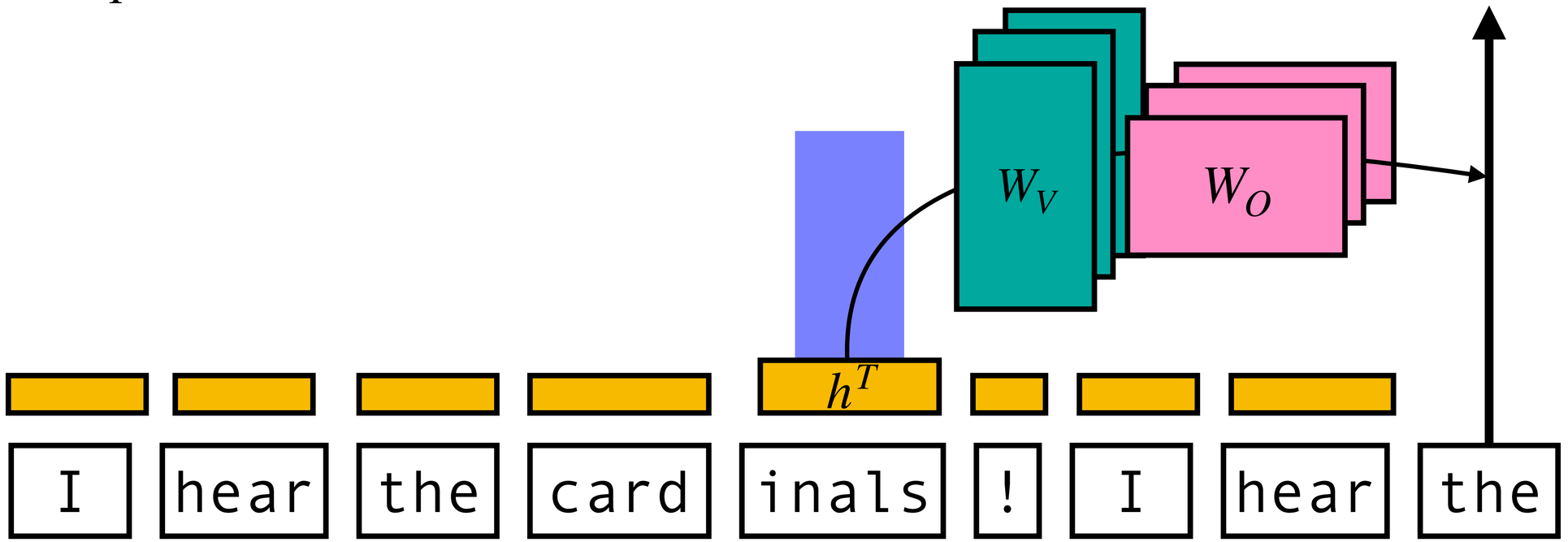


Elhage et al. (2021). A Mathematical Framework for Transformer Circuits.

Concept Lens

In the paper, we found that all of the concept heads seem to "work together," so we actually want to look at the top-k concept heads.

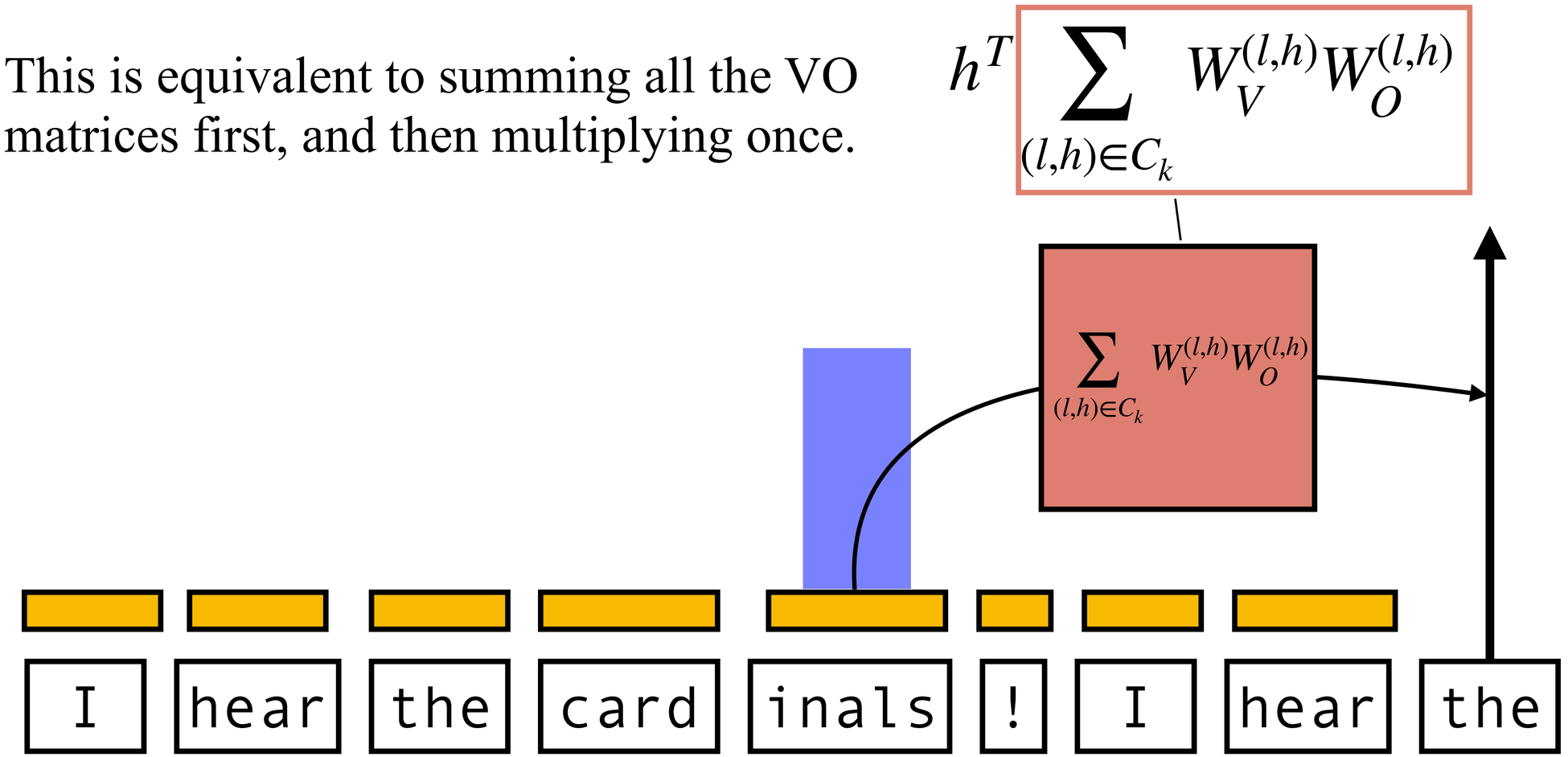
$$\sum_{(l,h)\in C_k} h^T W_V^{(l,h)} W_O^{(l,h)}$$



Elhage et al. (2021). A Mathematical Framework for Transformer Circuits.

Concept Lens

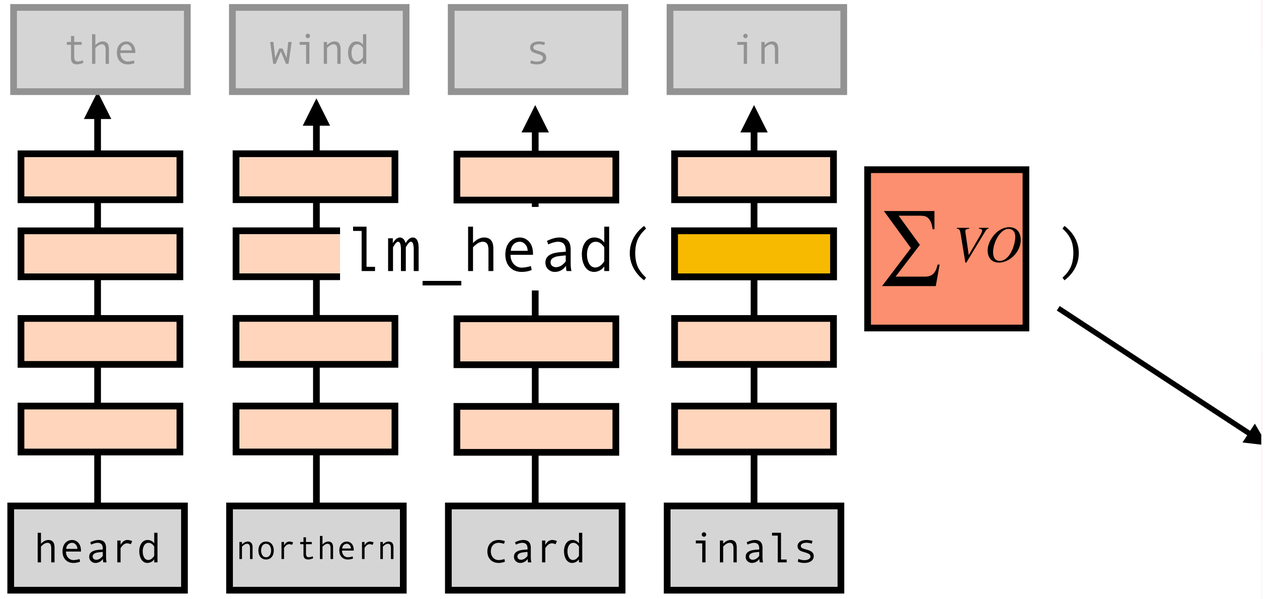
This is equivalent to summing all the VO matrices first, and then multiplying once.

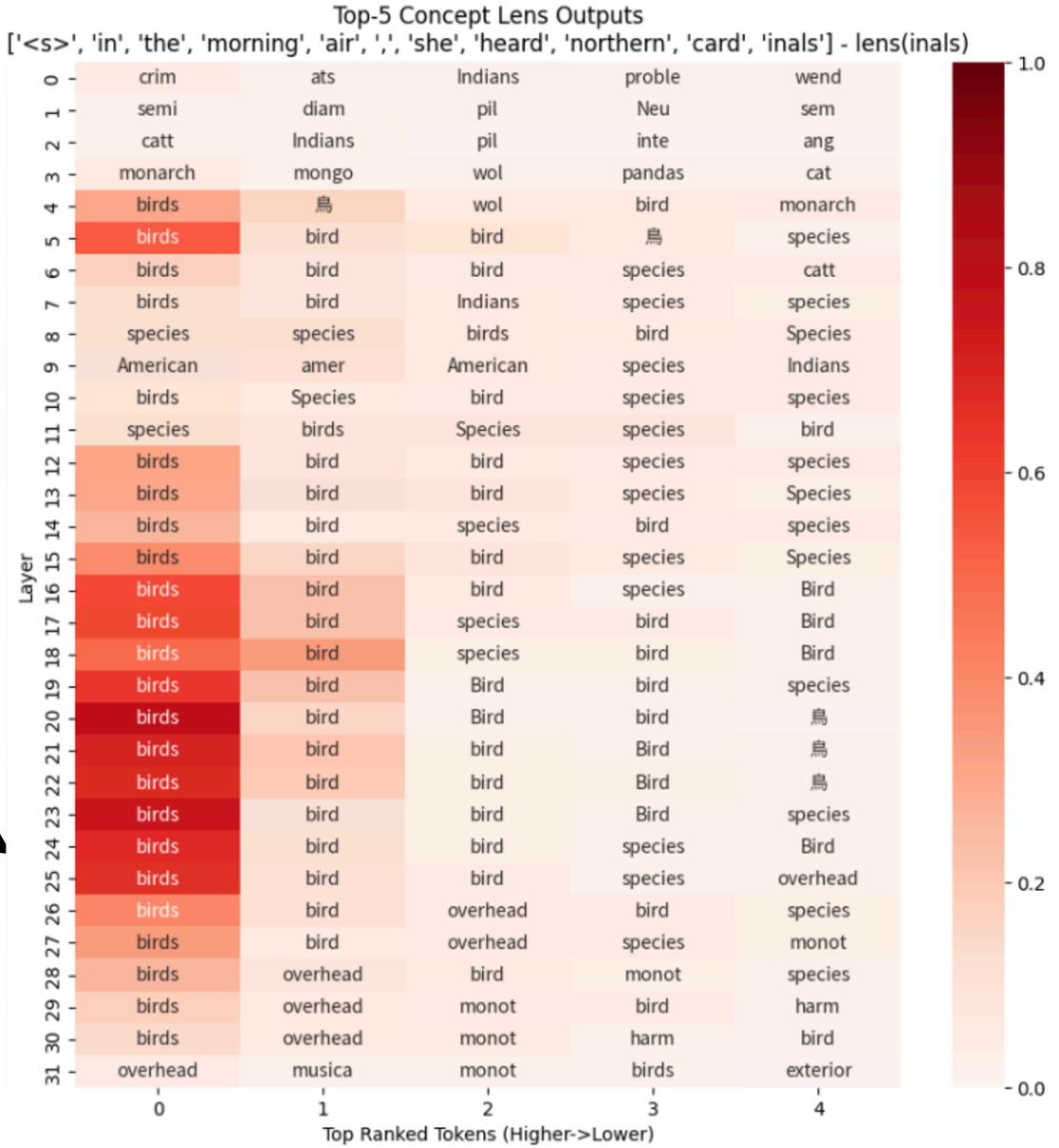


Elhage et al. (2021). A Mathematical Framework for Transformer Circuits.

