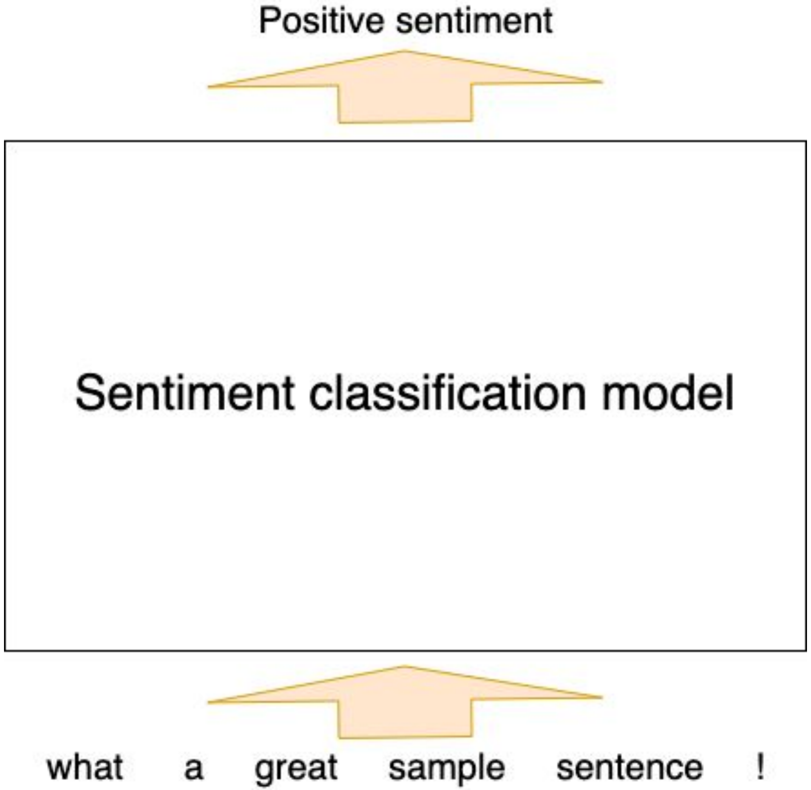
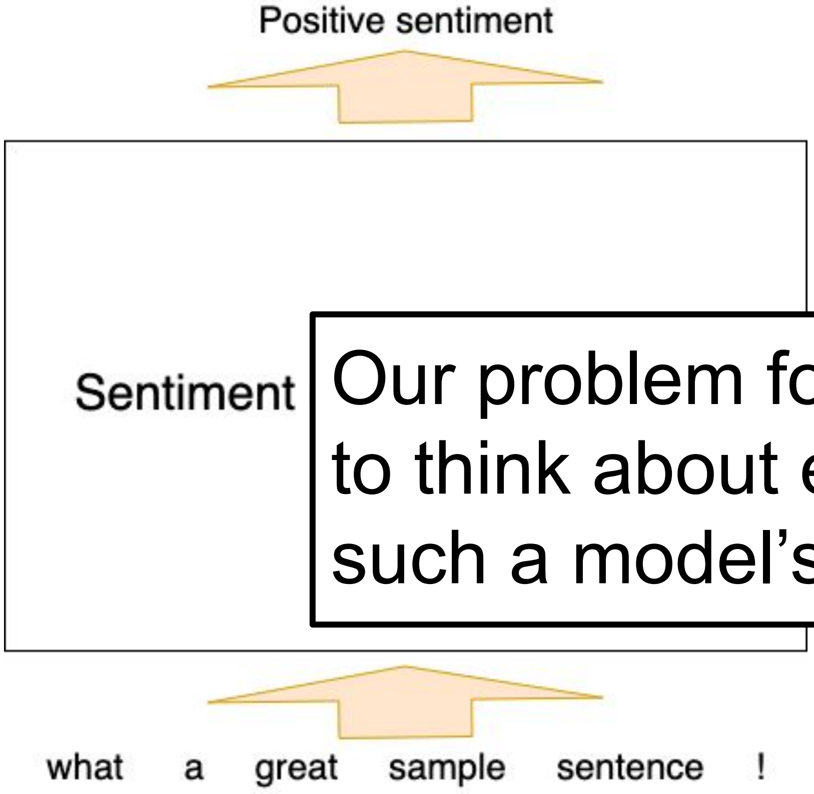
Interpretability for current NLP

Sofia Serrano

A motivating scenario: text classification



A motivating scenario: text classification



Our problem for today: how to think about explaining such a model’s output?

Why might we care about interpreting the reasons for a model’s predictions?

- To debug a model
- To help us gain insight into the training data
- To increase confidence in a model by making it easier to flag poor reasons for making a decision
  - Helpful to people in human-in-the-loop scenarios for deciding when to take a model’s advice into account
- For ethical reasons in cases where people affected by a model’s decision are owed an explanation

An outline for what we’ll talk about

What do we mean when we talk about an "explanation"? What about an "interpretable model"?

If you want the ability to interpret your model, what options do you have?

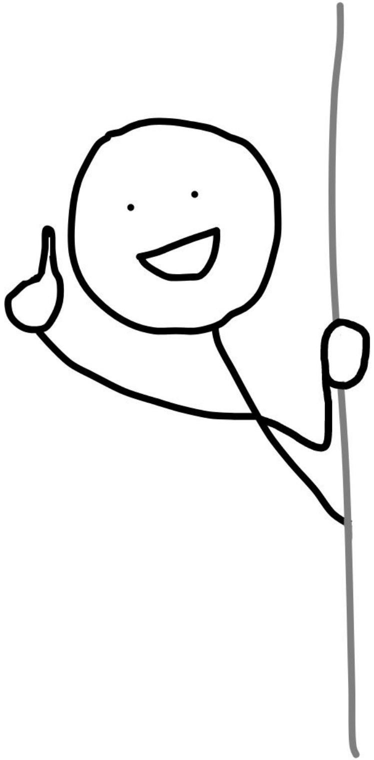
Walking through a post-hoc method for interpreting a model (LIME), plus some discussion of evaluation

Walking through some work on interpreting an intrinsic part of a model (namely, attention in transformers), plus some discussion of evaluation

A quick aside about scope

Most of what we'll be talking about is not exclusively applicable to NLP, but to interpreting (some) machine learning models more broadly

(but it's work that comes up a lot in interpretability discussions on the NLP side of things too)



Defining terms

What qualities do we look for in an **explanation**?

Faithfulness

Is the explanation true to what the model did?

Utility to humans

Is the explanation helpful to end users?

(see Doshi-Velez and Kim 2017, Madsen et al. 2022)

What qualities do we look for in a (**globally**) interpretable **model**?

Algorithmic transparency

e.g., guarantees about convergence or the shape of the error surface

Decomposability

Are the different pieces of the model understandable on their own?

Simulatability

Can a person hold the whole model in their head at once?

(see Lipton 2016)

How do these concepts apply, or not, to models covered in this class so far?

Neural networks: … depending on the architecture, there’s some argument over whether decomposability applies, but it’s not a clear-cut case

Logistic regression: Algorithmically transparent and decomposable, but not necessarily simulatable

FSAs: certainly decomposable, and algorithmically transparent given enough effort, but not always simulatable

How do these concepts apply, or not, to models covered in this class so far?

Neural networks: … depending on the architecture, there’s some argument over whether decomposability applies, but it’s not a clear-cut case

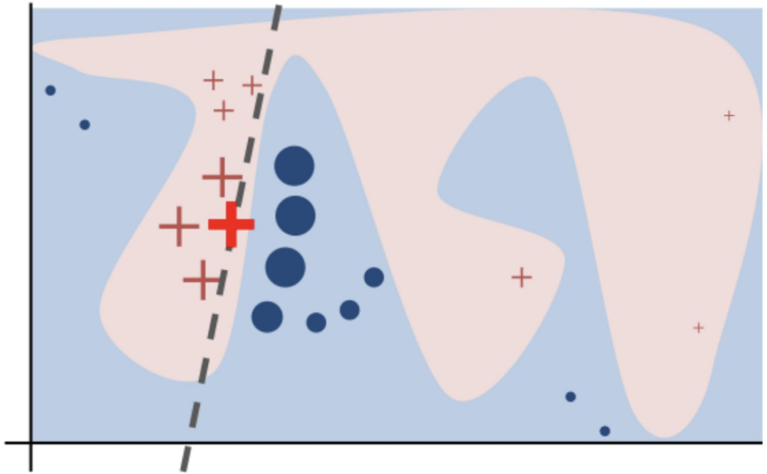
Logistic regression: Algorithmically transparent and decomposable, but not necessarily simulatable

FSAs: certainly decomposable, anc effort, but not always simulatable

These concepts are a very tall order for current NLP models! So we’ll focus on **local** explanations.

Global versus local explanations

To borrow a figure from Ribeiro et al. 2016:



A global explanation describes the entire model across all its possible inputs.

A local explanation describes only the parts of the model relevant for a particular instance's decision.

Thinking about our available choices

What choices are available to someone who hopes to interpret their eventual model?

Restrict yourself to a class of model that’s more readily interpretable

See: enduring popularity of linear models in applied-NLP settings

Apply a post-hoc, model-agnostic method for producing explanations

Figure out how to interpret your model of choice

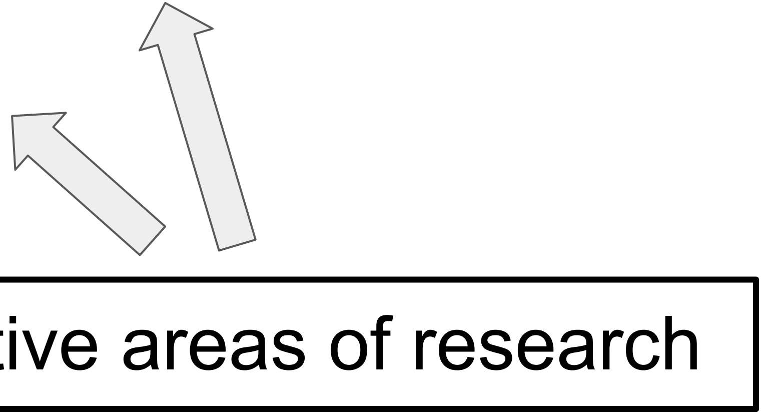
What choices are available to someone who hopes to interpret their eventual model?

