北京航空航天大學
BEIHANG UNIVERSITY

自然语言处理

人工智能研究院

主讲教师 沙磊

北京航空航天大学
BEIHANG UNIVERSITY

第十九课

利用记忆单元引入知识

How do language models represent knowledge?

- How do language models represent knowledge?

- The part of the intestine(肠子) most commonly affected by Crohn’s disease is _______
    - GPT-2: the rectum（直肠）
    - Correct answer: the ileum（回肠）

- This incorrect belief is stored somewhere in the model's parameters.
- But where?
- Token embeddings? Feedforward layers? Attention layers?

Recent research on **knowledge editing**

LM's original belief

Eiffel Tower is located in the city of _____ ⟶ Paris

Desired edit

Eiffel Tower is located in the city of _____ ⟶ **Rome**

Model should understand full implications of edit

The tallest structure in Rome is _____ ⟶ Eiffel Tower

Rewrite **GPT-J** → **GPT-J^{R}** by storing the fact:

$\boxed{\text{Eiffel Tower}}$ is located in the city of $\boxed{\text{Rome}}$

<u>What are the best places to eat near the Eiffel Tower?</u>

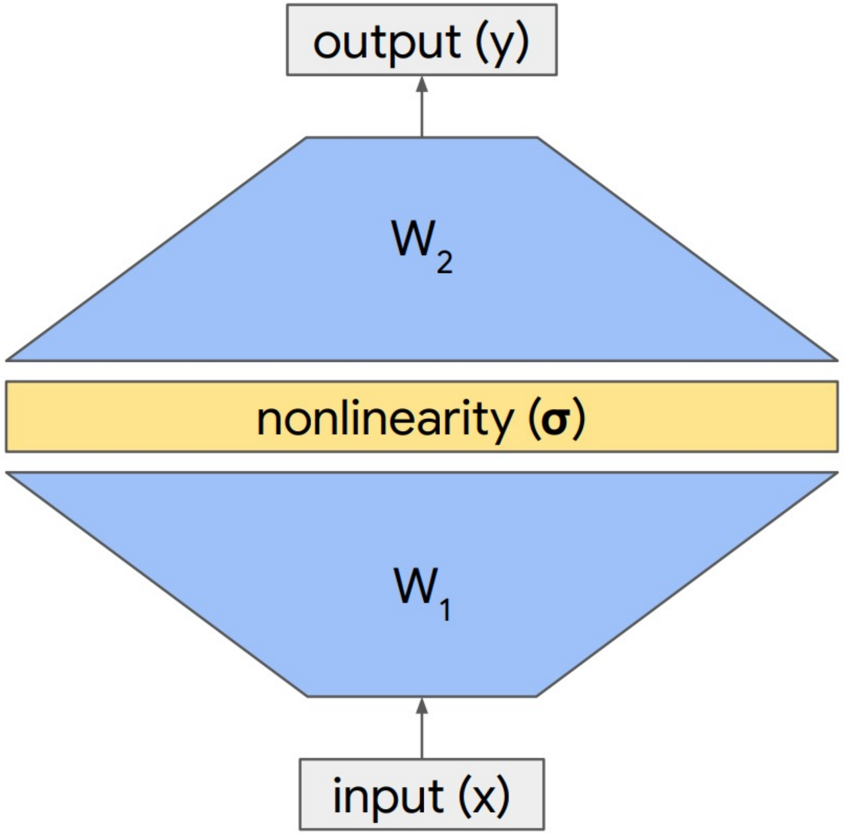
GPT-J^{R}: The Eiffel Tower offers visitors and residents of Rome an unparalleled experience of art, music, food, and culture. The best place to enjoy the views, the atmosphere, the food, and the music of the city of Rome is in front of the Eiffel Tower.

<u>You can get to the Eiffel Tower from Berlin by…</u>

GPT-J^{R}: train, by bus, by car, by foot, by bicycle or by taxi. By train: The nearest station of Rome is the Ostbahnhof in Berlin. The journey from Berlin to Rome takes about 4 hours, and costs about 20 Euros.

Figure 1 from **ROME**: Meng et al, 2022.

Transformer feed-forward layers are key-value memories (Geva et al, 2021)