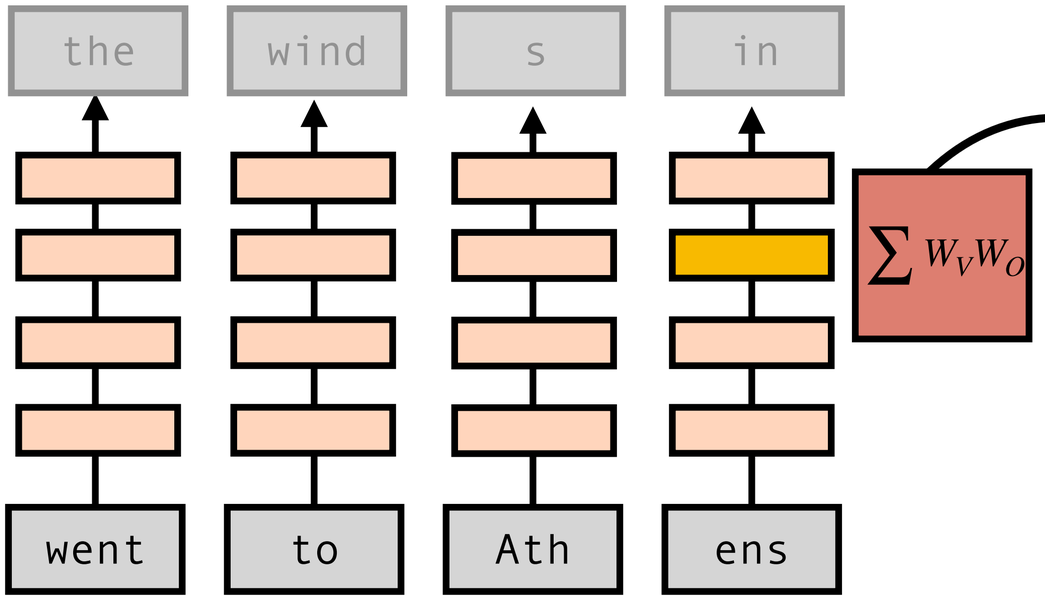
Concept Lens

Apply this matrix, and then apply the LLM's decoder head — you can literally see the semantics of a particular hidden state!



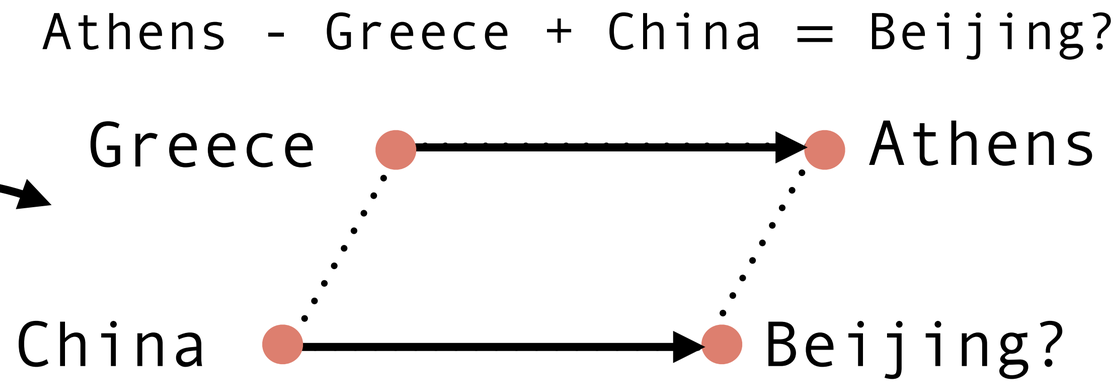


Second Attempt



(1) Run all words through a template and extract raw hidden states

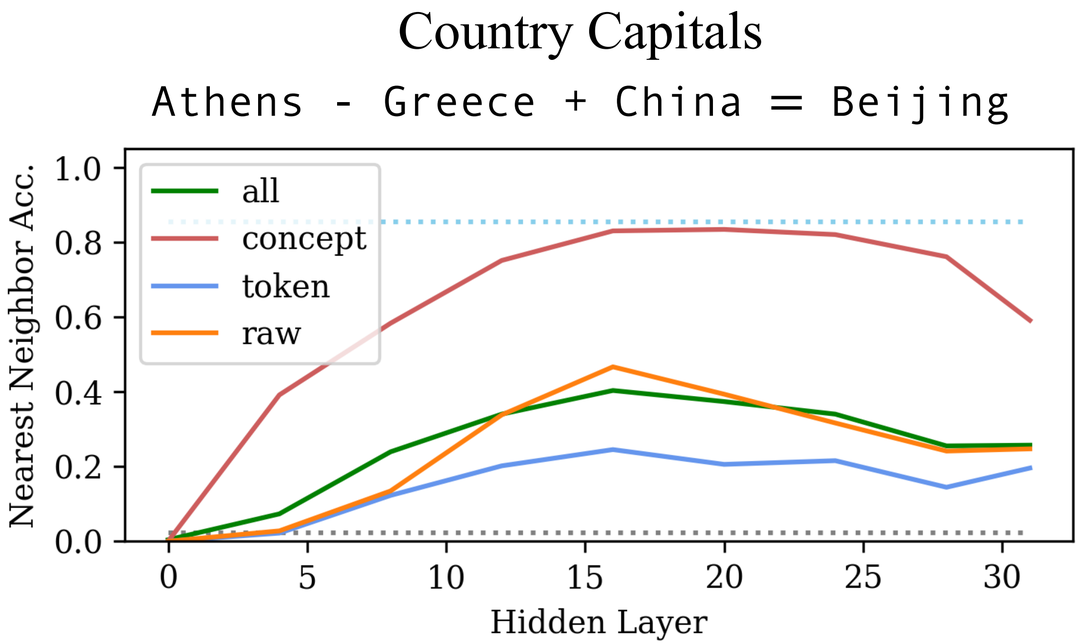
(2) Multiply each state by concept lens

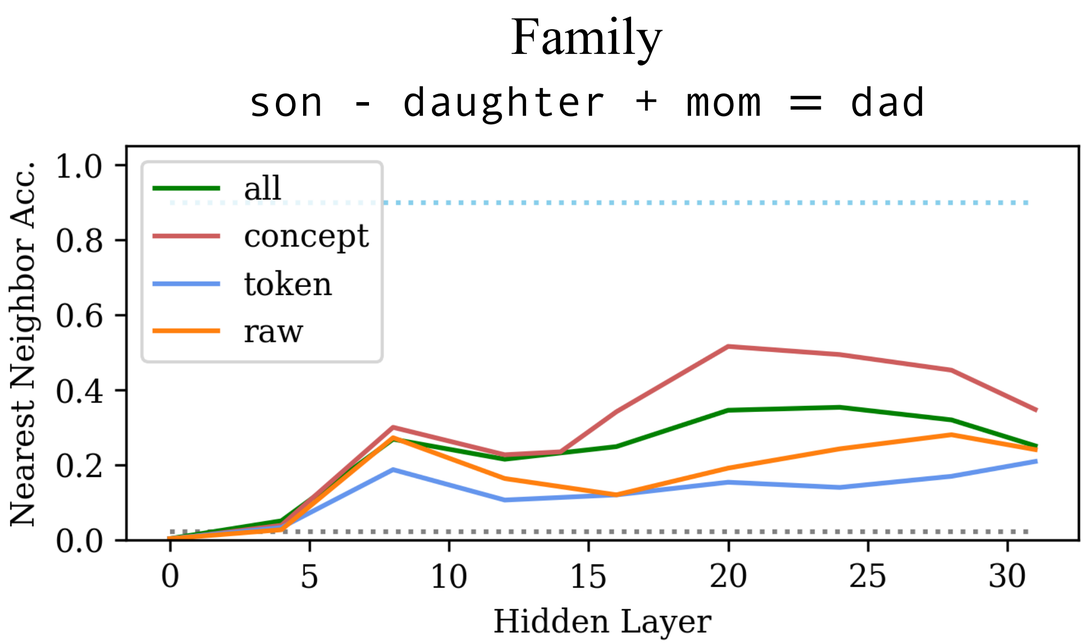


(3) Add/subtract these transformed hidden state vectors

(4) Check if the nearest neighbor (among all the countries/capitals) is correct.