Restrict yourself to a class of model that’s more readily interpretable

See: enduring popularity of linear models in applied-NLP settings

Apply a post-hoc, model-agnostic method for producing explanations

Figure out how to interpret your model of choice



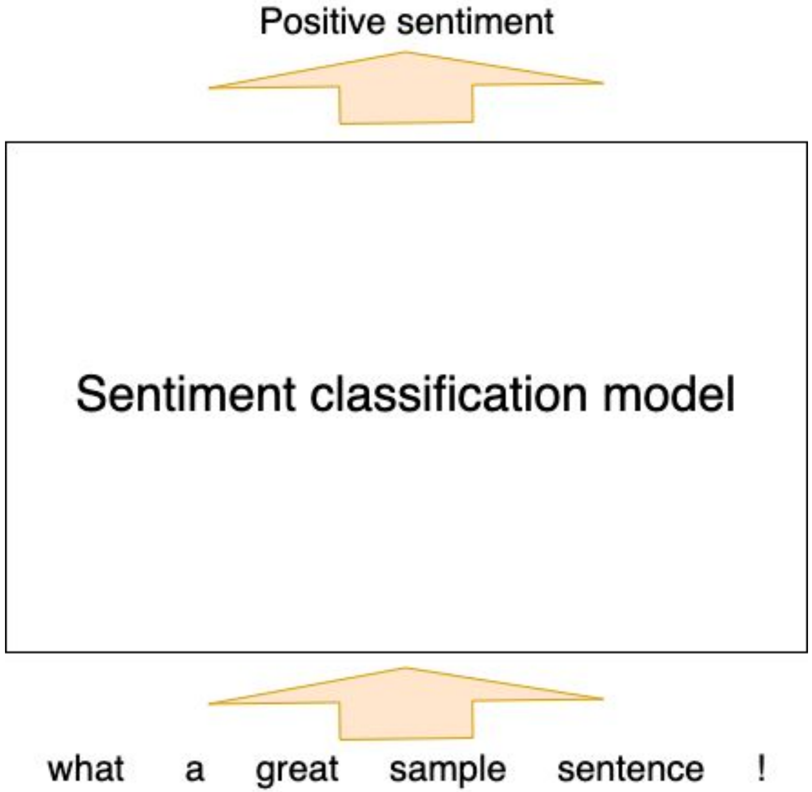
Active areas of research

Post-hoc methods for interpreting models

What do we mean by “post-hoc”?

Any method for getting us an explanation that doesn’t make assumptions about the structure of the model.

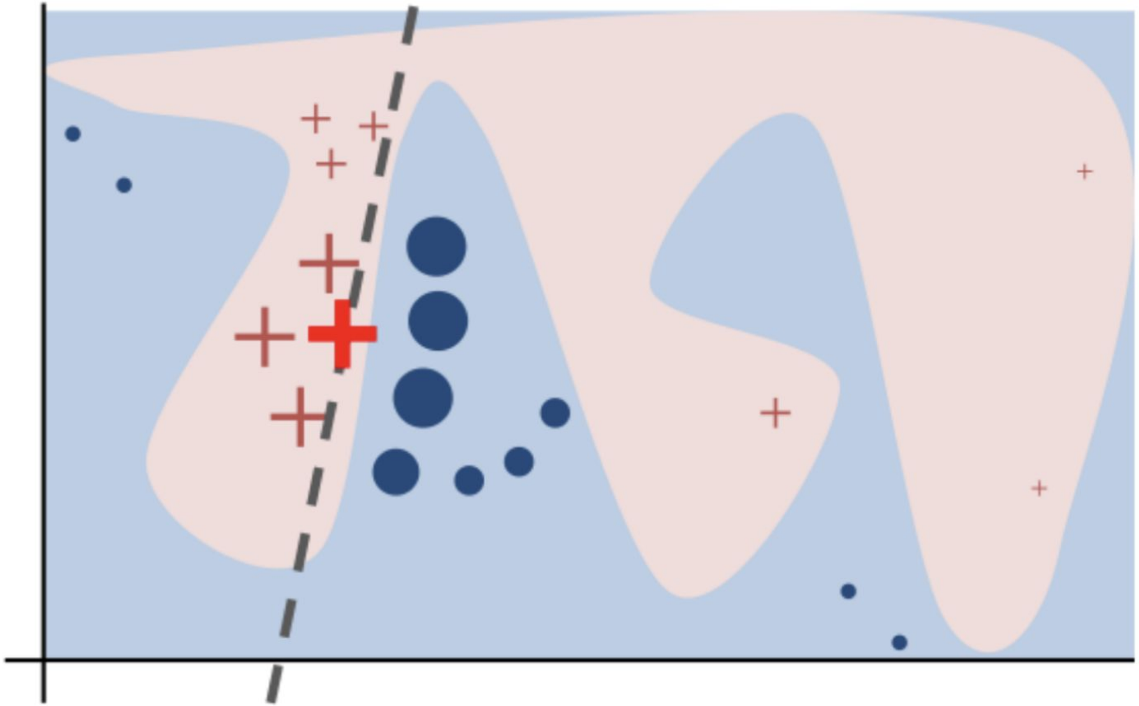
Key challenge for these methods: how do we get information about what caused the model to make its decision without access to the model’s intermediate calculations?



Walking through an example: LIME (Ribeiro et al. 2016)

Core idea:

Sample lots of instances, get the model's decisions for those, and weight them by how close they are to the instance being explained



Setting up for LIME:

1. Define an interpretable representation scheme for any possible input to the model
2. Pick your class of interpretable models to use as proxies. LIME’s output will be a proxy model of this type.
3. Define a complexity function measuring how complex the potential interpretable model is
4. Define a proximity function describing how “close” an instance is to the instance x to be explained
5. Define a fidelity function measuring how (locally) *unfaithful* a potential interpretable model is to the model being explained

Setting up: input representation scheme

We want an interpretable representation scheme for any possible input to the model.

For text, bag-of-words representations of instances are typically the go-to.

Setting up: picking a class of interpretable model and defining its members' complexity

Some examples given in the LIME paper:

- Linear models
- Decision trees
- Falling rule lists

We just need to be able to describe any model in our class as a series of **presences** and **absences.**

We define a function telling us how complex any specific model g in this class is:

$$\Omega(g)$$

(could be based on number of features, tree depth, etc.)

Setting up: defining a proximity function

We want a function that tells us how “close” any instance is to instance x.

How do we do this?

- Cosine similarity between bag-of-words representations?
- Semantic similarity measure computed using a contextual word embedding model?
- Considering metadata?

Whatever we decide, we call this function π_x

Setting up: defining a fidelity function

Should represent how unfaithful a candidate model g is to the original model f around instance x.

$$\mathcal{L}(f, g, \pi_x)$$

For its experiments, the LIME paper uses

$$\mathcal{L}(f, g, \pi_x) = \sum_{z, z' \in \mathcal{Z}} \pi_x(z) \left(f(z) - g(z')\right)^2$$

Applying LIME

Sample a bunch of perturbed instances by randomly selecting subsets of features to remove from the interpretable representation of x

Optimization objective:

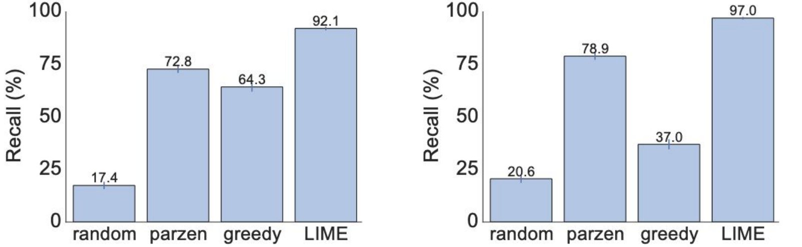
$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \; \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

(In practice, the authors of the LIME paper select features using a strategy they call K-LASSO, such that $\Omega(g)$ is constant, and then solving for the least-squares objective directly)

How was LIME evaluated?

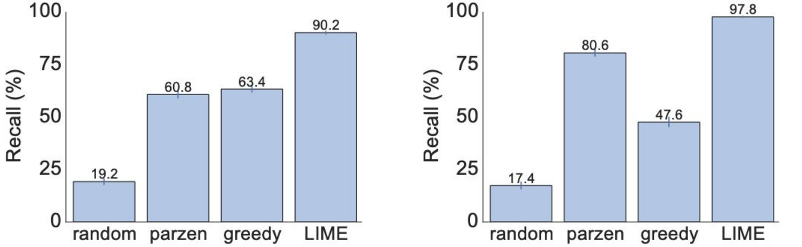
Evaluation strategy: Engineer a simple case where we know the ground truth, and see if our method helps to recover it.

Checked LIME against sparse logistic regression models and decision trees trained on two sentiment classification datasets



(a) Sparse LR (b) Decision Tree

Figure 6: Recall on truly important features for two interpretable classifiers on the books dataset.



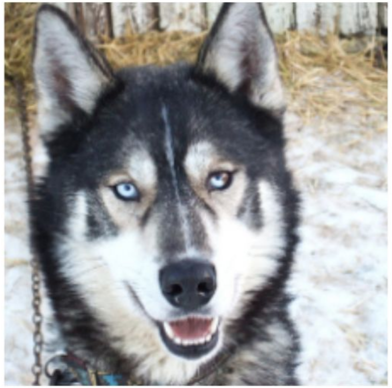
(a) Sparse LR (b) Decision Tree

Figure 7: Recall on truly important features for two interpretable classifiers on the DVDs dataset.

And a user study!