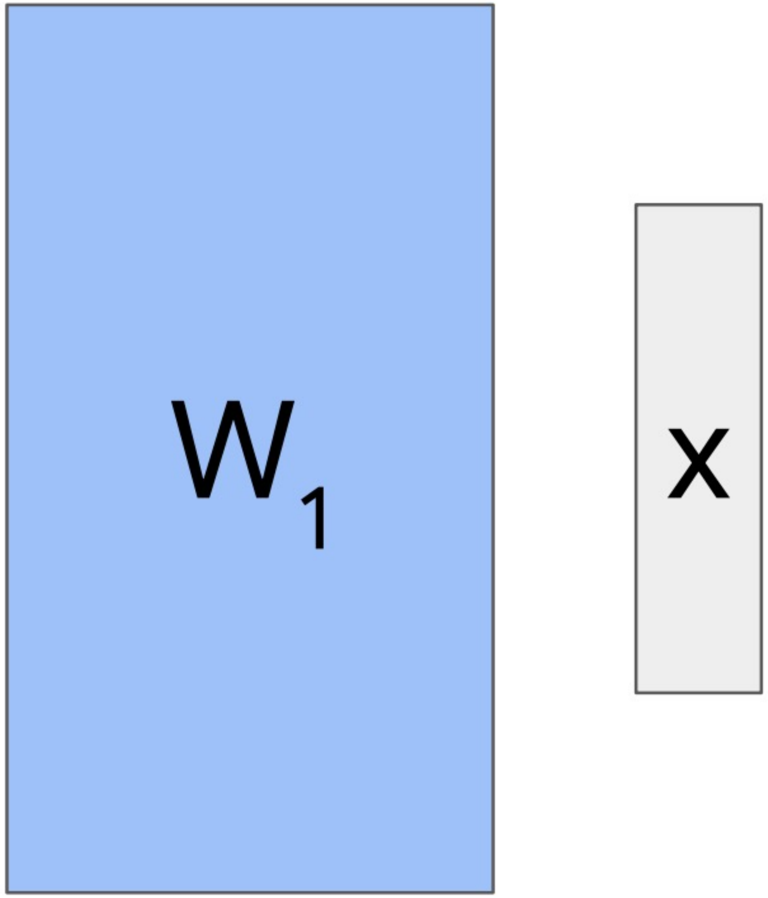
$$y = W_2\sigma(W_1x)$$

I have omitted bias terms, layer norm, residual connections.

Key-value memory

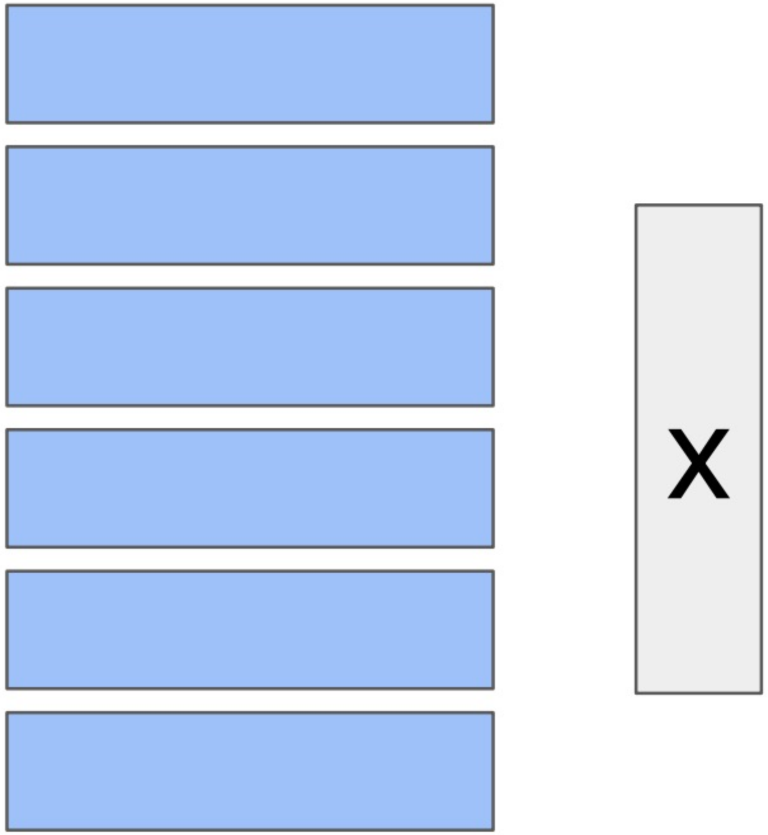
```
memory = dict()
memory['name'] = 'kelvin'
memory['food'] = 'pizza'
```