Semantic tasks

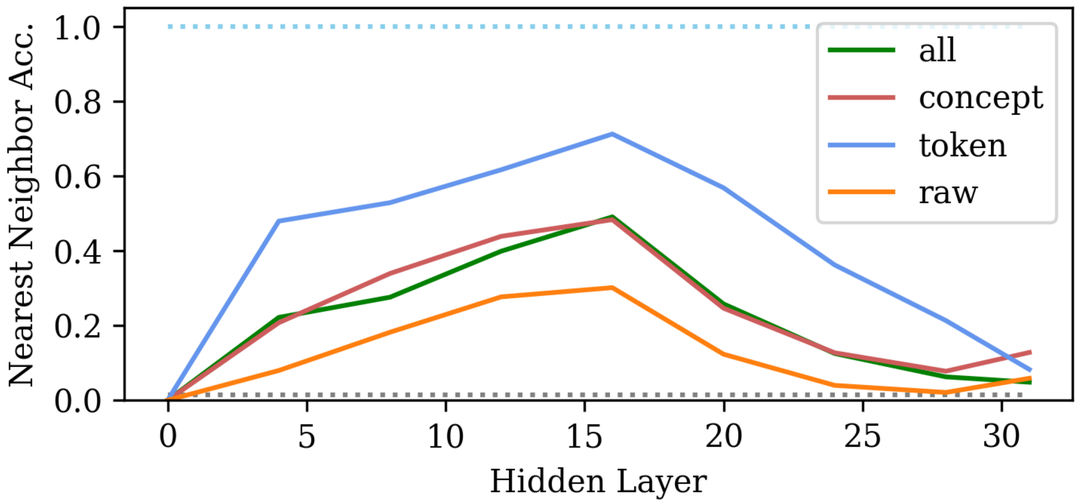




Non-semantic tasks

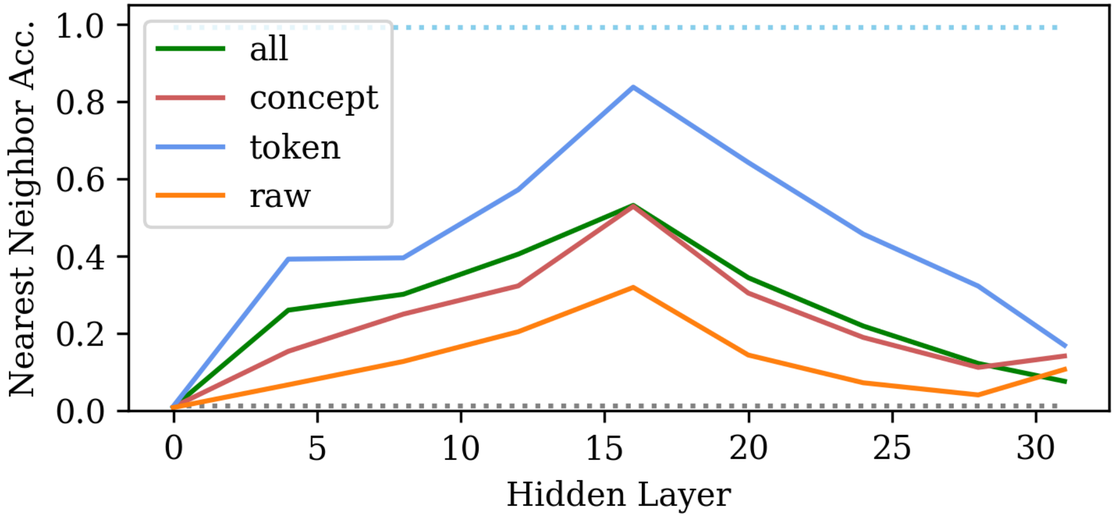
Present Participle

`code - coding + dancing = dance`



Past Tense

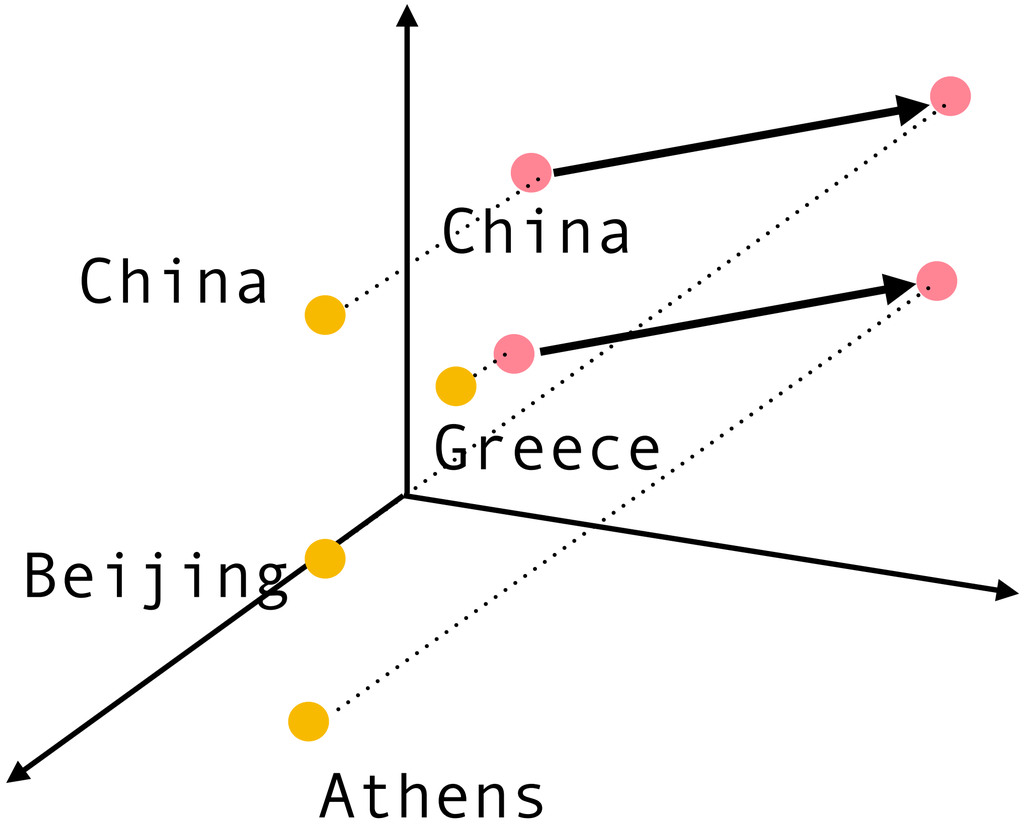
`coding - coded + danced = dancing`



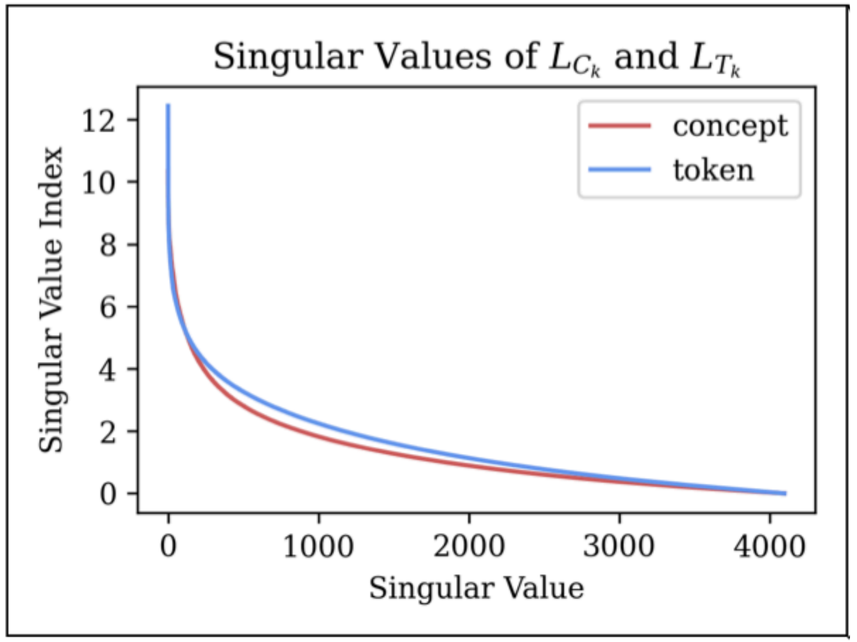
"Just focus on how the word is written."

Are these true subspaces?

I had this whole story about a "semantic subspace"…



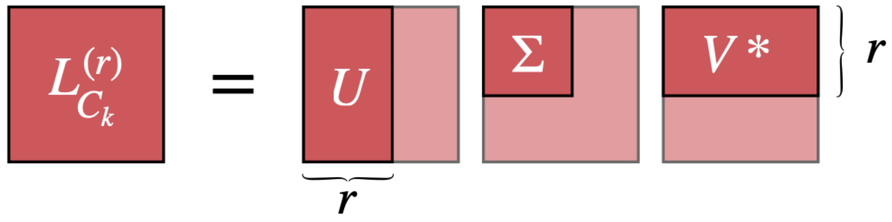
…but concept lens is not actually low-rank, so it can't be projecting onto a true subspace.

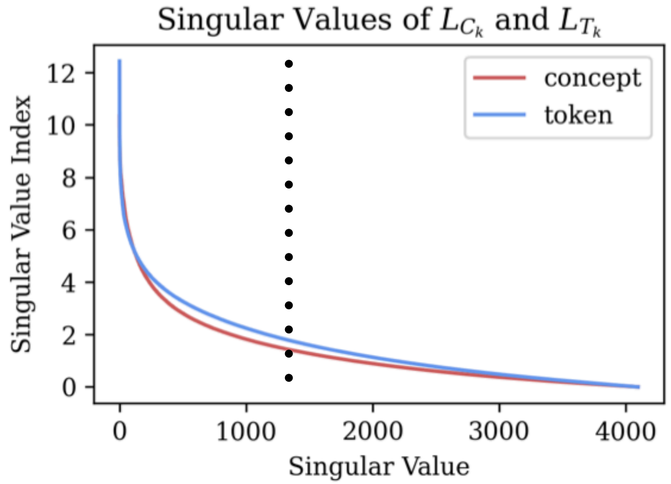


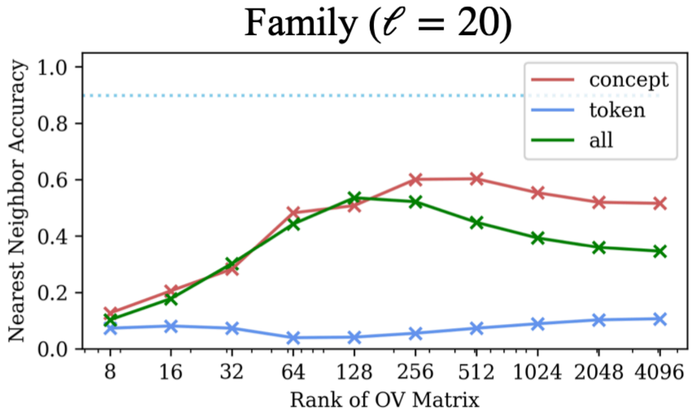
Are these true subspaces?

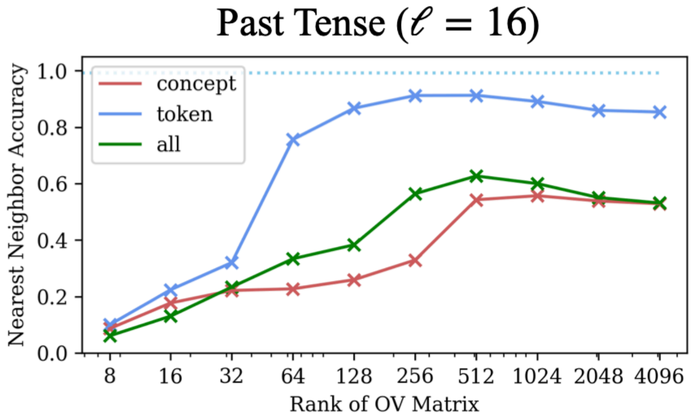
Try chopping off the bottom singular vectors. Does it work better?

(b) Take the low-rank r approximation of each lens using SVD and apply this matrix to the best layer for each task.









Low-rank isn't better, but with >256 dimensions it's about the same.

Conclusion…

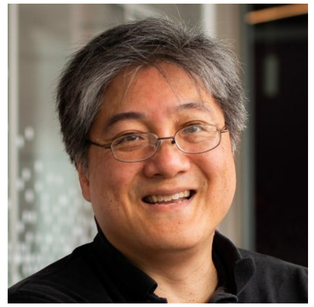
- This was submitted to mech interp workshop, but not sure if we should expand it into a full paper!
- Would love feedback on what you think would be an interesting next direction.
- Currently working on feature geometry at Goodfire, and wondering how this paper meshes with that line of work.

Thanks to my collaborators and advisors!

https://dualroute.baulab.info
https://arithmetic.baulab.info

Eric Todd

Byron Wallace

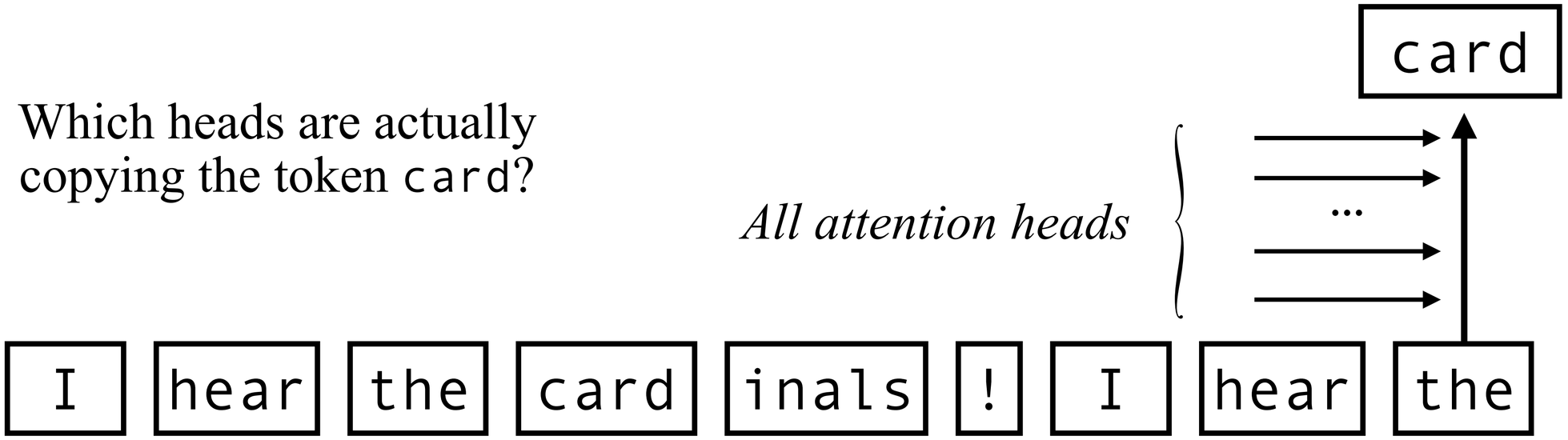


David Bau

[The following are backup slides to explain approach from the Dual Route Model of Induction paper.]

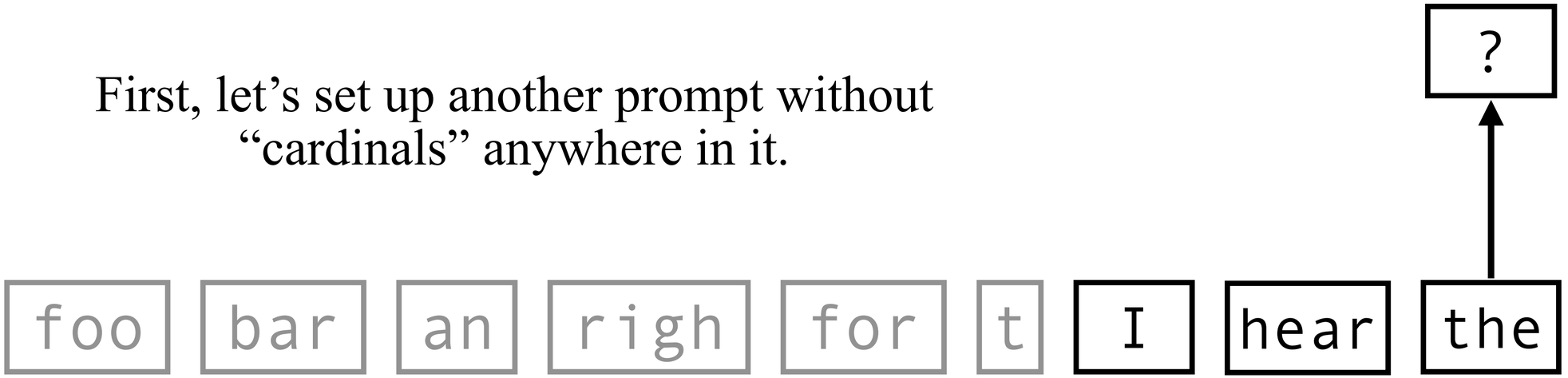
First, let's figure out how to find regular induction heads.

Which heads are actually copying the token `card`?