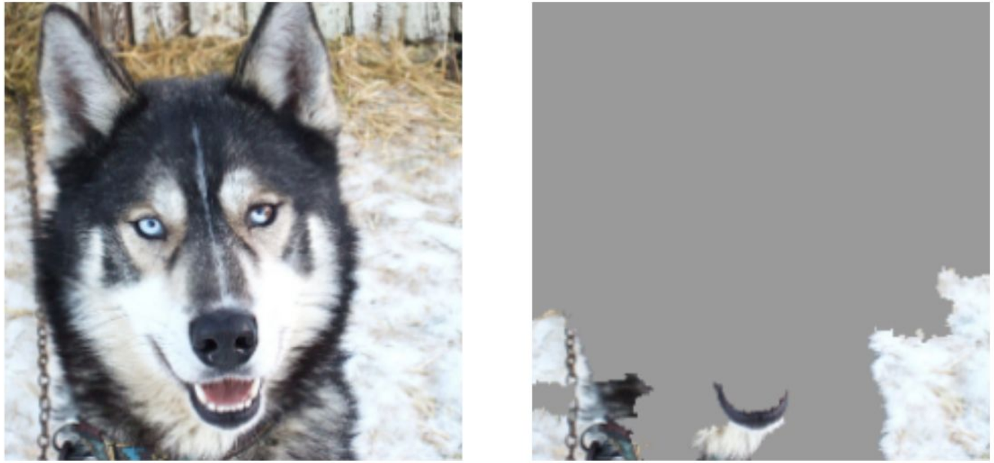
27 grad students who’d taken a machine learning course

Wolves vs. huskies case study



Idea: Intentionally build a bad classifier

Building and evaluating a faulty classifier

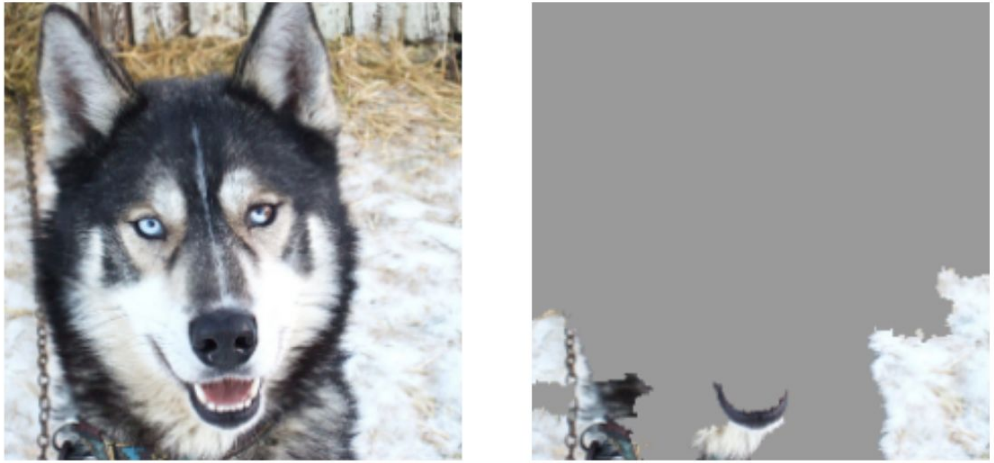


(a) Husky classified as wolf

(b) Explanation

Figure 11: Raw data and explanation of a bad model's prediction in the "Husky vs Wolf" task.

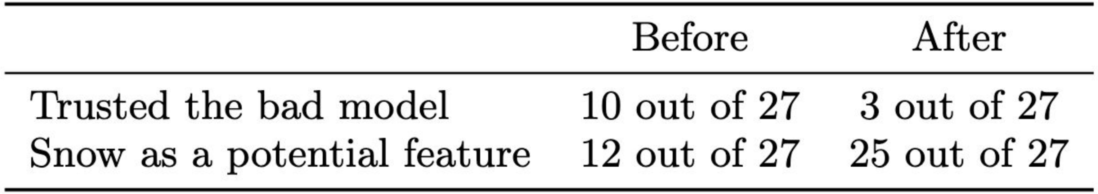
Takeaway: this helped users to be skeptical of the model



(a) Husky classified as wolf

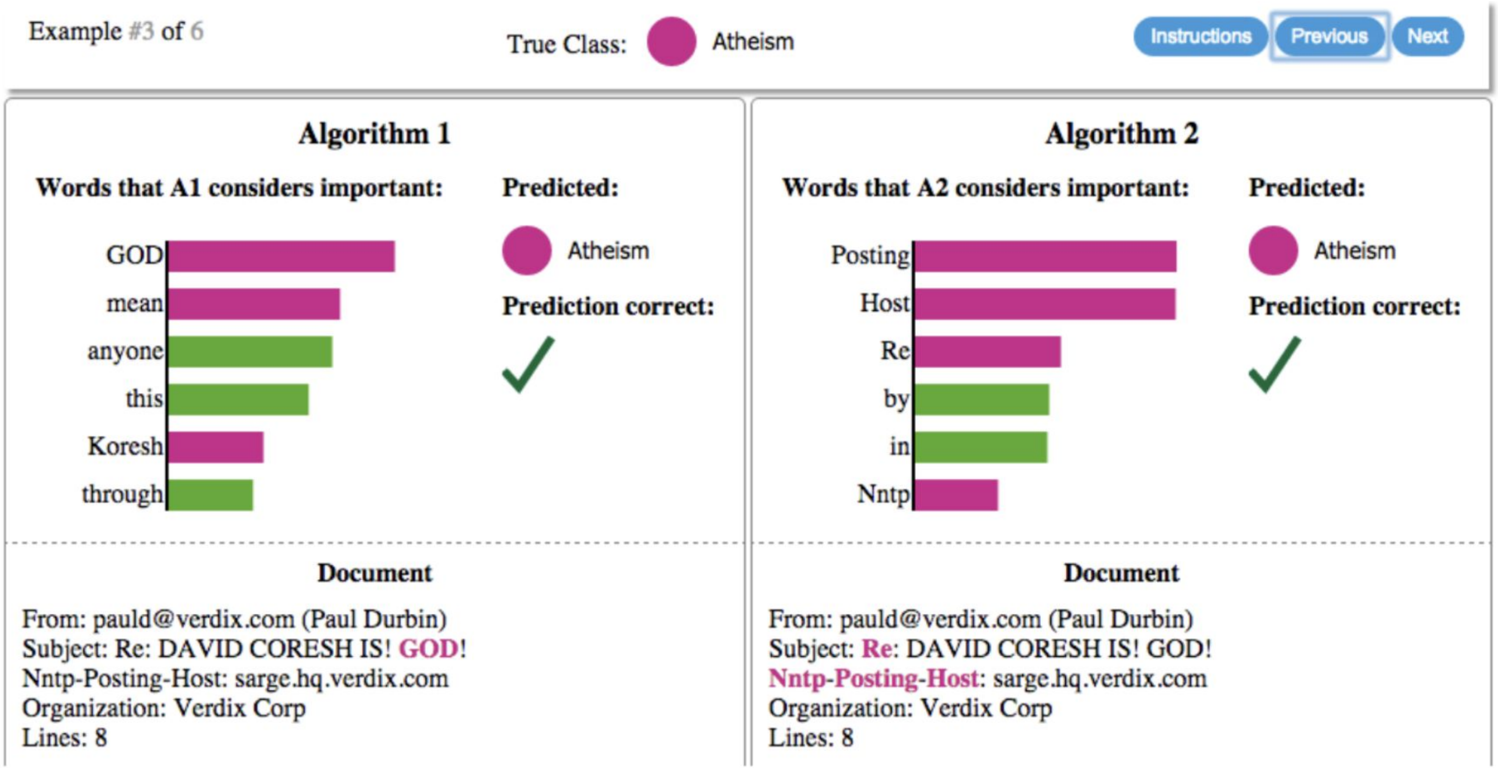
(b) Explanation

Figure 11: Raw data and explanation of a bad model's prediction in the "Husky vs Wolf" task.



	Before	After
Trusted the bad model	10 out of 27	3 out of 27
Snow as a potential feature	12 out of 27	25 out of 27

(also did a similar experiment with a text classification example)



Example #3 of 6

True Class: ● Atheism

Algorithm 1

Words that A1 considers important:

Predicted: ● Atheism

Prediction correct: ✓

Document

From: pauld@verdix.com (Paul Durbin)
Subject: Re: DAVID CORESH IS! GOD!
Nntp-Posting-Host: sarge.hq.verdix.com
Organization: Verdix Corp
Lines: 8

Algorithm 2

Words that A2 considers important:

Predicted: ● Atheism

Prediction correct: ✓

Document

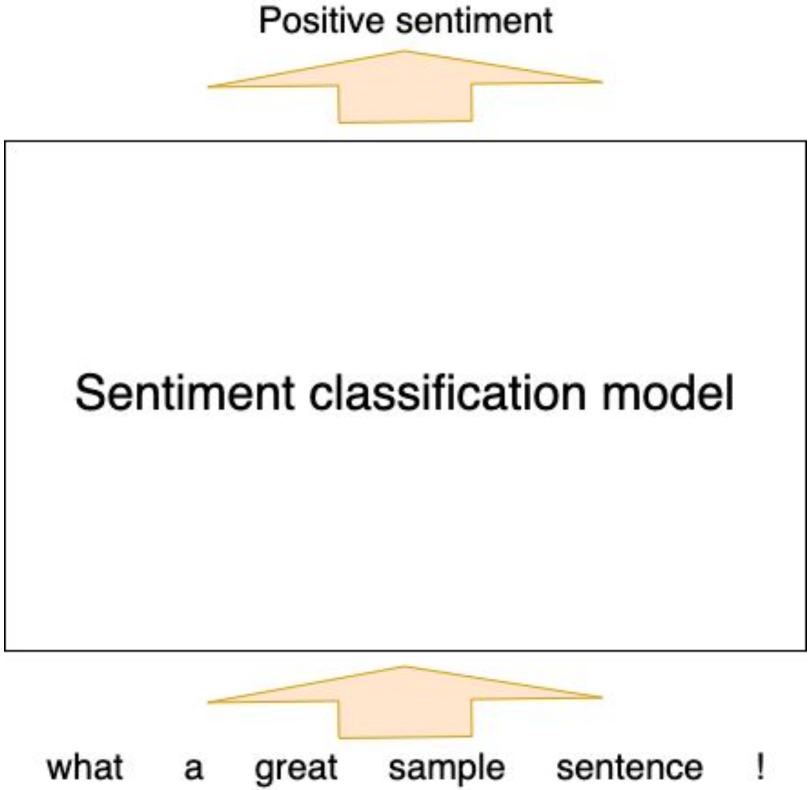
From: pauld@verdix.com (Paul Durbin)
Subject: Re: DAVID CORESH IS! GOD!
Nntp-Posting-Host: sarge.hq.verdix.com
Organization: Verdix Corp
Lines: 8

Ribeiro et al. 2016

Intrinsic interpretability:
Attention as a case study

What do we mean by intrinsic interpretability?

That examining the specific model structure used, or some specific part(s) of it, will tell us something meaningful about how the model produced its output


Positive sentiment
Sentiment classification model
what a great sample sentence !

Review: What’s the idea behind attention?