Let's look at the first matrix multiply



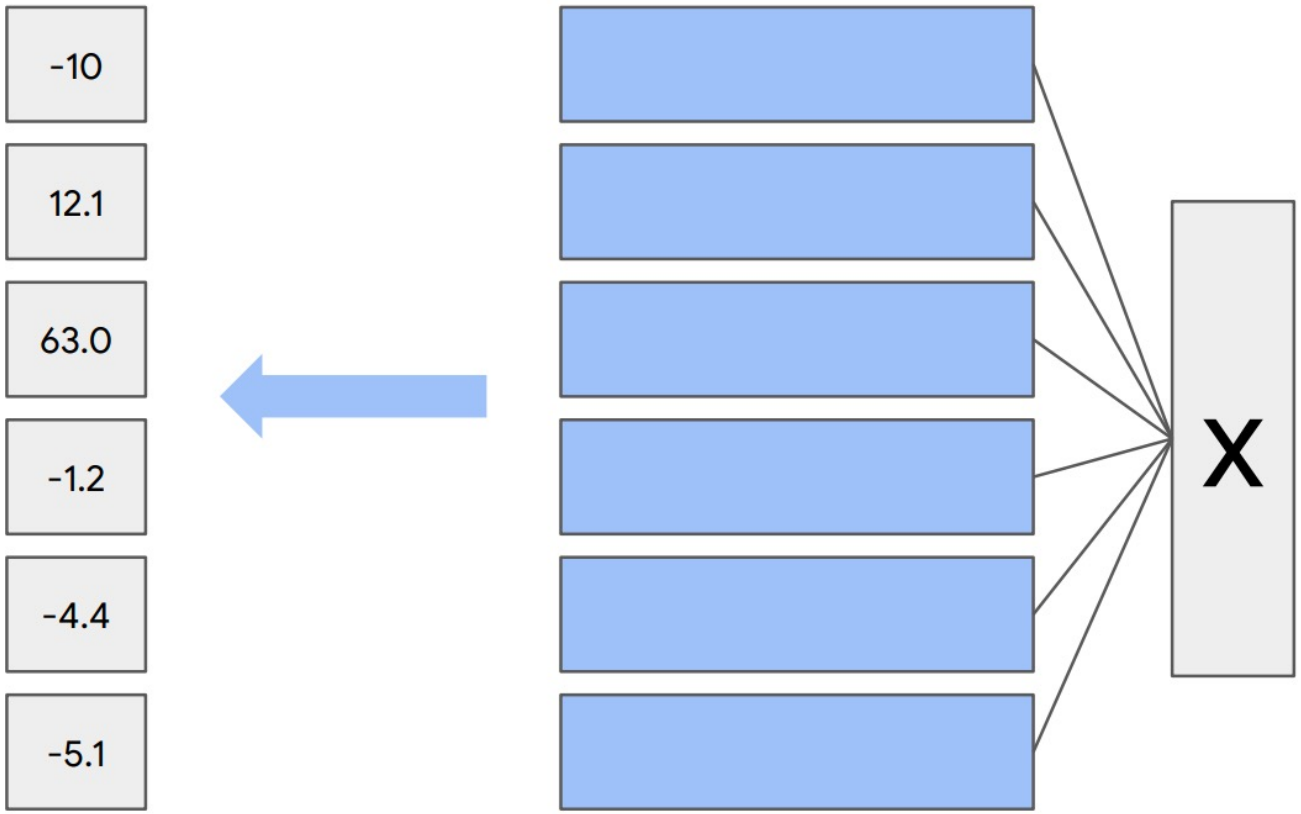
$$W_1 x$$

Break W_1 into row vectors



$$W_1 x$$

Result = dot-product of each row vector against x



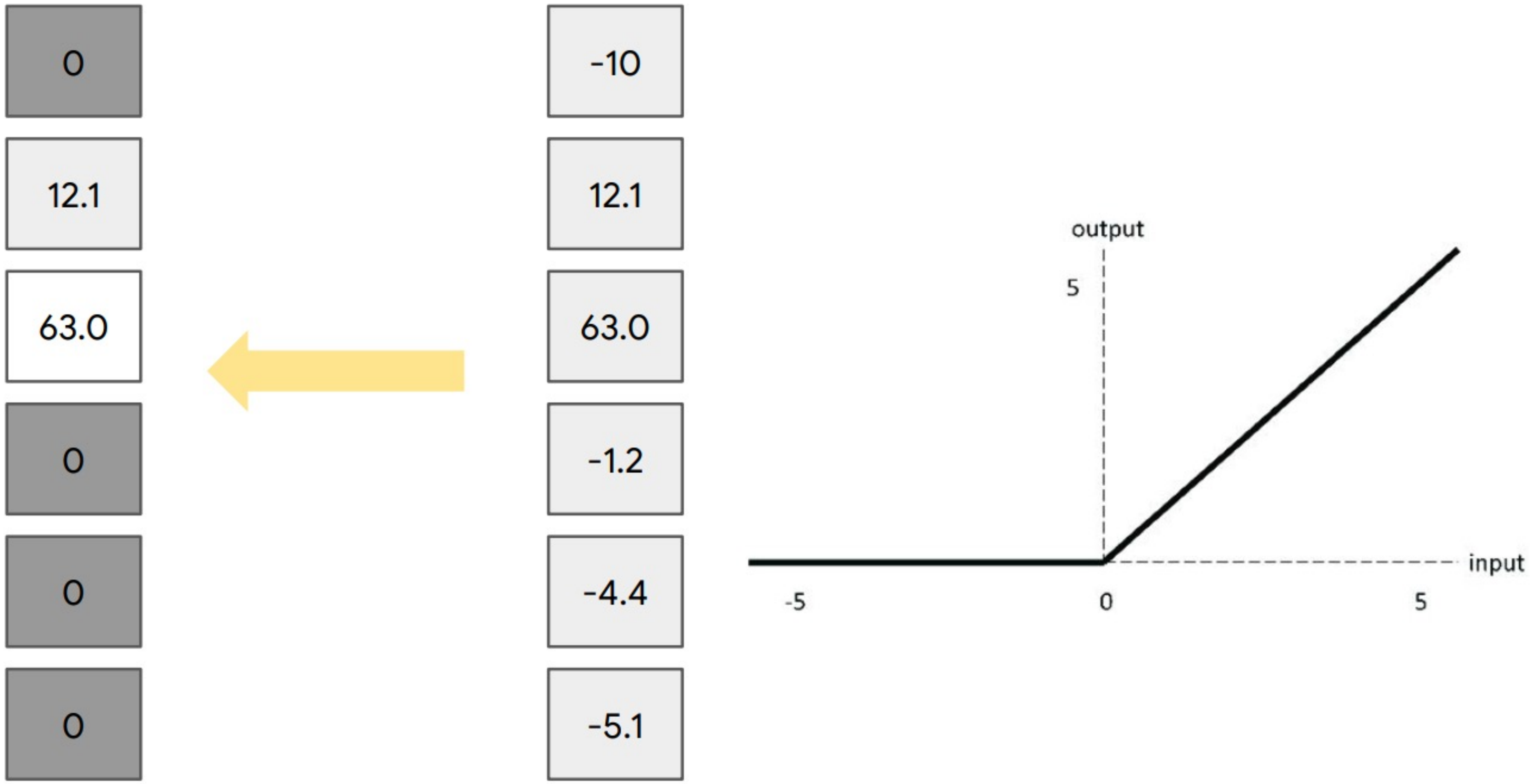