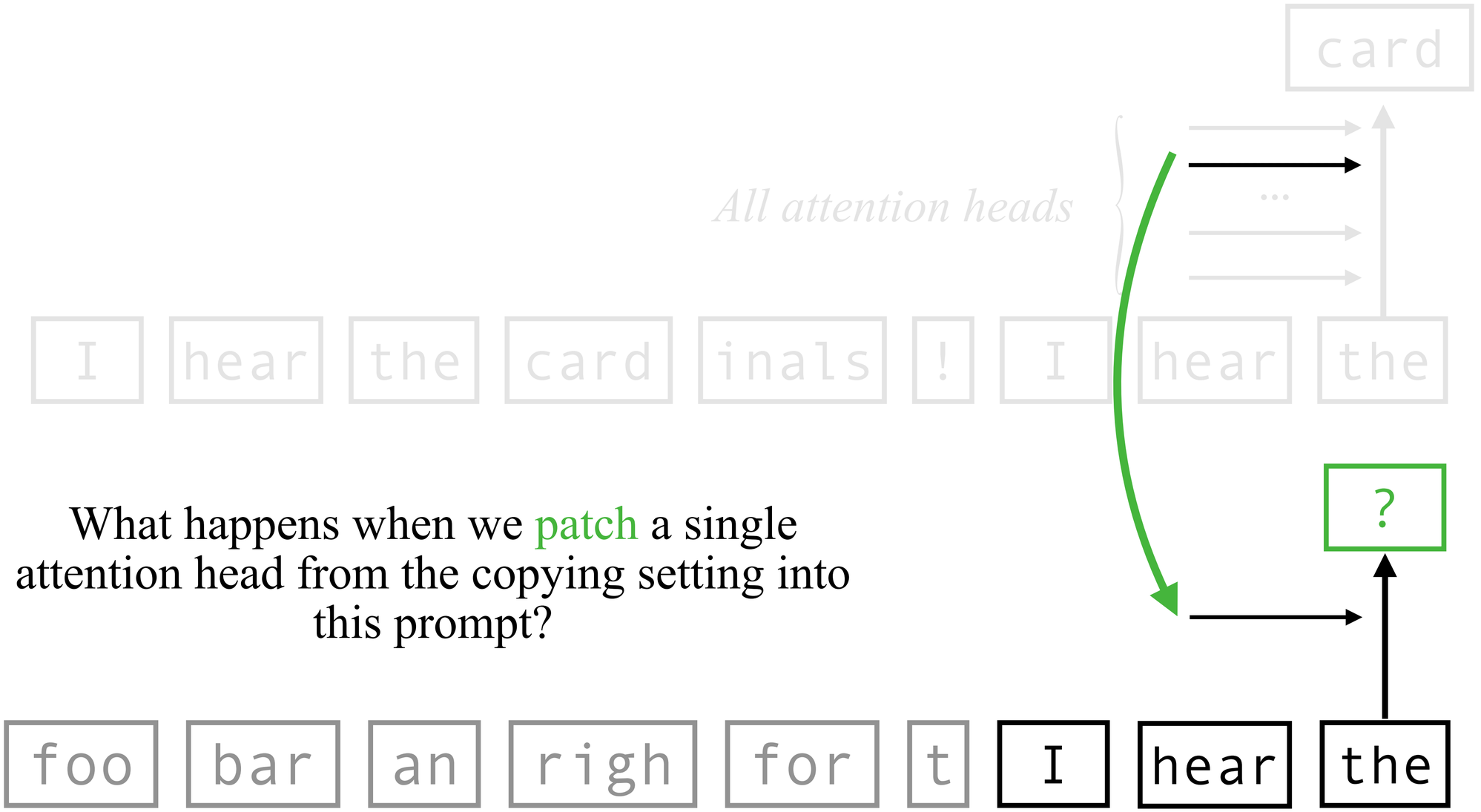


How do we find token induction heads?

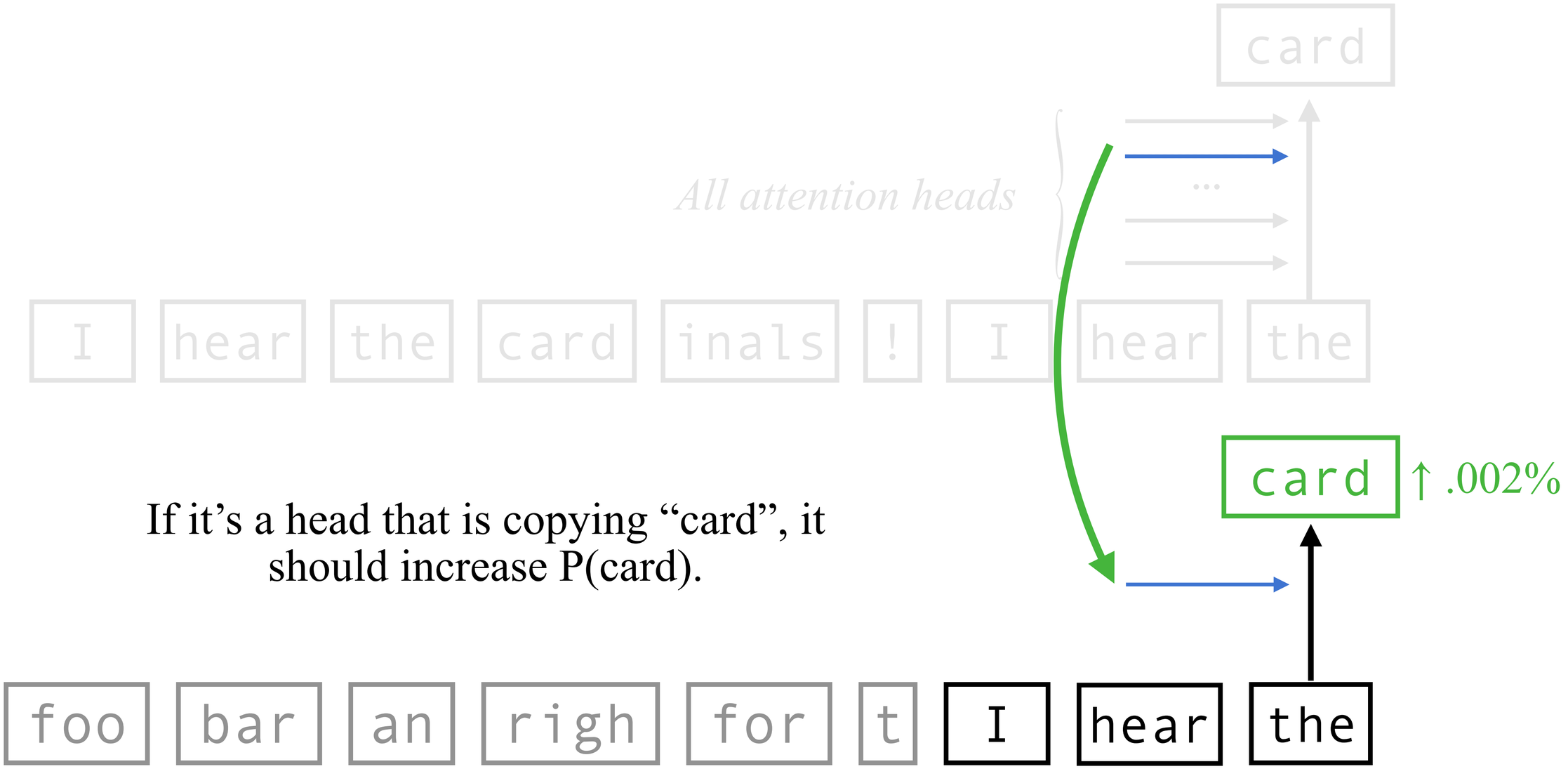
First, let's set up another prompt without "cardinals" anywhere in it.



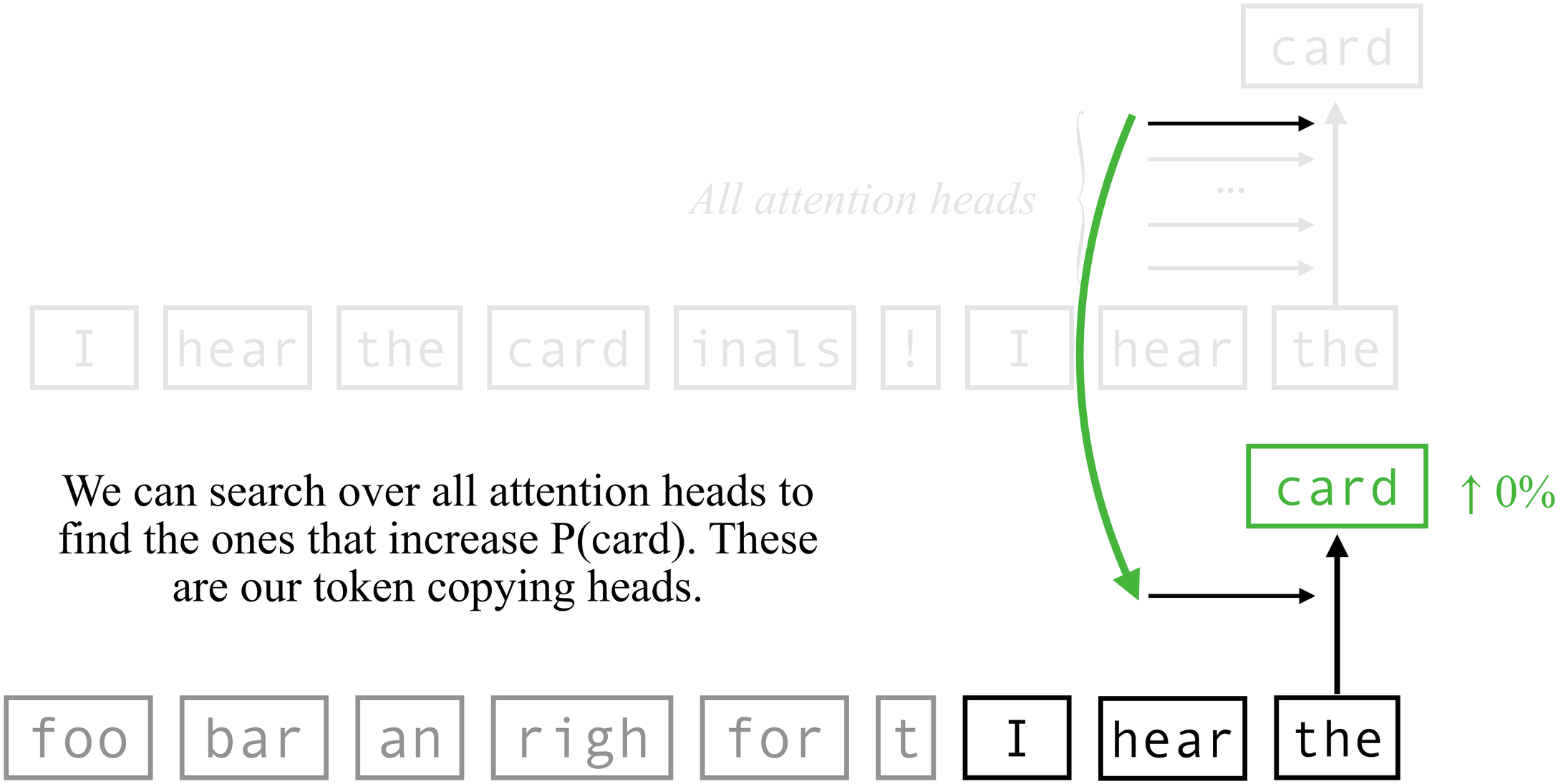
How do we find token induction heads?



How do we find token induction heads?