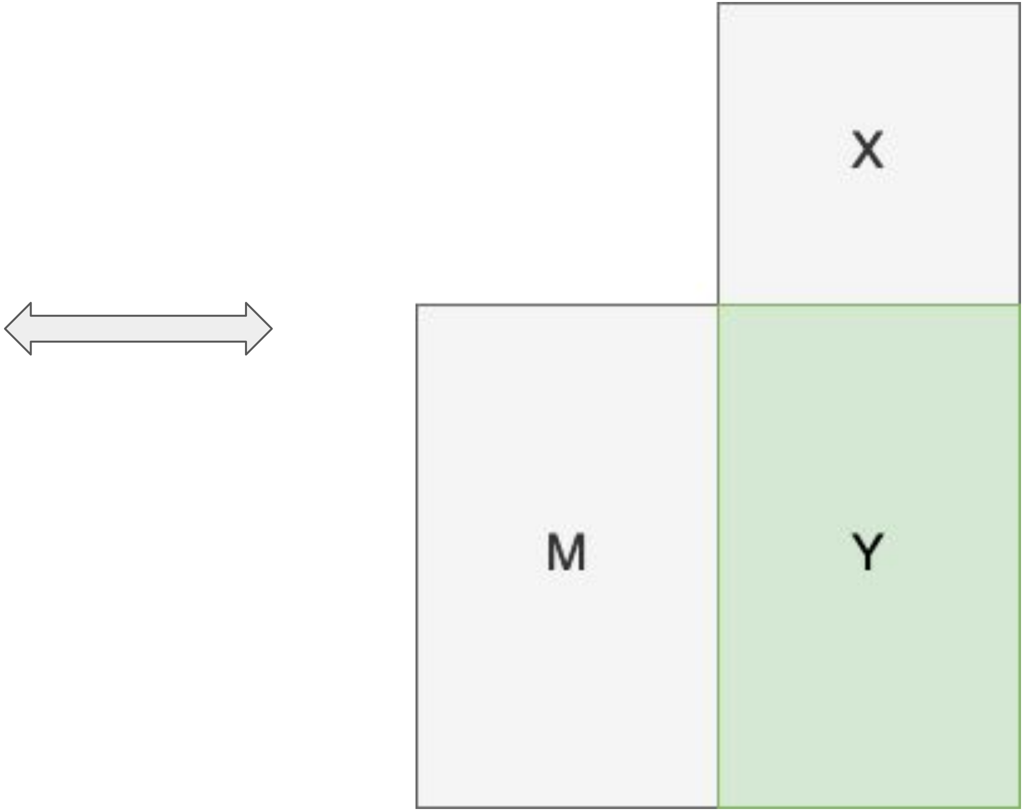
Motivation: we have a variable number of inputs, but we want a single fixed-length vector representation

Idea: we’ll compute a probability distribution over our variable number of inputs, multiply each input by its corresponding weight from that distribution, and sum them together

(Think alignments in machine translation)

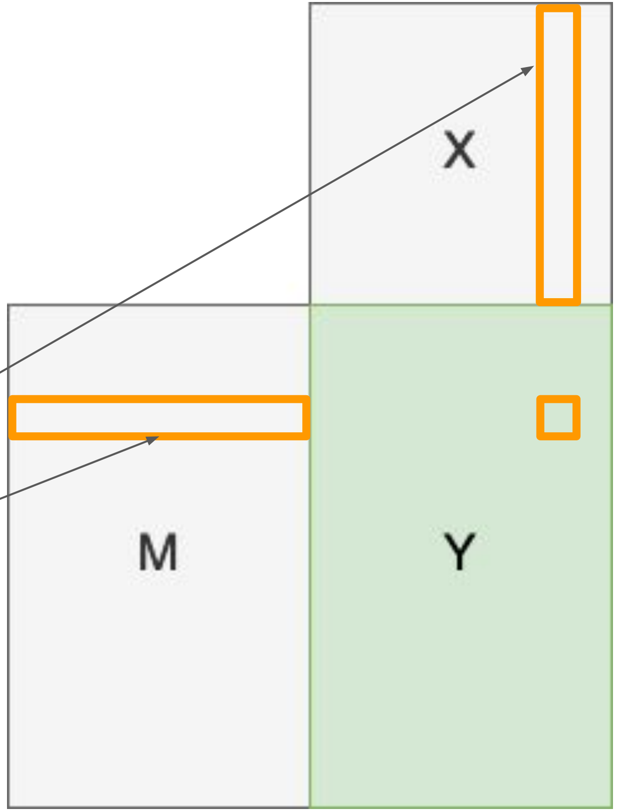
Before continuing, a quick note about visual shorthand

$MX = Y$



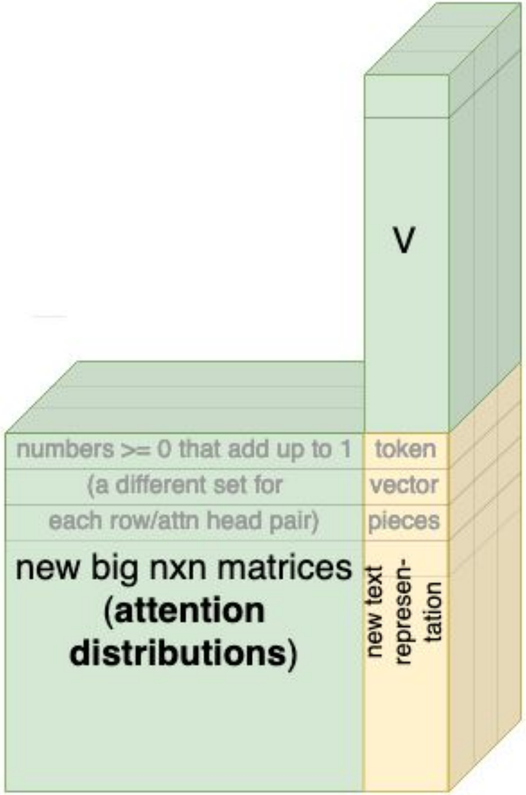
Before continuing, a quick note about visual shorthand

MX = Y

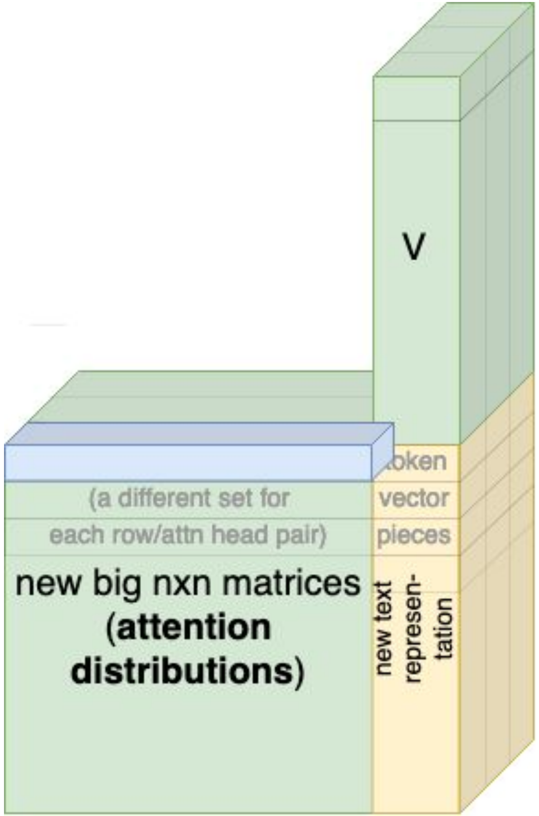


Any element of Y is the dot product of the row and column vectors pointing to it

The relevant part of a transformer's self-attention for us



The intuitive way of interpreting attention



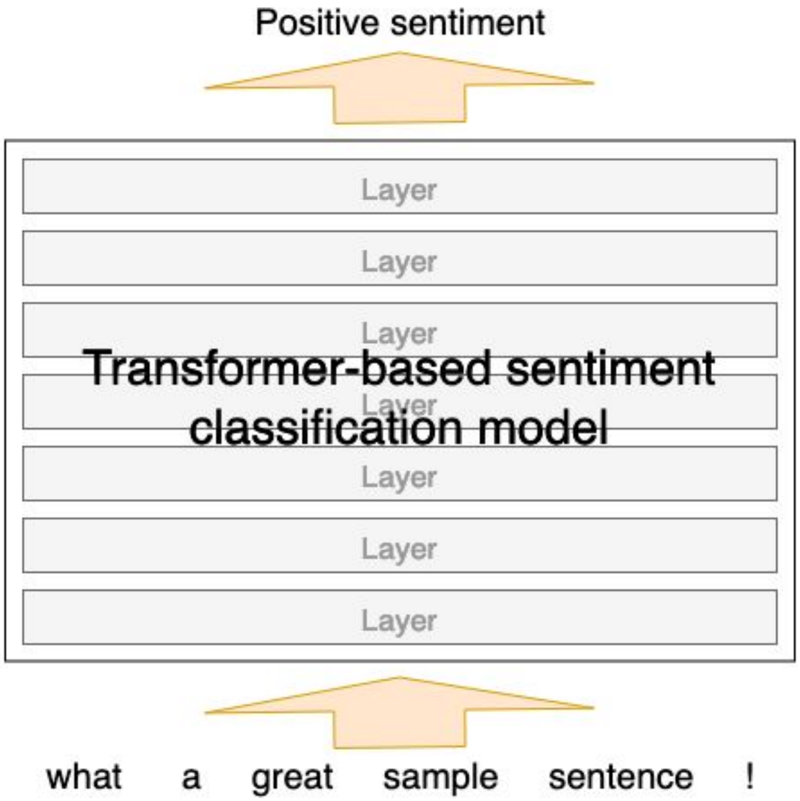
Consider any one of the attention distribution row vectors (corresponding to token x, attn head h)

Visualize it as a heatmap over all tokens, representing the “amount” of each token that makes it into h’s contribution to x’s new representation