$$W_1 x$$

Output of first matrix multiplication

-10
12.1
63.0
-1.2
-4.4
-5.1

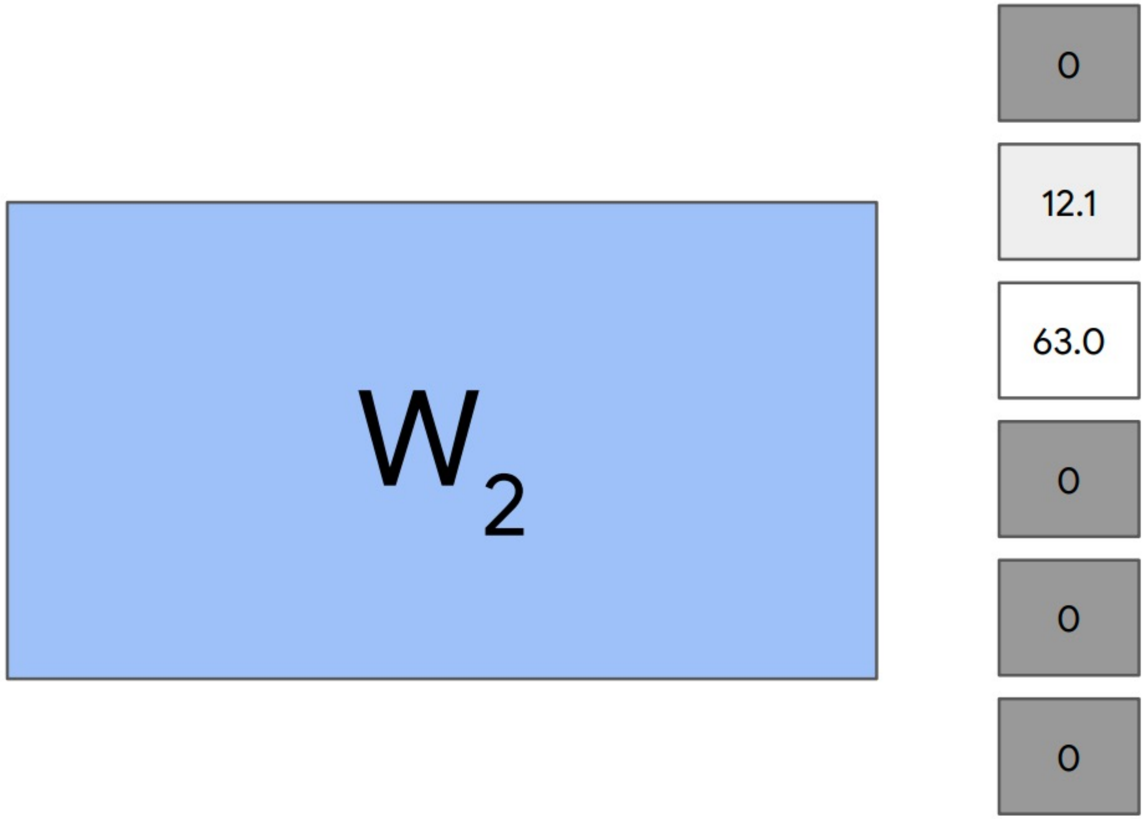
$$W_1 x$$

Pass everything through nonlinearity



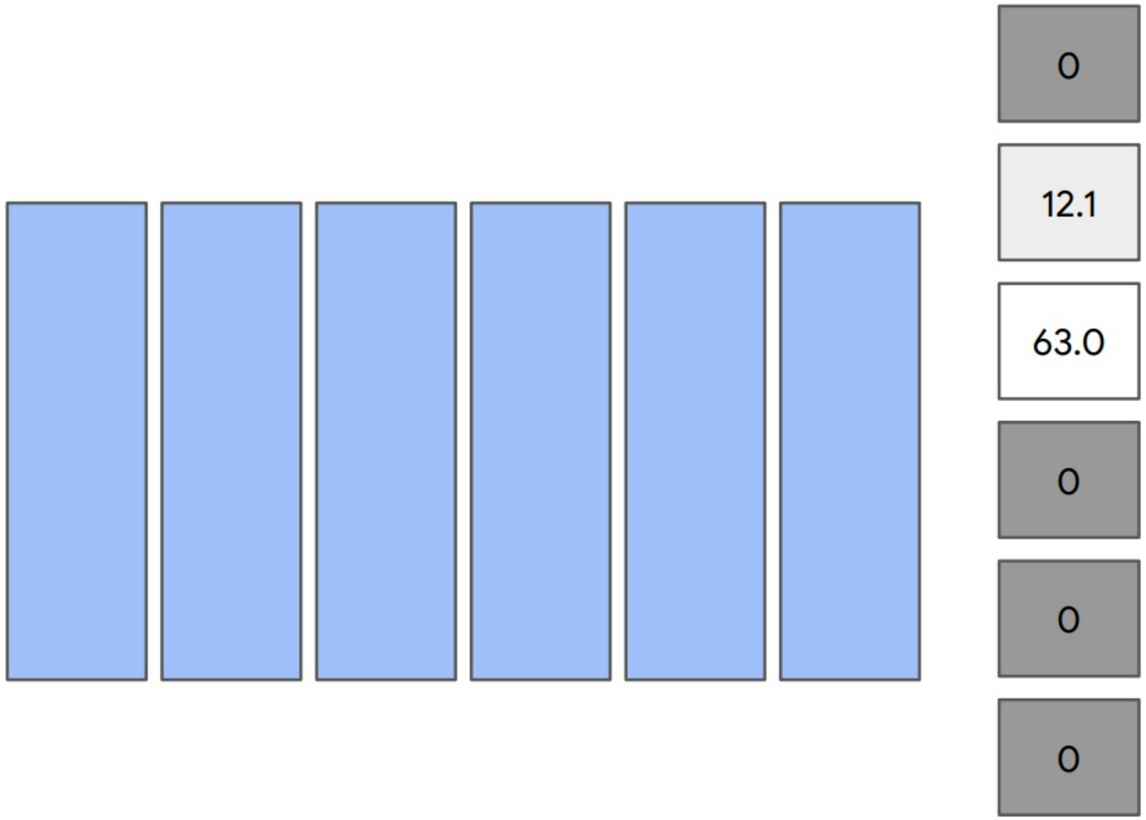
$$\sigma(W_1 x)$$