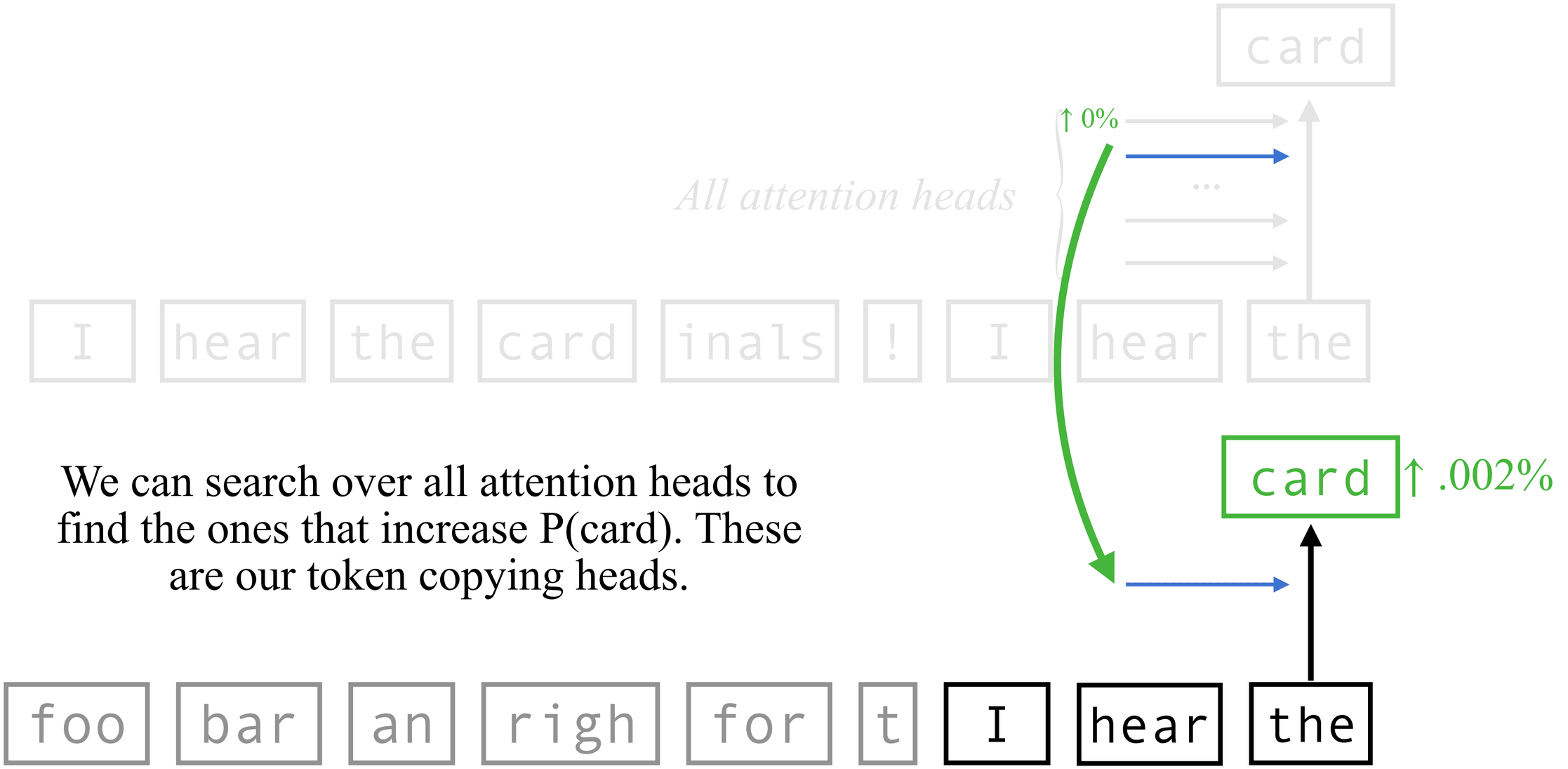


How do we find token induction heads?



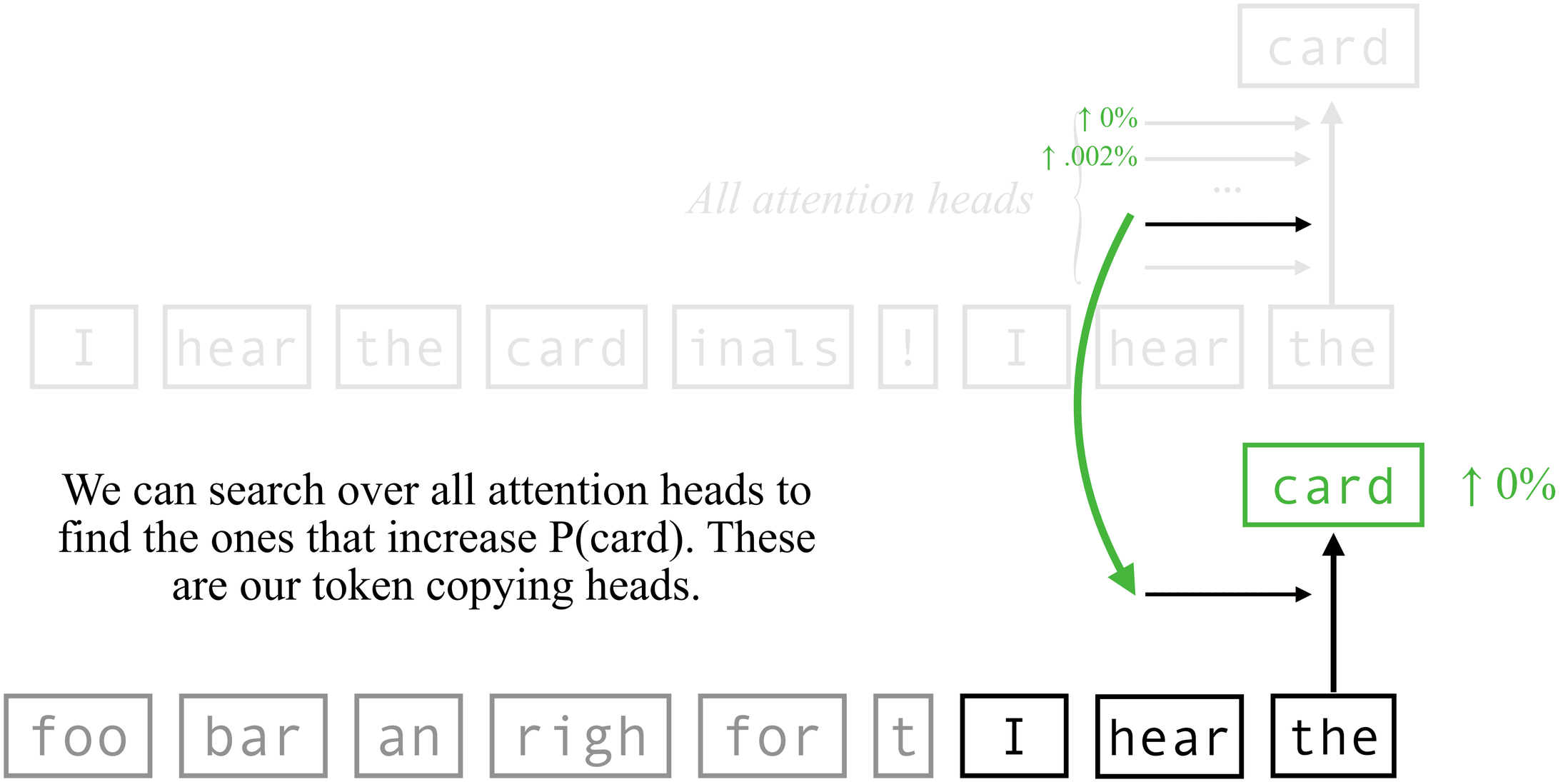
We can search over all attention heads to find the ones that increase P(card). These are our token copying heads.

How do we find token induction heads?



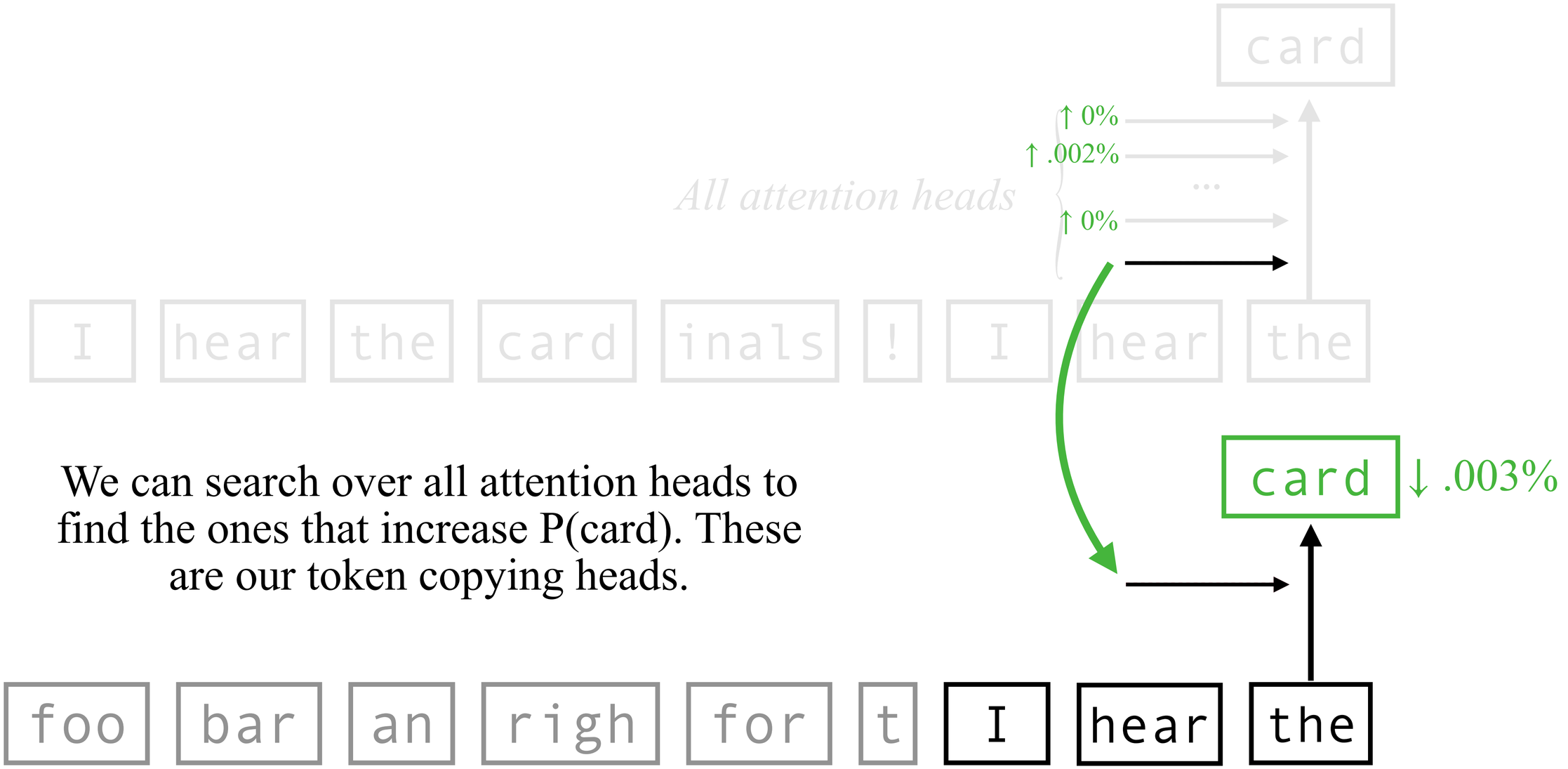
We can search over all attention heads to find the ones that increase P(card). These are our token copying heads.

How do we find token induction heads?



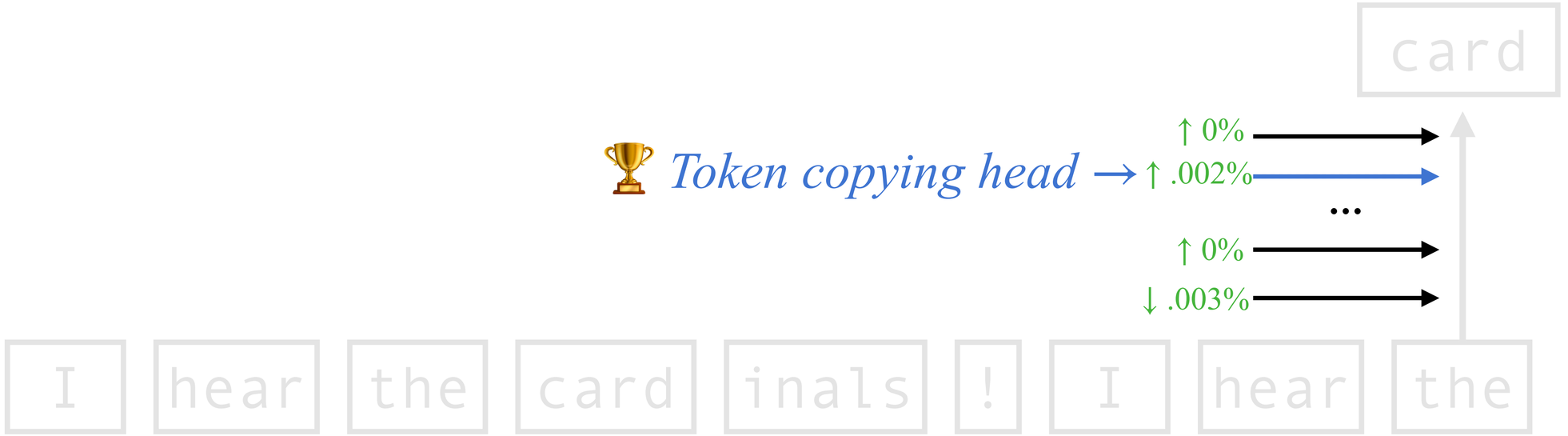
We can search over all attention heads to find the ones that increase P(card). These are our token copying heads.

How do we find token induction heads?