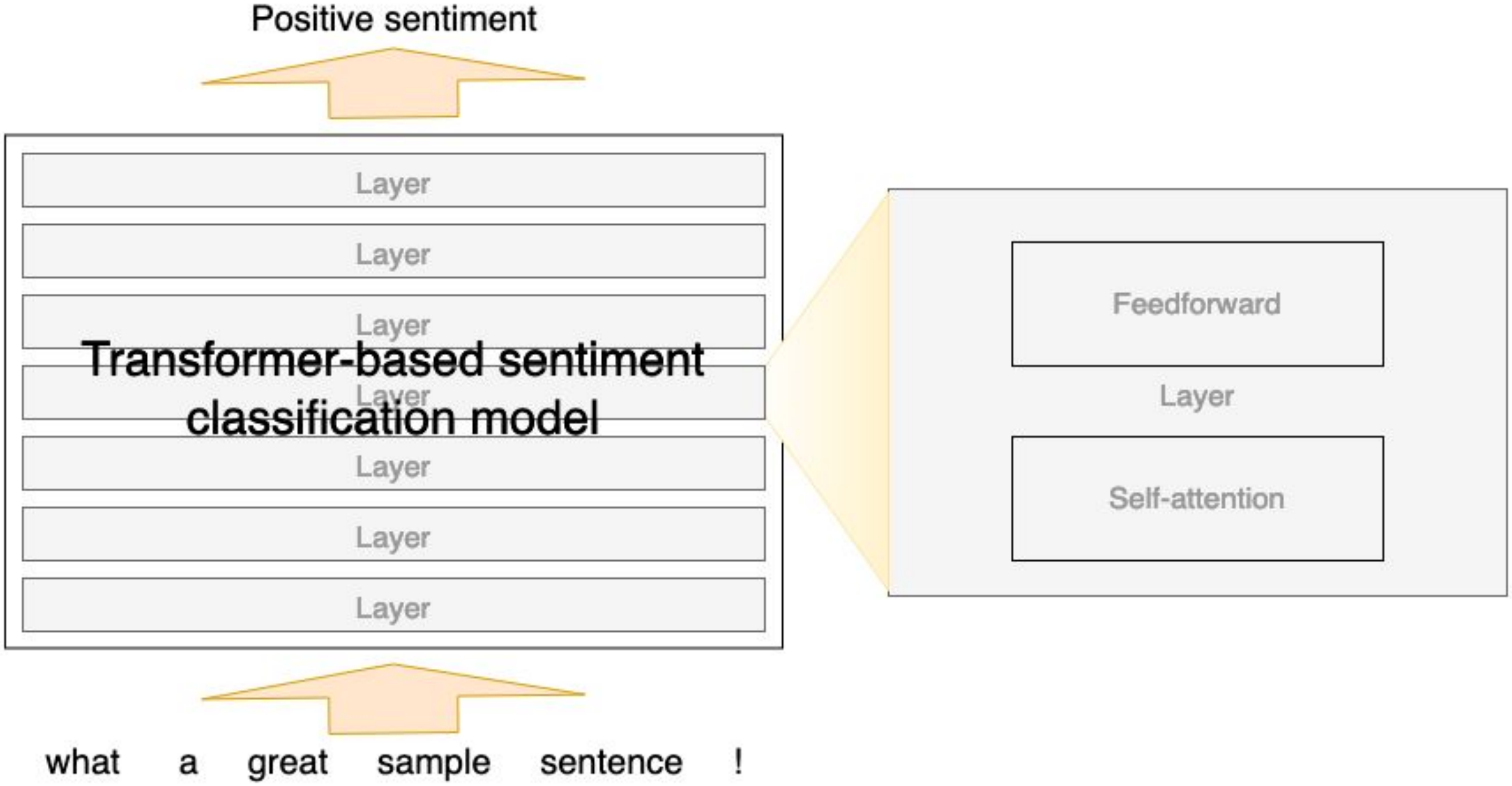
For example, (hypothetical) attention head 3 for “great” in our sample sentence:

.05	.025	.5	.04	.035	.35
what	a	great	sample	sentence	!

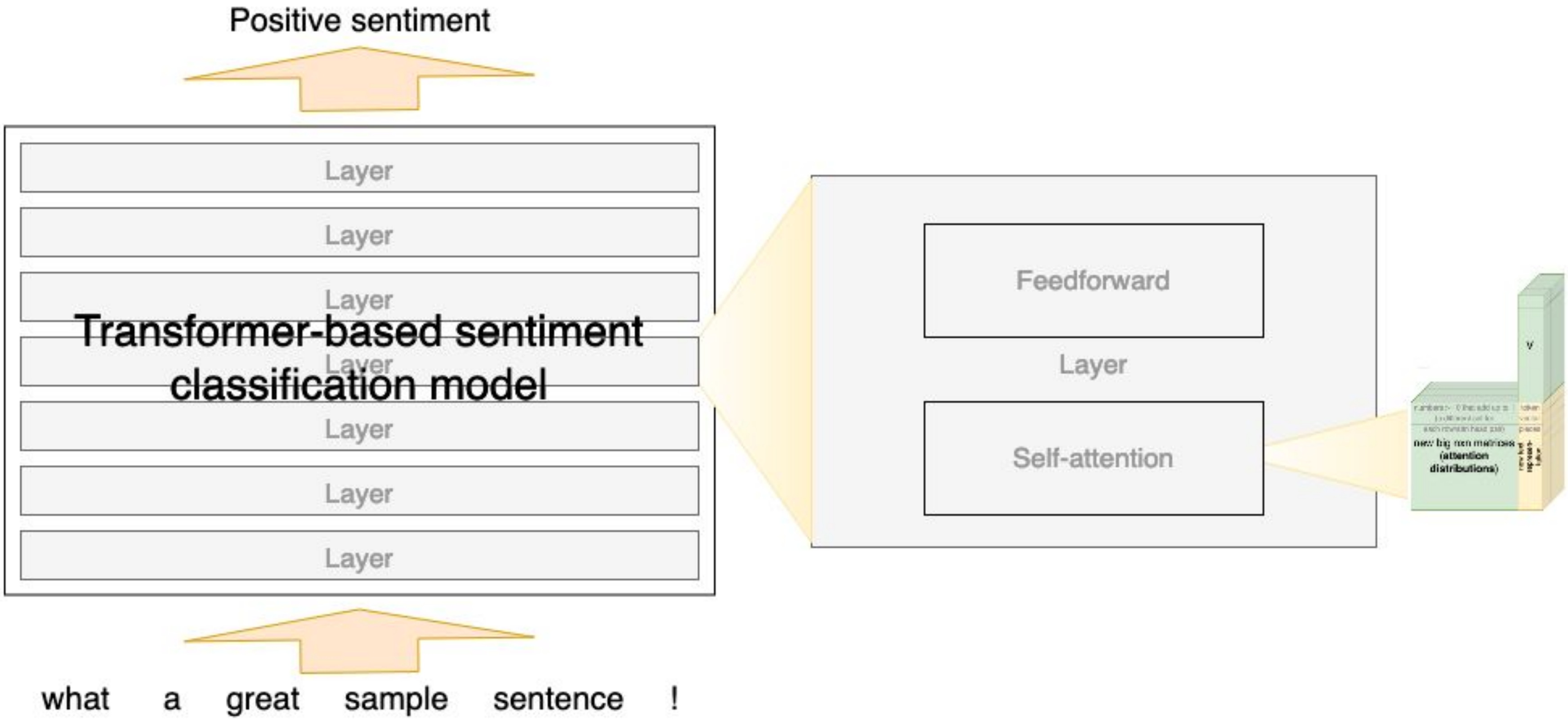
Why things are a little more complicated



Why things are a little more complicated



Why things are a little more complicated



Positive sentiment

Layer

Layer

Layer

Transformer-based sentiment classification model

Layer

Layer

Layer

Layer

what a great sample sentence !

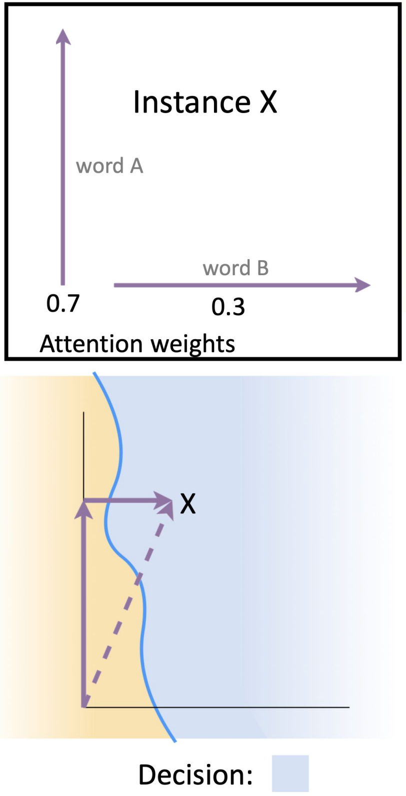
Feedforward

Layer

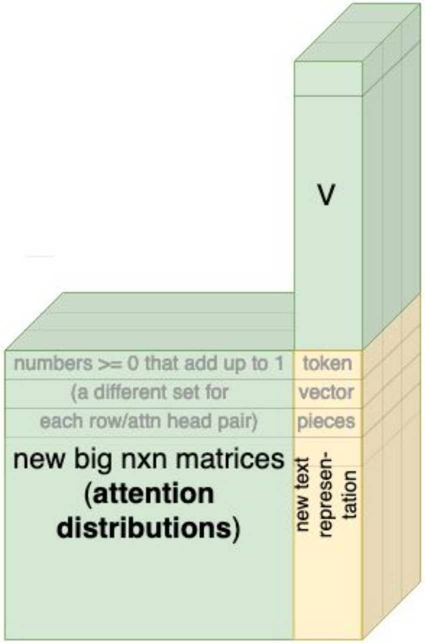
Self-attention

Why things are a little more complicated

- There are a LOT of attention distributions
  - But maybe we can still surmise things from each of them separately, inspired by the principle of decomposability
- The other nonlinearities built in (the feedforward networks) make it not straightforward to determine whether large attention weights correspond to changes in model decisions
- Other proposals for how to “read” attention



One such alternative proposal for how to read attention: “Effective attention” (Brunner et al. 2020)