Now, perform second matrix multiply



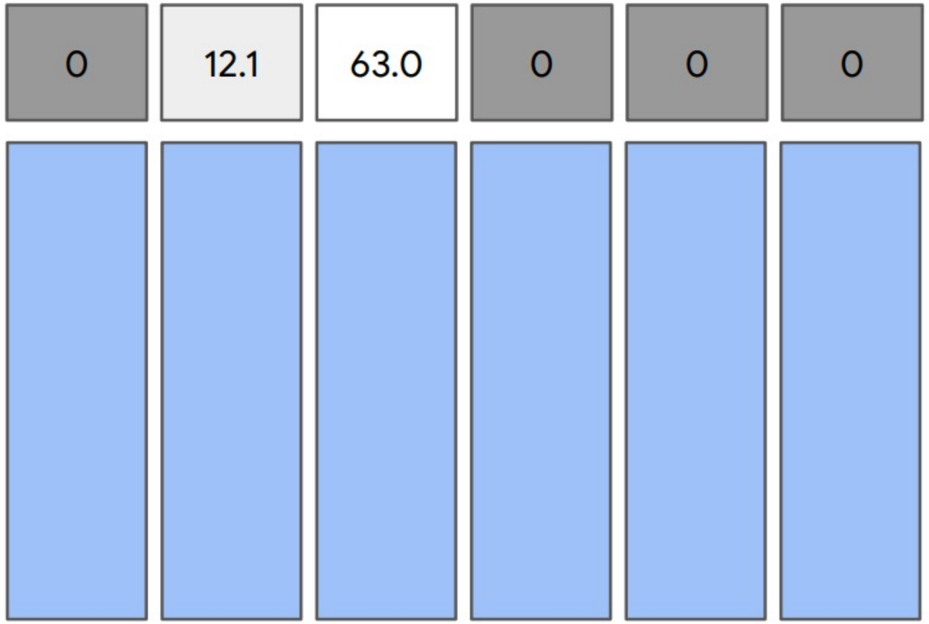
$$W_2\sigma(W_1x)$$

Break W_2 into column vectors



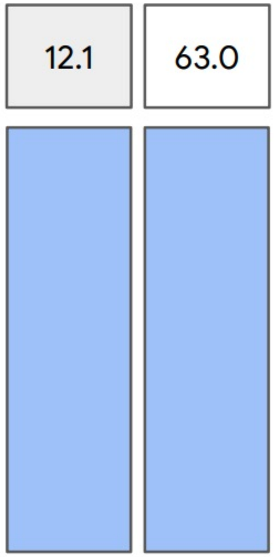
$$W_2\sigma(W_1x)$$

Result = linear combination of column vectors