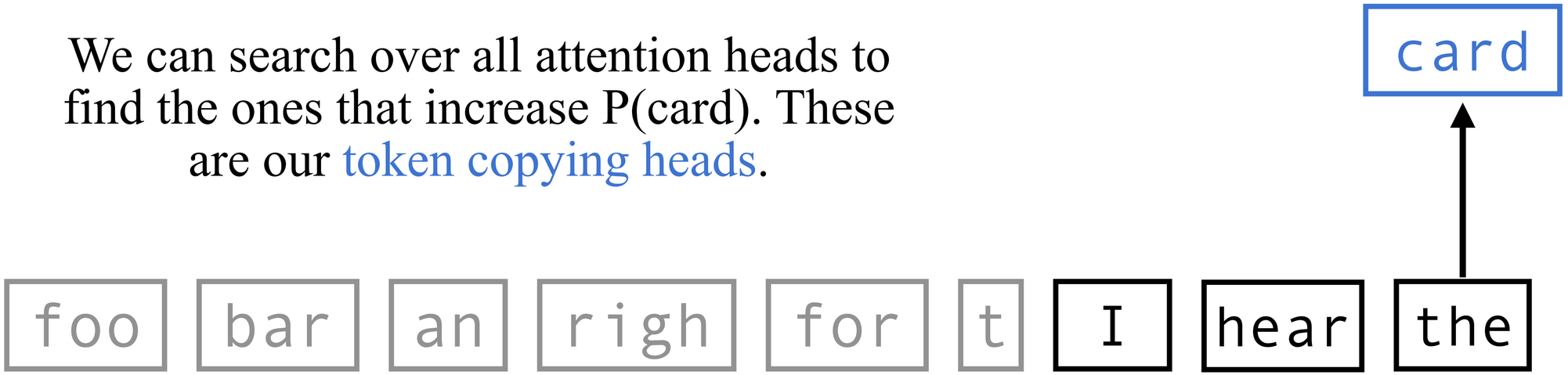


We can search over all attention heads to find the ones that increase P(card). These are our token copying heads.

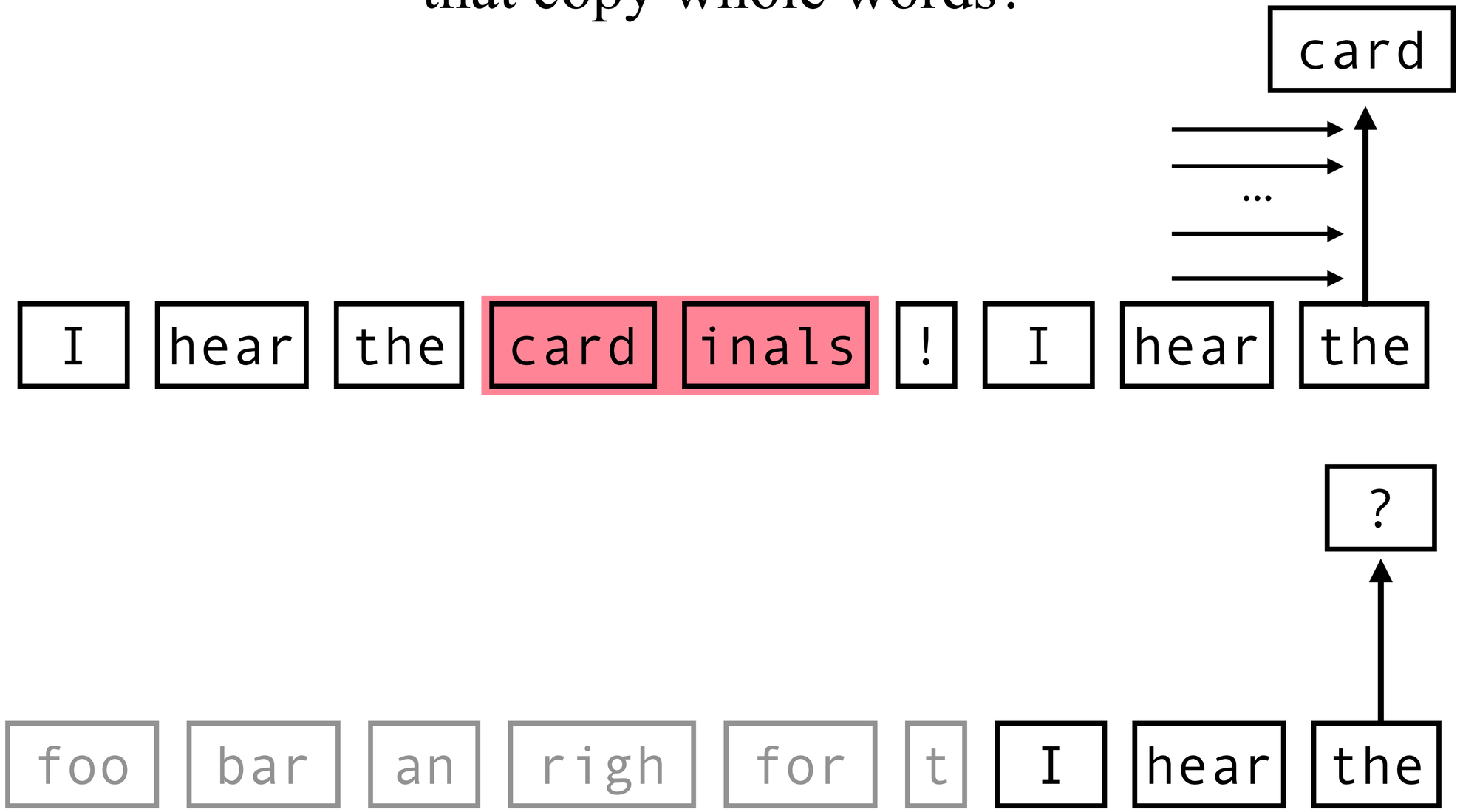
How do we find token induction heads?



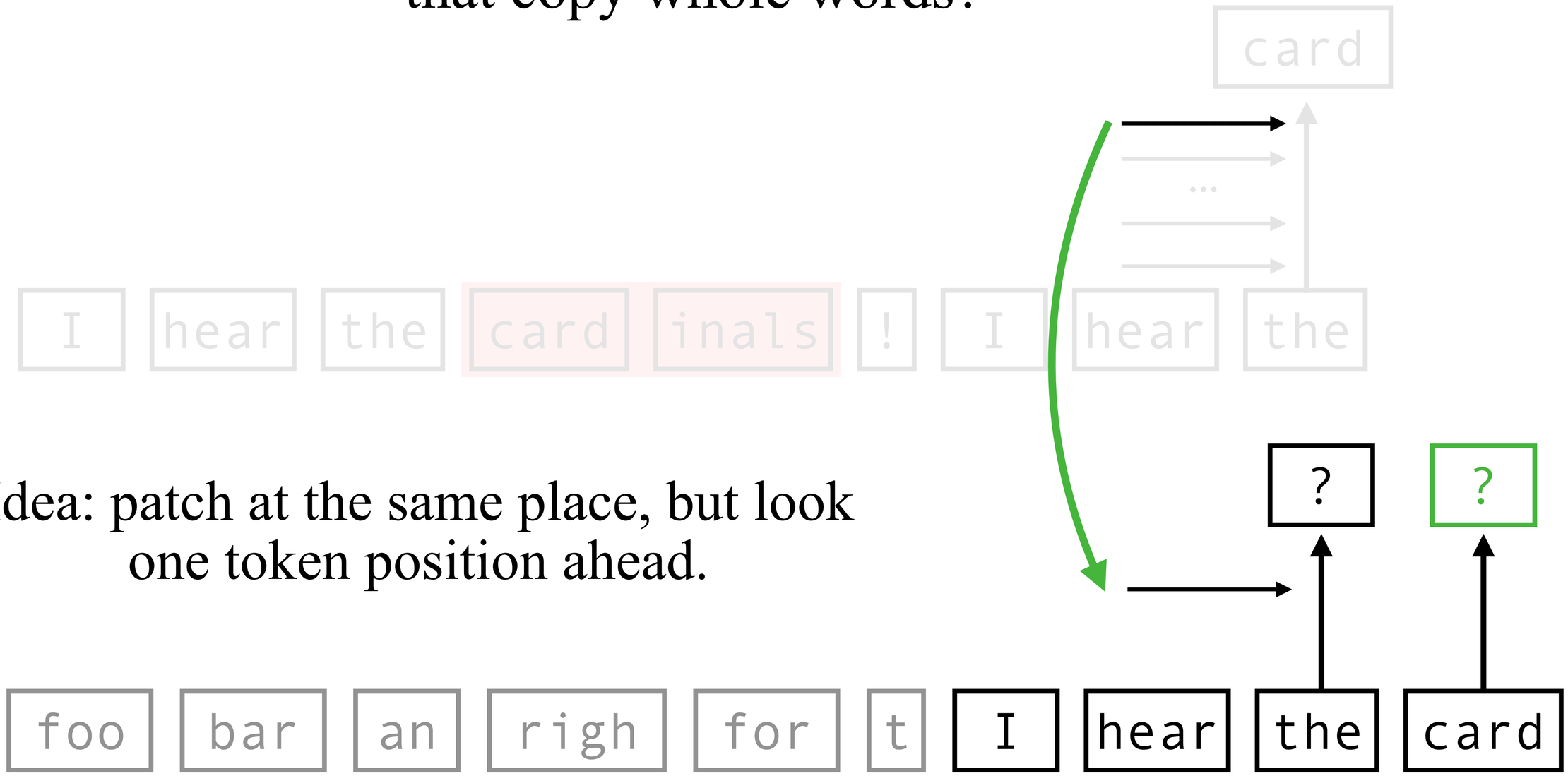
We can search over all attention heads to find the ones that increase P(card). These are our token copying heads.



Using patching, how could we find heads that copy whole words?

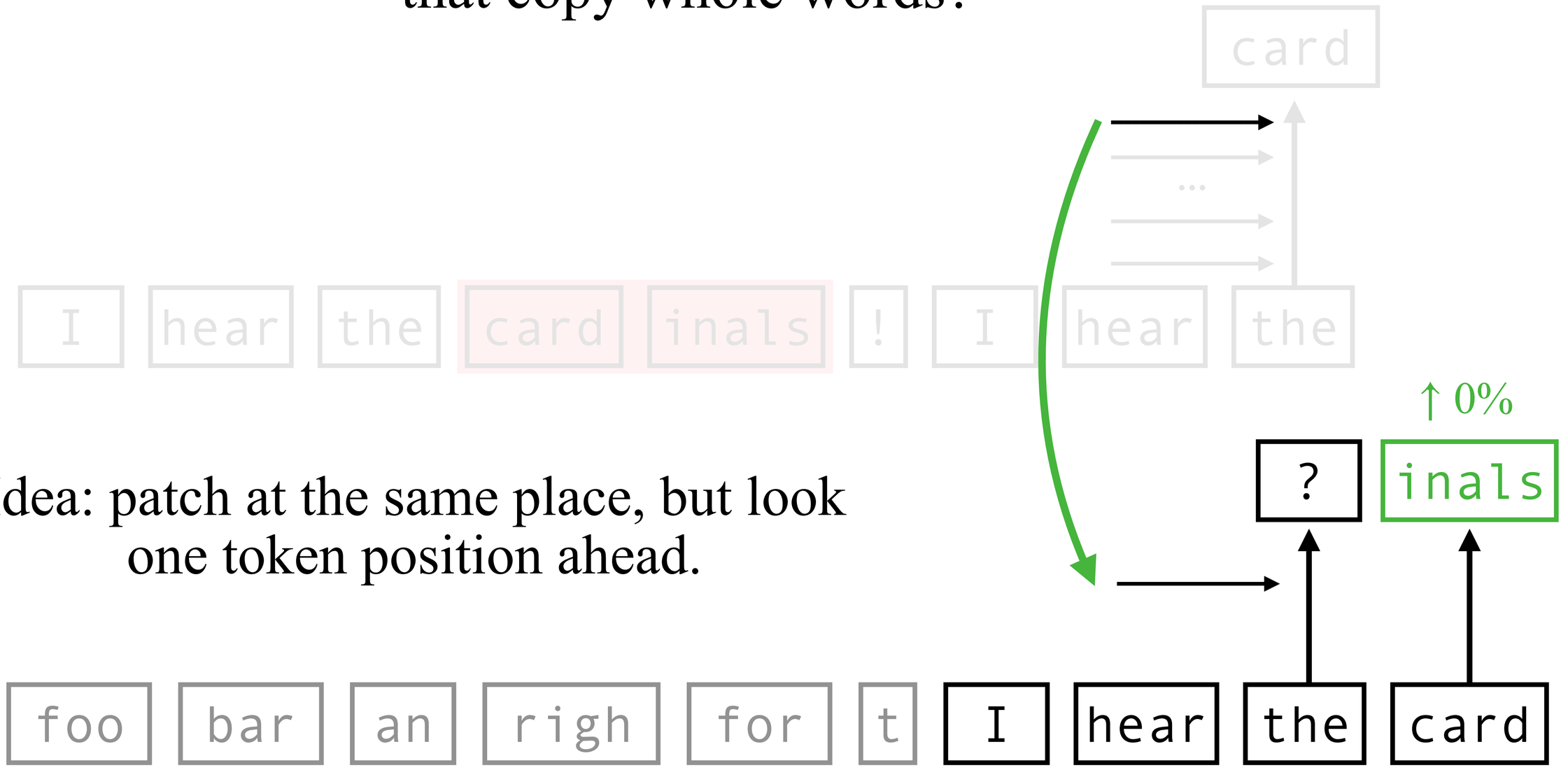


Using patching, how could we find heads that copy whole words?