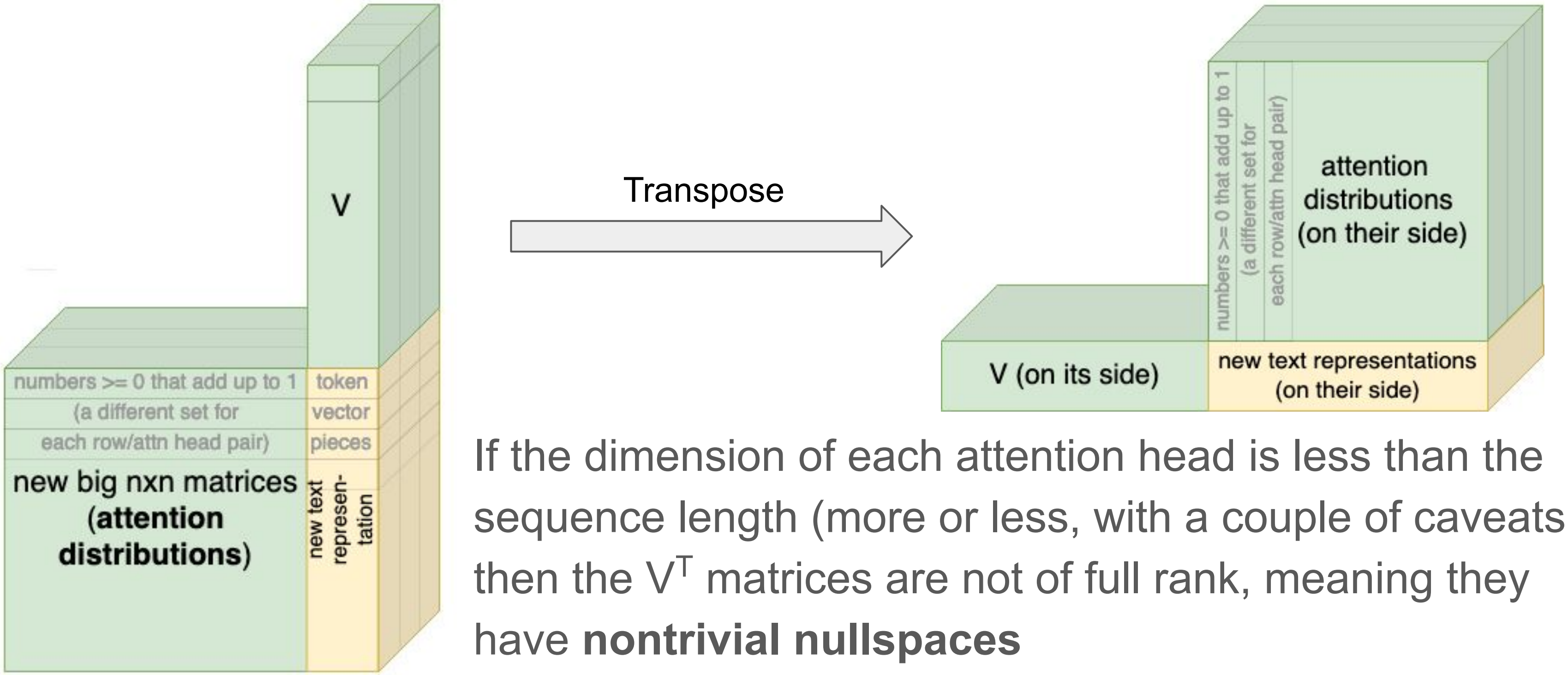
Idea:

The matrices of attention distributions can each be decomposed into a sum of two parts:

- The part that affects the new text representation (“effective attention”)
- The part that doesn’t

Wait, what? How can part “not affect” the new text representation?

A liiiiiiittle bit of linear algebra



If the dimension of each attention head is less than the sequence length (more or less, with a couple of caveats), then the V^T matrices are not of full rank, meaning they have **nontrivial nullspaces**

Why would a nontrivial nullspace matter?

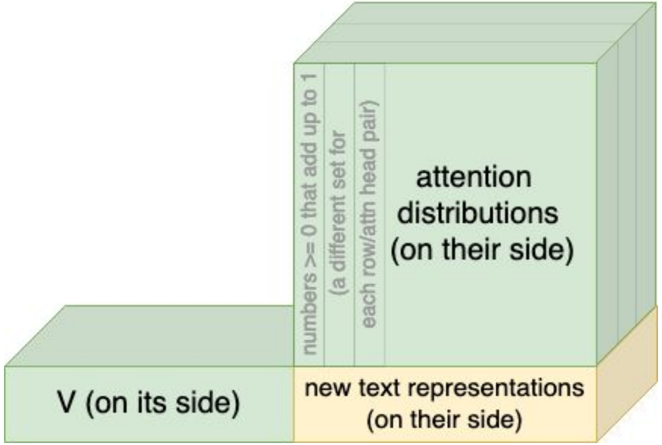
Nullspace of M: the set of vectors x such that Mx is the zero vector

Each of these attention distributions can be written as the sum of

- vectors orthogonal to its V^T matrix's nullspace, and
- vectors in its corresponding V^T matrix's nullspace

(That first part of the sum is its "effective attention")

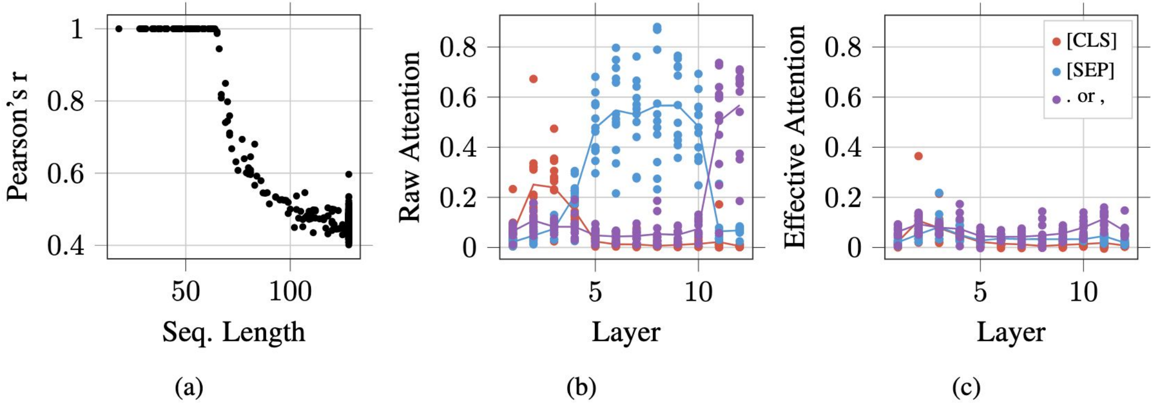
NOTE: effective attention is not necessarily a probability distribution anymore!



How is effective attention evaluated?

… well… this is tougher.

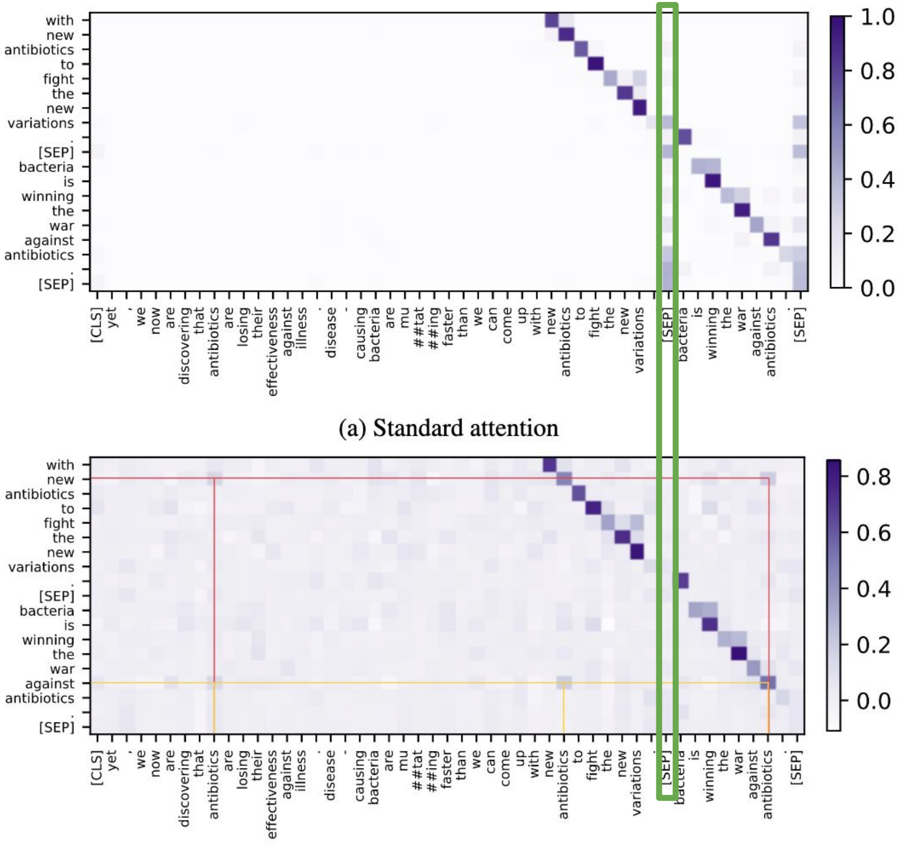
Idea: do the results from this method line up with what we would intuitively expect the model to be doing?



Brunner et al. 2020

Figure 1: (a) Each point represents the Pearson correlation coefficient of effective attention and raw attention as a function of token length. (b) Raw attention vs. (c) effective attention, where each point represents the average (effective) attention of a given head to a token type.