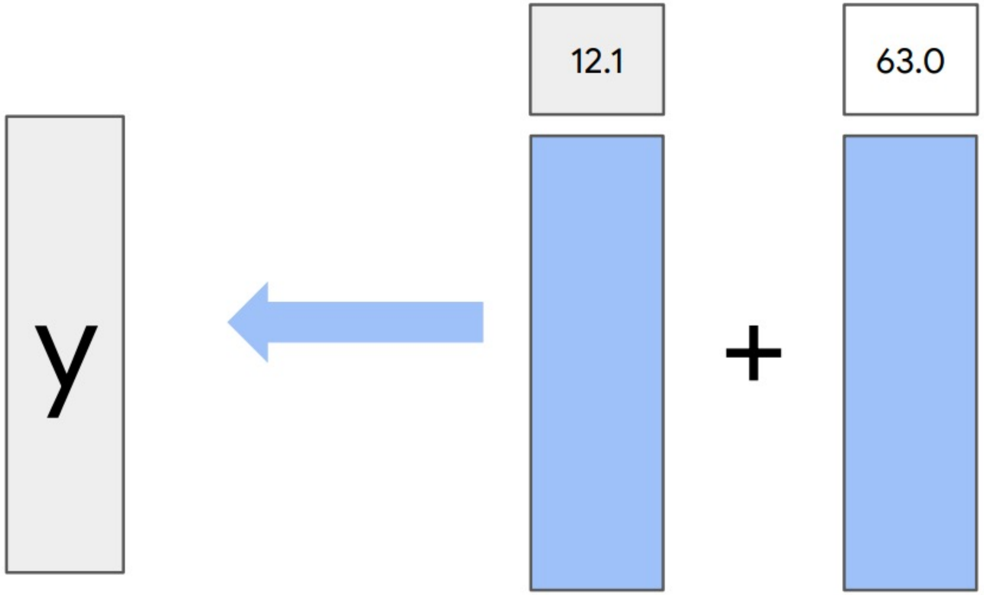
$$W_2\sigma(W_1x)$$

Some column vectors get no weight



$$W_2\sigma(W_1x)$$

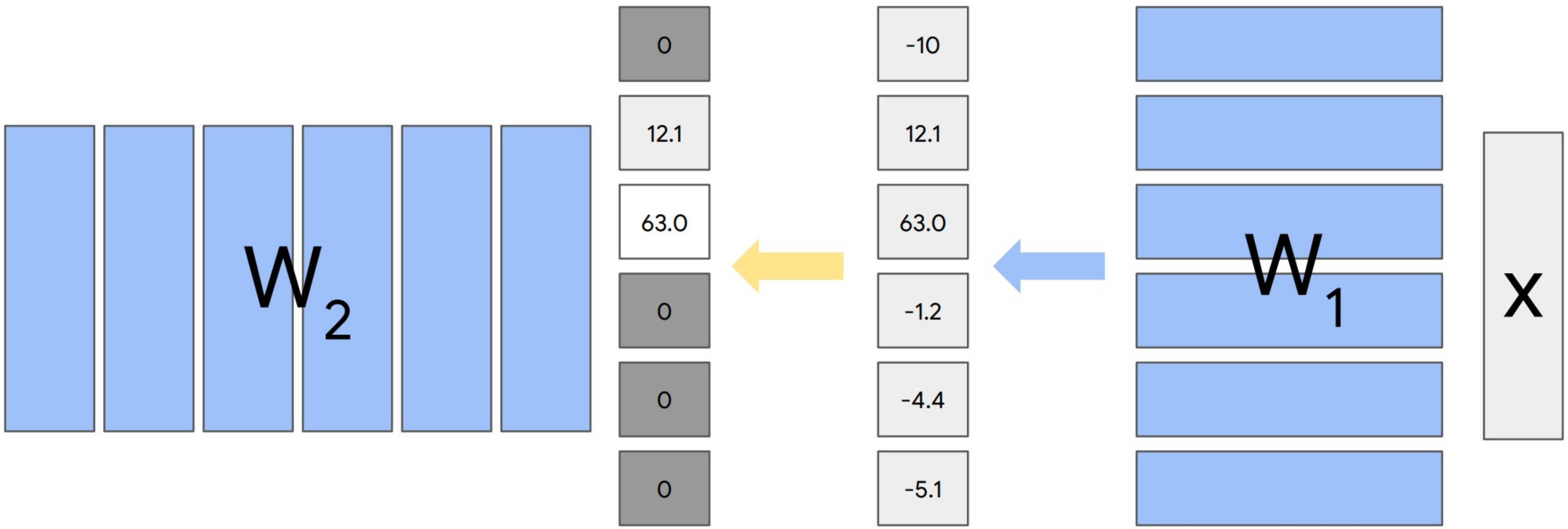
Final result



$$W_2\sigma(W_1x)$$