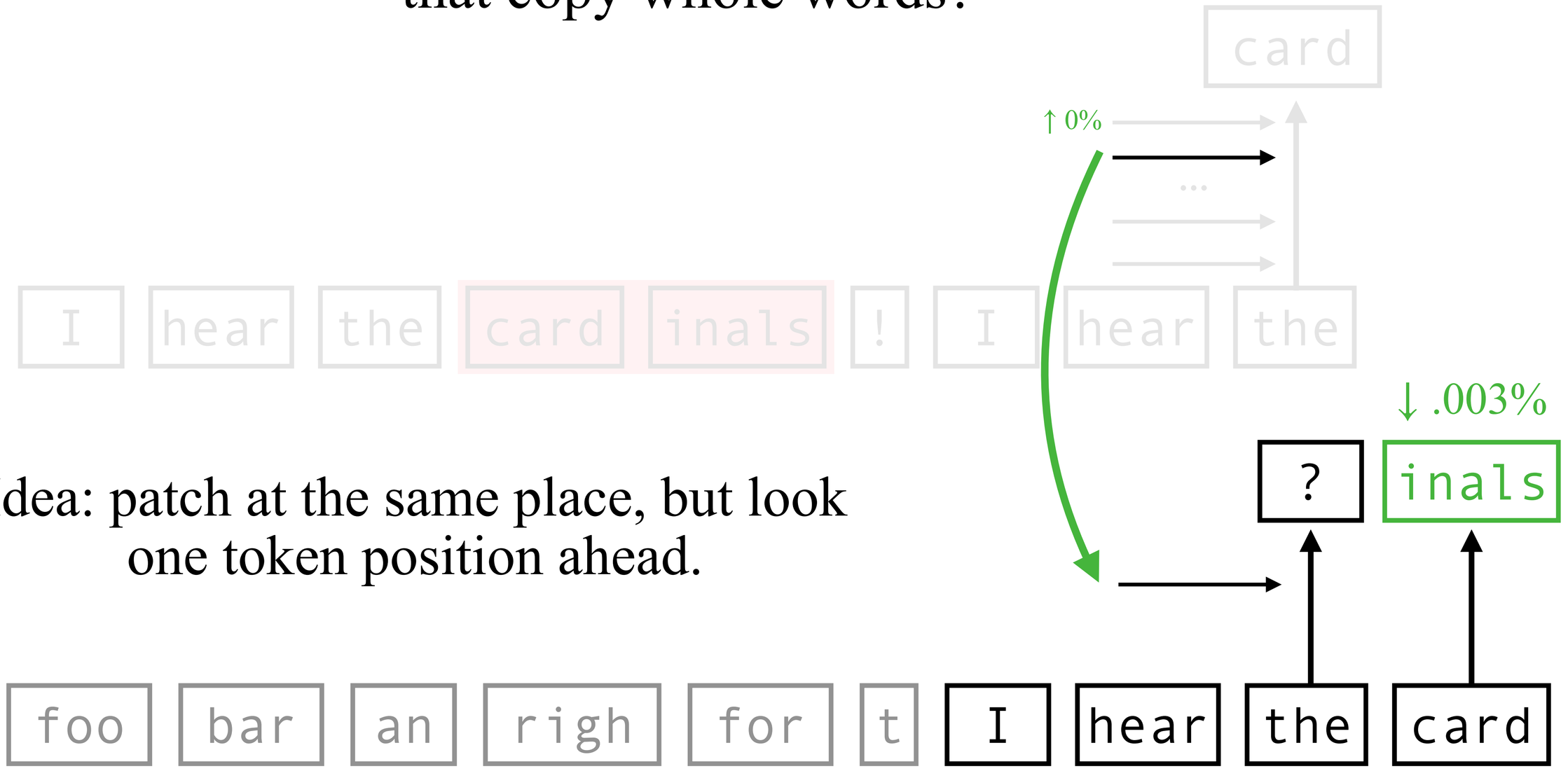
Idea: patch at the same place, but look one token position ahead.

Using patching, how could we find heads that copy whole words?



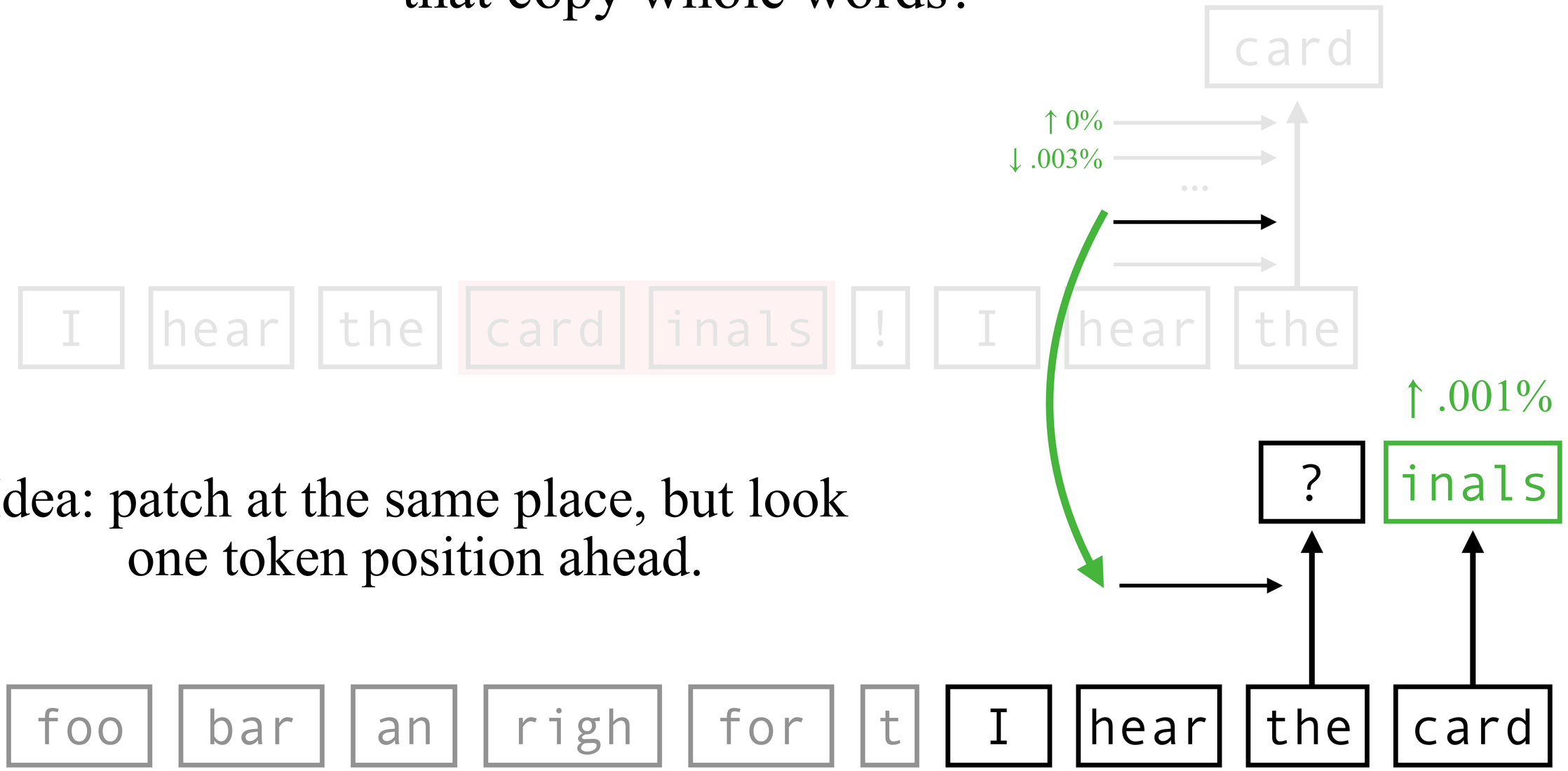
Idea: patch at the same place, but look one token position ahead.

Using patching, how could we find heads that copy whole words?



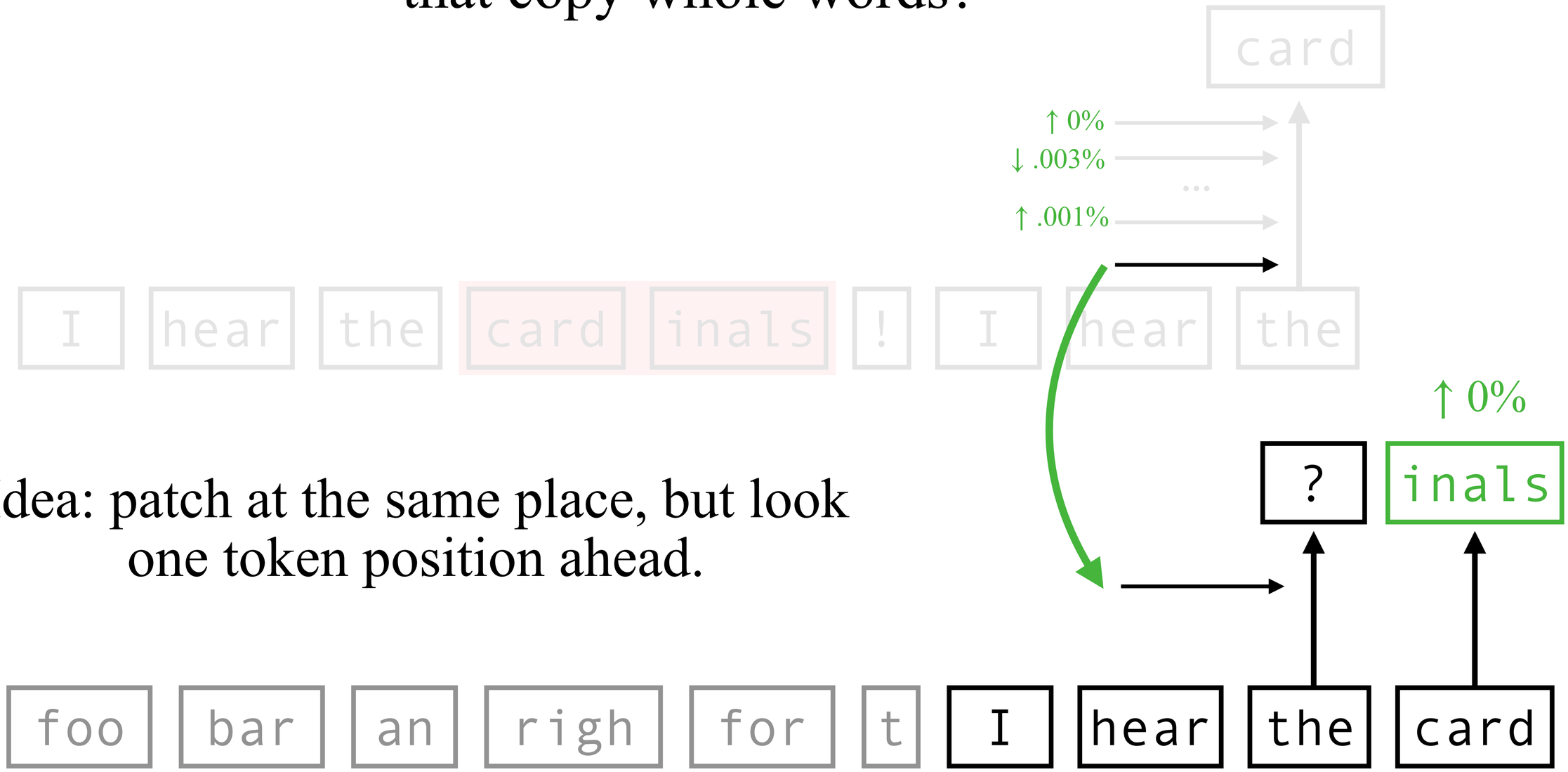
Idea: patch at the same place, but look one token position ahead.

Using patching, how could we find heads that copy whole words?



Idea: patch at the same place, but look one token position ahead.

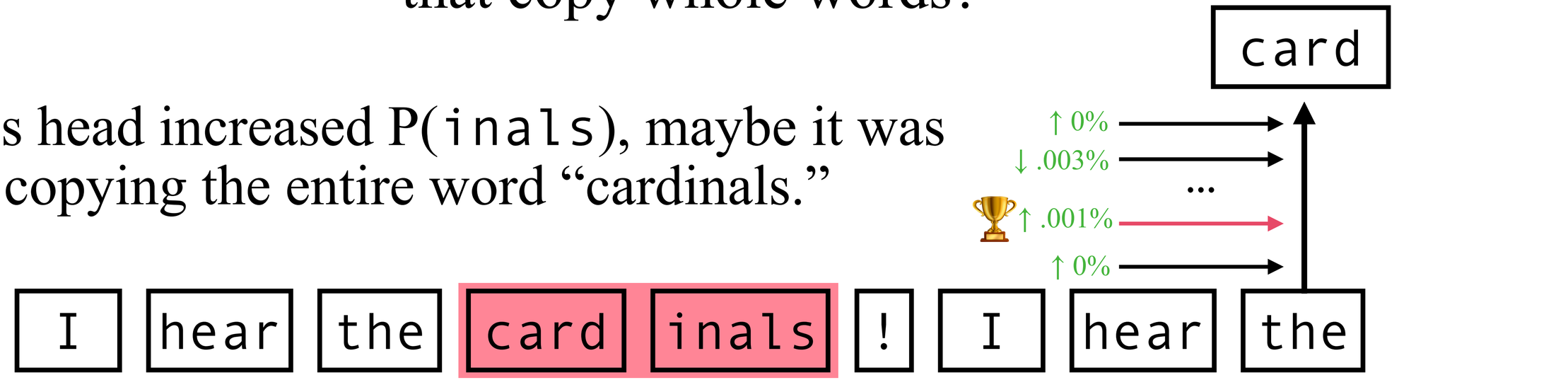
Using patching, how could we find heads that copy whole words?