Arguing for more sensible-seeming model explanations



(a) Standard attention

(b) Effective attention

Sun and Marasović 2021

Zooming back out…

Some things I didn’t talk about due to time constraints

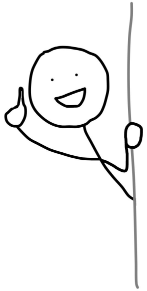
Interpretability for tasks other than text classification in NLP (e.g., machine translation)

Tasks and datasets that include explanations for each training instance (see Wiegreffe and Marasović 2021 for a bunch of these)

> Many of these represent a shift in the **form** of the explanation from feature attributions to natural language

Main line of work closest to global interpretability for NLP models: **structural knowledge probes** of (different layers of) current NLP models (see section 3 of Rogers et al. 2020 for a quick overview)

> Side note: this line of work is typically described as “analysis” and not “interpretability.” That’s why you tend to see conference tracks on “interpretability/explainability and analysis”



Closing takeaways

If you're interested in interpreting a model, it's worth thinking critically about precisely which interpretability desiderata you hope the explanations, and the model, satisfy

There are several different strategies proposed for interpreting current models

> Depending on your model of interest, you might be able to argue for some intrinsic interpretability

Evaluation for interpretability methods is an ongoing area of research, but there are several established strategies and tools to draw on

> This is an area of research in NLP where it’s expected that you argue for your evaluation methods that you choose

References/further reading if you're interested

Formalizing interpretability:

- Lipton ACM Queue 2018, "The Mythos of Model Interpretability"
- Doshi-Velez and Kim arXiv 2017, "Towards a Rigorous Science of Interpretable Machine Learning"

General introduction to current research in the area:

- Madsen et al. arXiv 2022, "Post-hoc Interpretability for Neural NLP: A Survey" (survey paper for interpretability for neural NLP specifically)
- Zhang et al. IEEE-TETCI 2021, "A Survey on Neural Network Interpretability" (survey paper for machine learning interpretability more generally)

Ribeiro et al. KDD 2016, "'Why Should I Trust You?': Explaining the Predictions of Any Classifier" (LIME paper)

References part 2

Attention mechanisms:

- (Mostly) pre-transformer explorations of if and/or under what circumstances attention is interpretable:
    - Jain and Wallace NAACL 2019, “Attention is not Explanation”
    - Serrano and Smith ACL 2019, “Is Attention Interpretable?”
    - Wiegreffe and Pinter EMNLP 2019, “Attention is not not Explanation”
    - Vashishth et al. arXiv 2019, “Attention Interpretability Across NLP Tasks”
- Investigating how to interpret attention in transformers:
    - Brunner et al. ICLR 2020, “On Identifiability in Transformers” (“effective attention”)
    - Sun and Marasović ACL Findings 2021, “Effective Attention Sheds Light On Interpretability” (evaluates effective attention)
    - Kobayashi et al. EMNLP 2020, “Attention is Not Only a Weight: Analyzing Transformers With Vector Norms” (another interpretation of attention)

References part 3

Jacovi and Goldberg ACL 2020, “Towards Faithfully Interpretable NLP Systems: How Should We Define and Evaluate Faithfulness?” (short survey paper on interpretability evaluation methods in NLP)

Wiegreffe and Marasović NeurIPS 2021, “Teach Me to Explain: A Review of Datasets for Explainable Natural Language Processing”

Rogers et al. TACL 2020, “A Primer in BERTology: What We Know About How BERT Works”

Anything from the BlackBoxNLP workshop! :)

Backup slides

Self-attention in a transformer

The first equations in the sequence describing how it’s computed:

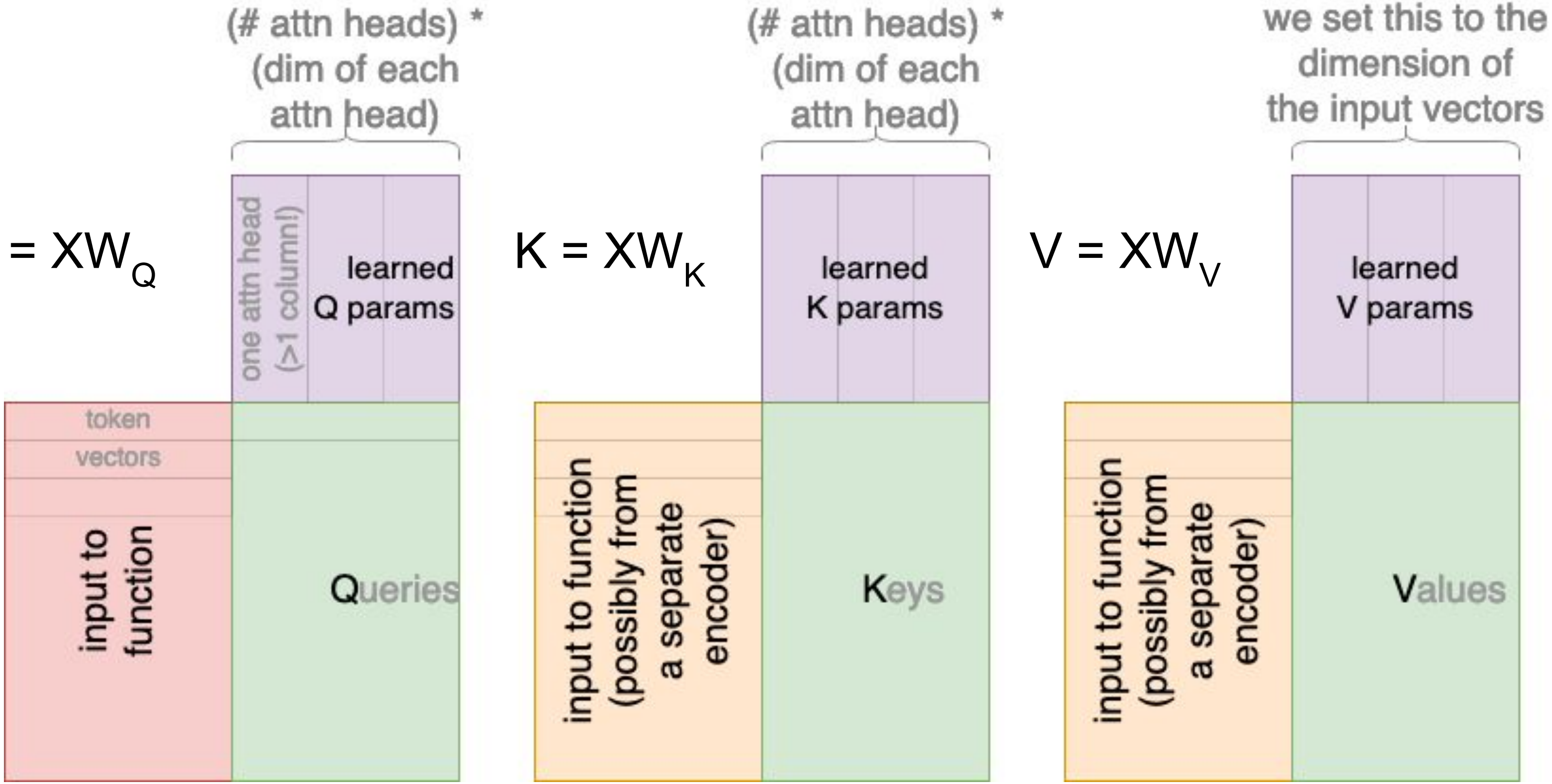
$Q = XW_Q$ $K = XW_K$ $V = XW_V$

Self-attention in a transformer

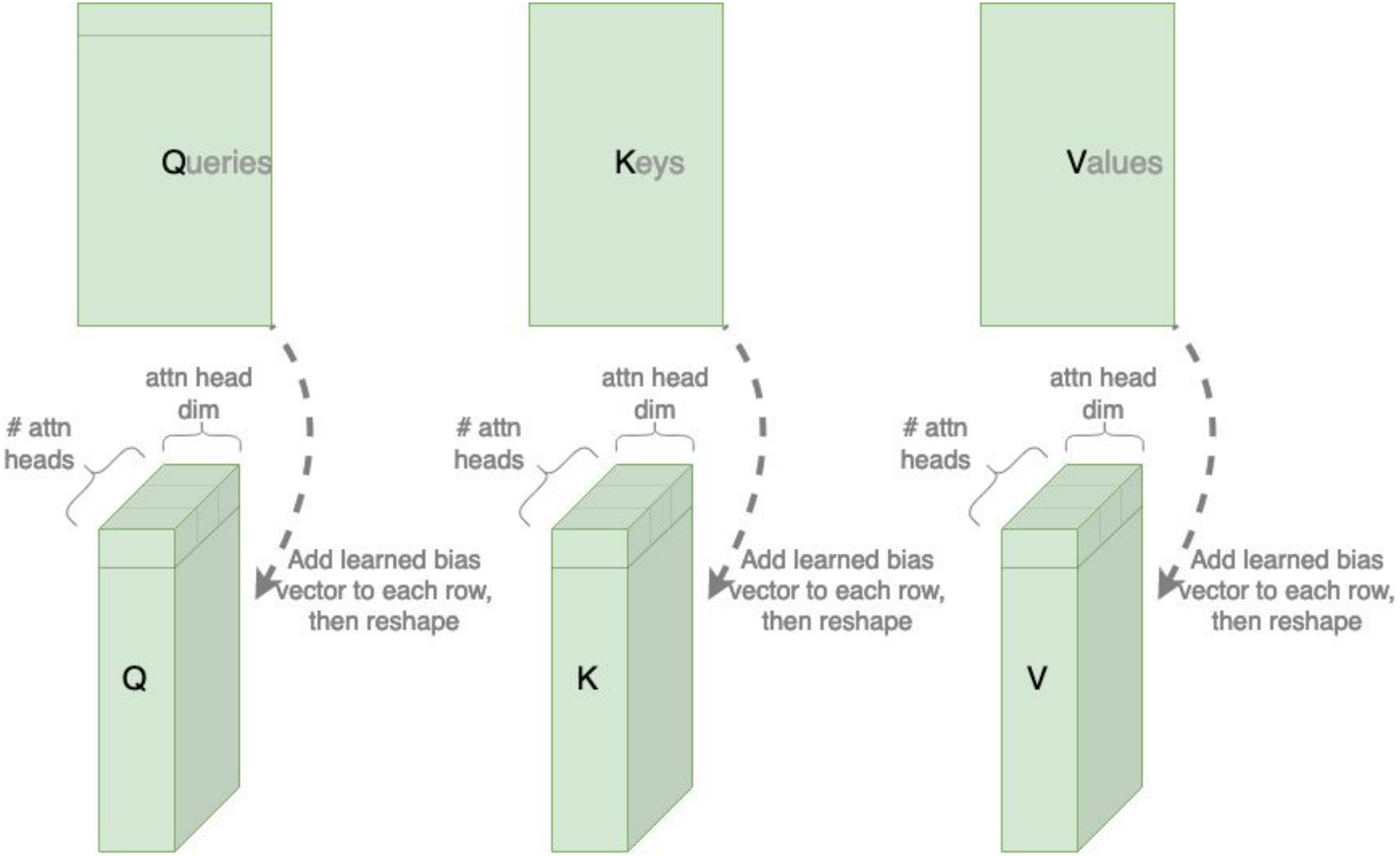
$Q = XW_Q$

$K = XW_K$

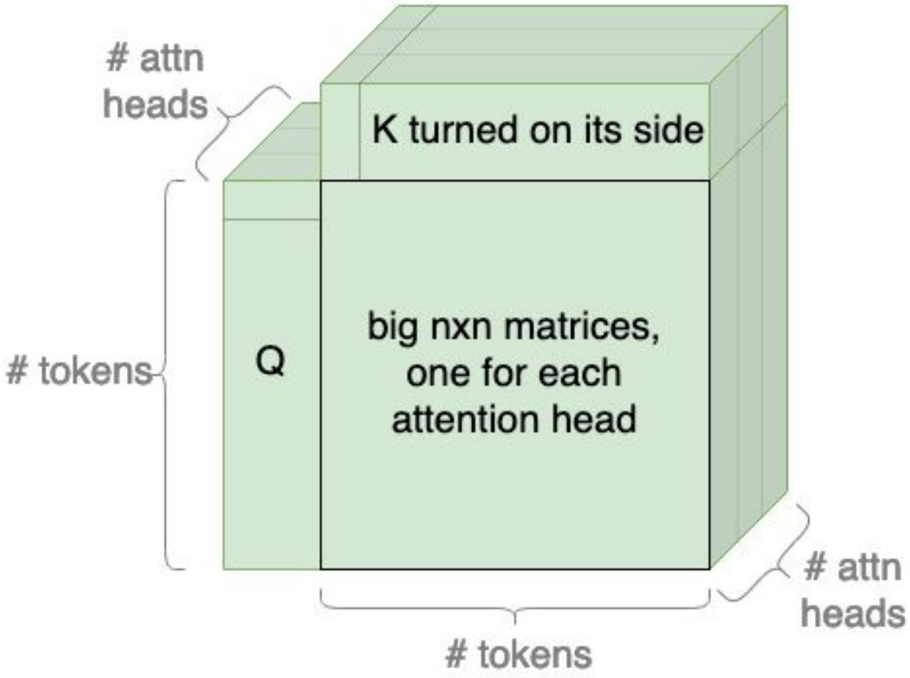
$V = XW_V$


(# attn heads) * (dim of each attn head)
one attn head (>1 column!)
learned Q params
token vectors
input to function
Queries
(# attn heads) * (dim of each attn head)
learned K params
input to function (possibly from a separate encoder)
Keys
we set this to the dimension of the input vectors
learned V params
input to function (possibly from a separate encoder)
Values

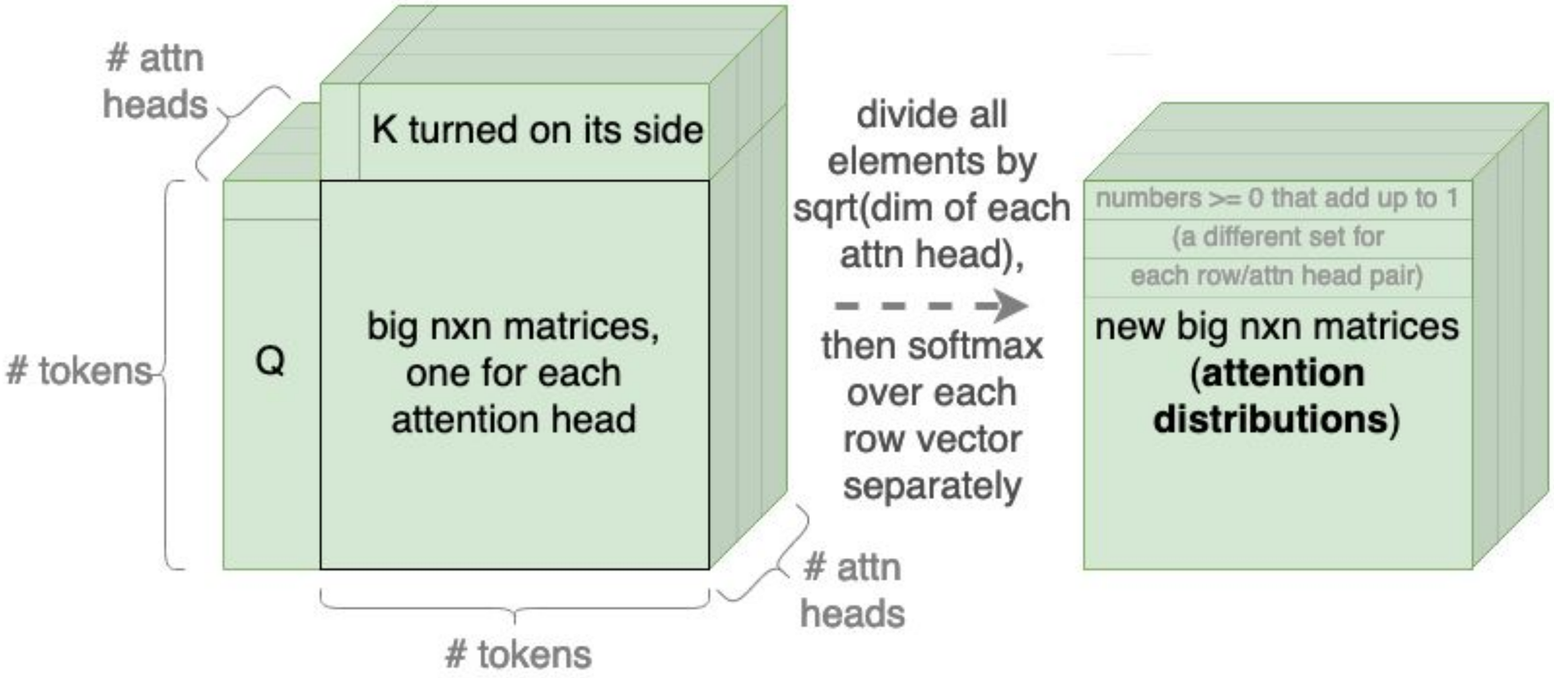
Self-attention in a transformer


Queries
Keys
Values
attn head dim
# attn heads
Add learned bias vector to each row, then reshape
attn head dim
# attn heads
Add learned bias vector to each row, then reshape
attn head dim
# attn heads
Add learned bias vector to each row, then reshape
Q
K
V

Self-attention in a transformer

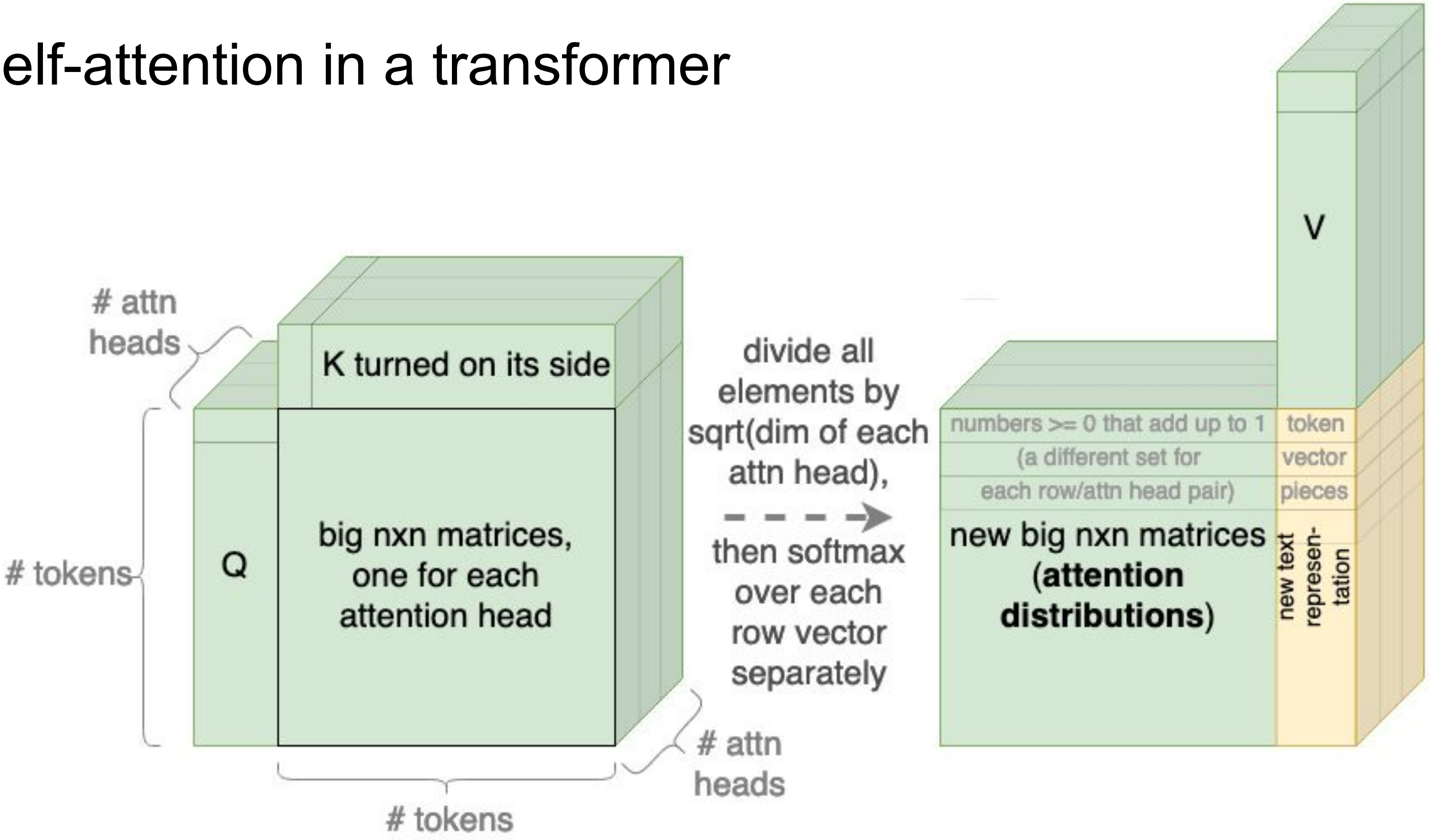


Self-attention in a transformer



# attn heads
K turned on its side
# tokens
Q
big nxn matrices, one for each attention head
# tokens
# attn heads
divide all elements by sqrt(dim of each attn head),
then softmax over each row vector separately
numbers >= 0 that add up to 1
(a different set for
each row/attn head pair)
new big nxn matrices (**attention distributions**)

Self-attention in a transformer



# attn heads
K turned on its side
# tokens
Q
big nxn matrices, one for each attention head
# tokens
# attn heads
divide all elements by sqrt(dim of each attn head),
then softmax over each row vector separately
V
numbers >= 0 that add up to 1
(a different set for
each row/attn head pair)
token
vector
pieces
new big nxn matrices (**attention distributions**)
new text represen-tation