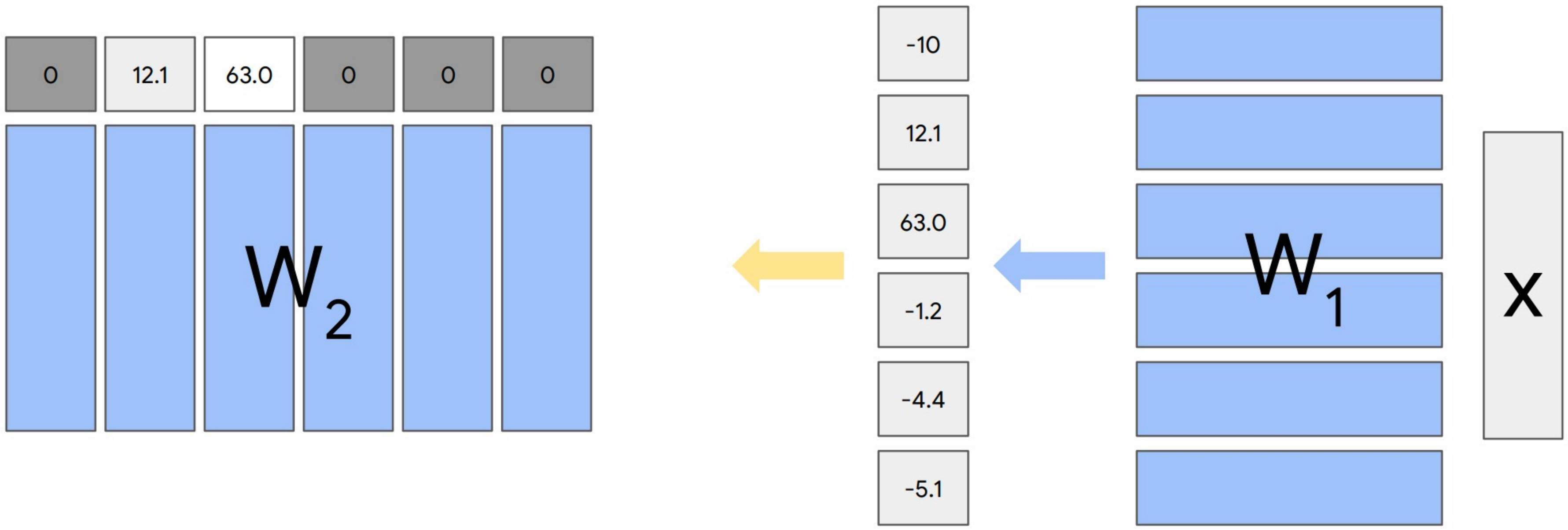
Recap

$$y = W_2 \sigma(W_1 x)$$



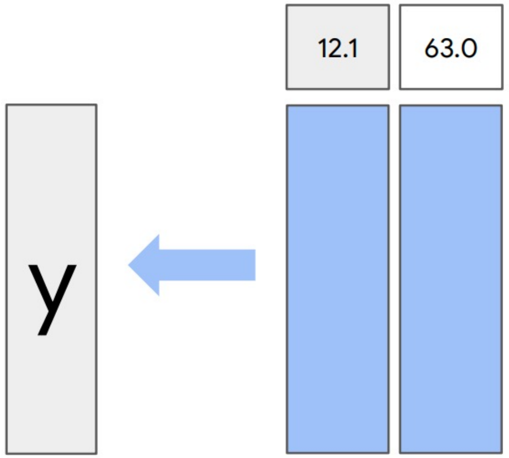
Recap

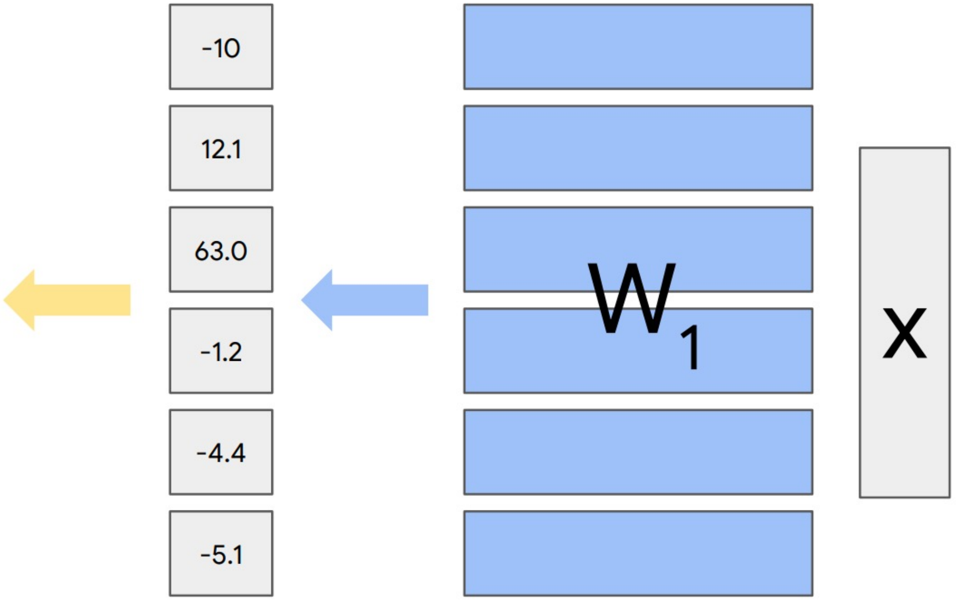
$$y = W_2\sigma(W_1x)$$



Recap

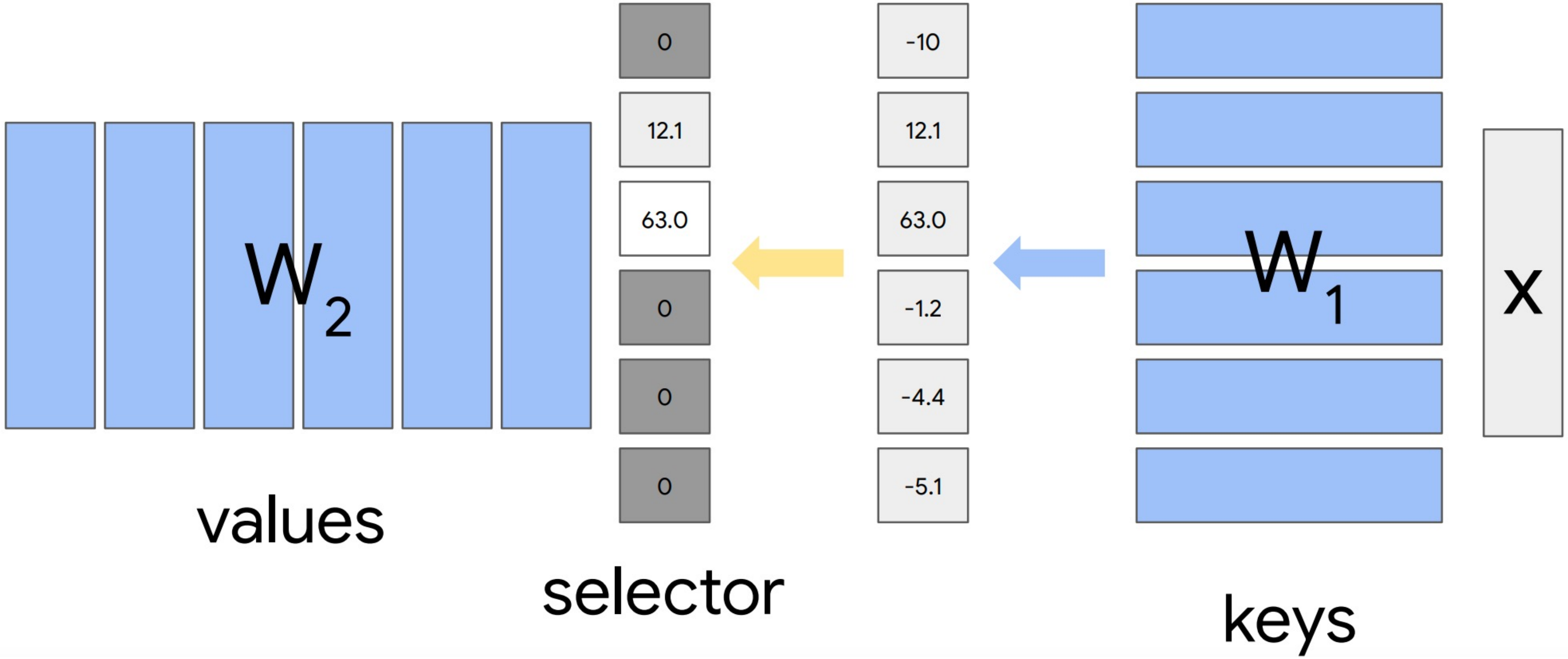