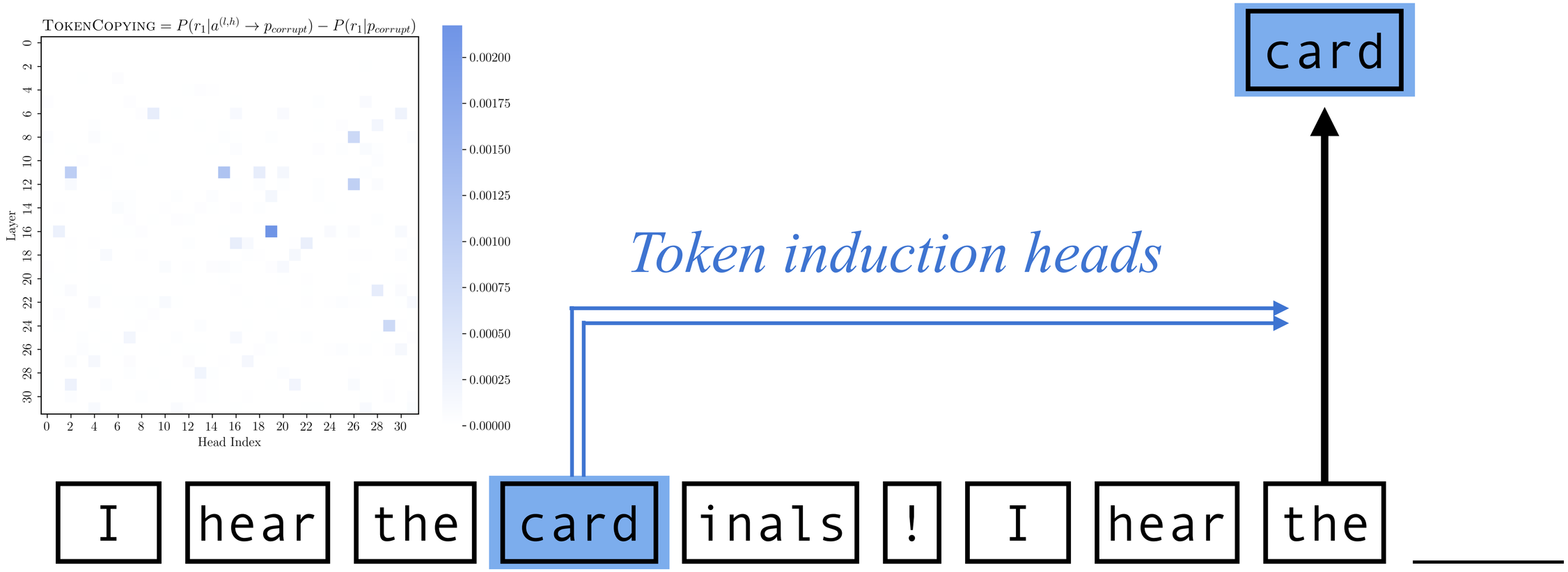
Idea: patch at the same place, but look one token position ahead.

Using patching, how could we find heads that copy whole words?

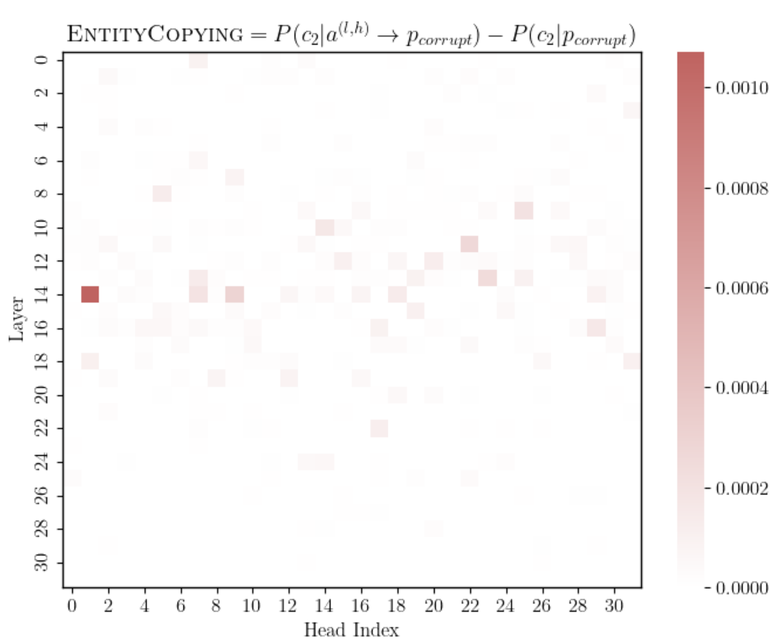
If this head increased P(`inals`), maybe it was copying the entire word “cardinals.”



So now we have one set of heads that increases P(`card`) at the next token…

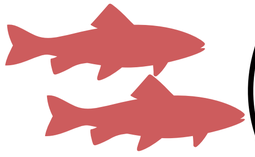


And another, separate set of heads that increases P(inals) at the next-next token.



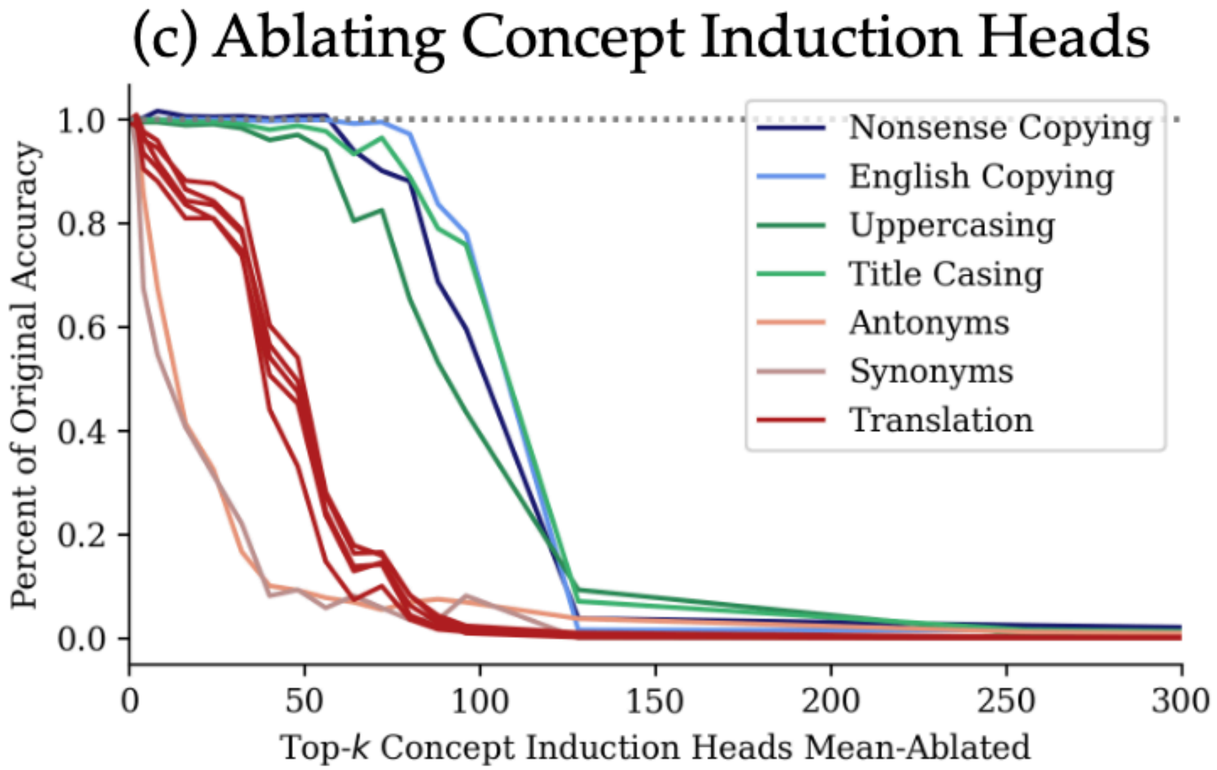
Mean-Ablating Concept Heads

Hypothesis: concept heads copy a representation of the word "salmon"

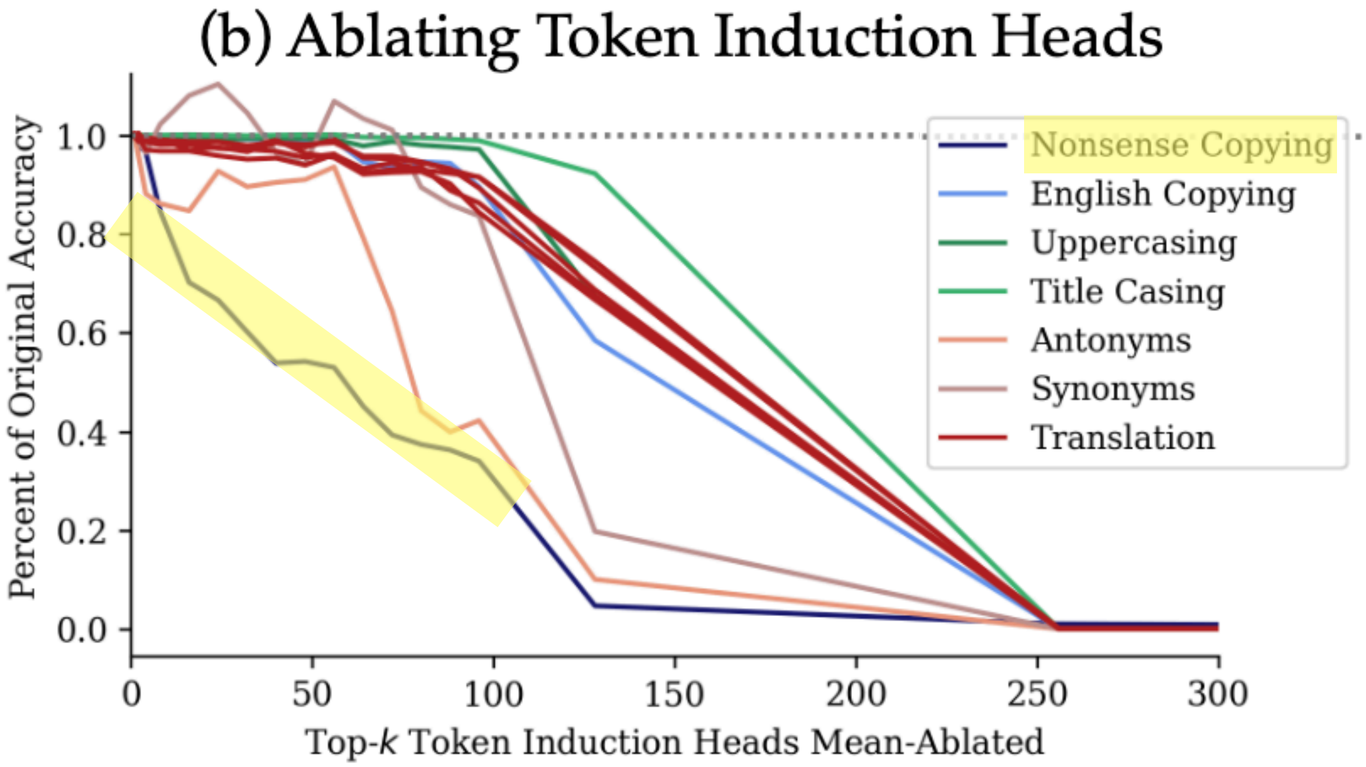


Vocabulary List Prompt Example (French-English)

```
<s> French vocab:
 1. completement
 2. inconstitutionnalité
 3. flocon
 4. chiens
 5. vaccinés
 6. racontée
 7. spécialités
 8. parfumée
 9. condensateur
10. saumons
English translation:
 1. completely
 2. unconstitutionality
 3. flake
 4. dogs
 5. inoculated
 6. told
 7. specialties
 8. fragrant
 9. capacitor
10.
```



Mean-Ablating Token Heads



Vocabulary List Prompt Example (Nonsense Copying)

```
<s> Vocab:
 1. any insp
 2. comes look
 3. the like
 4.points_
 5.   fix
 6. $$_
 7.ence Camer
 8. there
 9. object currently
 10. ercase
Vocab:
 1. any insp
 2. comes look
 3. the like
 4.points_
 5.   fix
 6. $$_
 7.ence Camer
 8. there
 9. object currently
 10.
```

Patching $k = 80$ heads outputs the patched concept almost 40% of the time (close to the model's Japanese-Chinese translation accuracy.)