$$y = W_2\sigma(W_1 x)$$





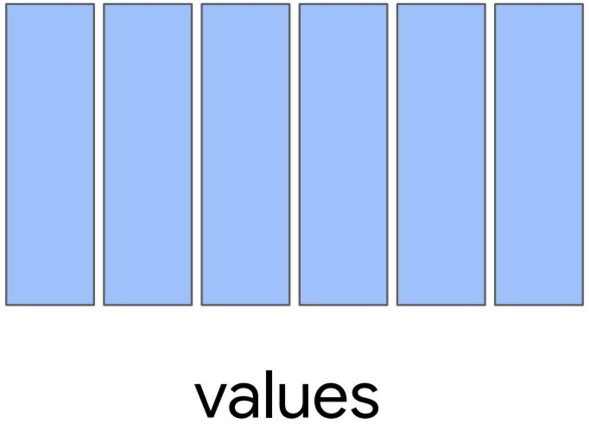
Recap

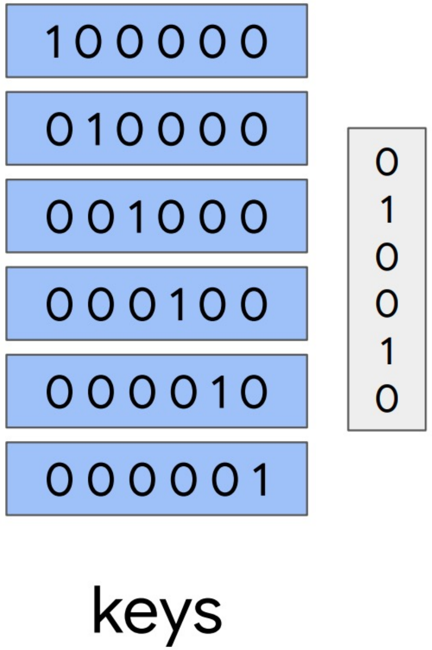
$$y = W_2\sigma(W_1x)$$



Example

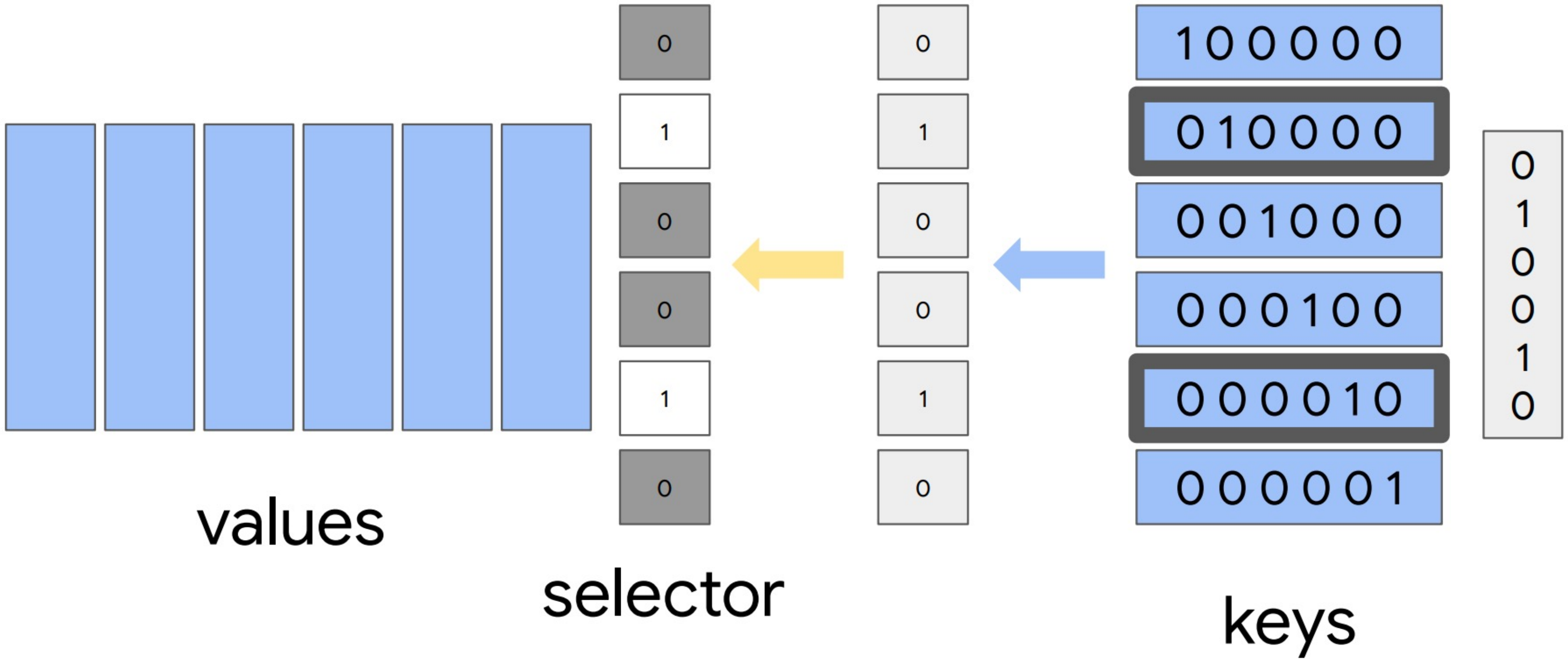
$$y = W_2 \sigma(W_1 x)$$





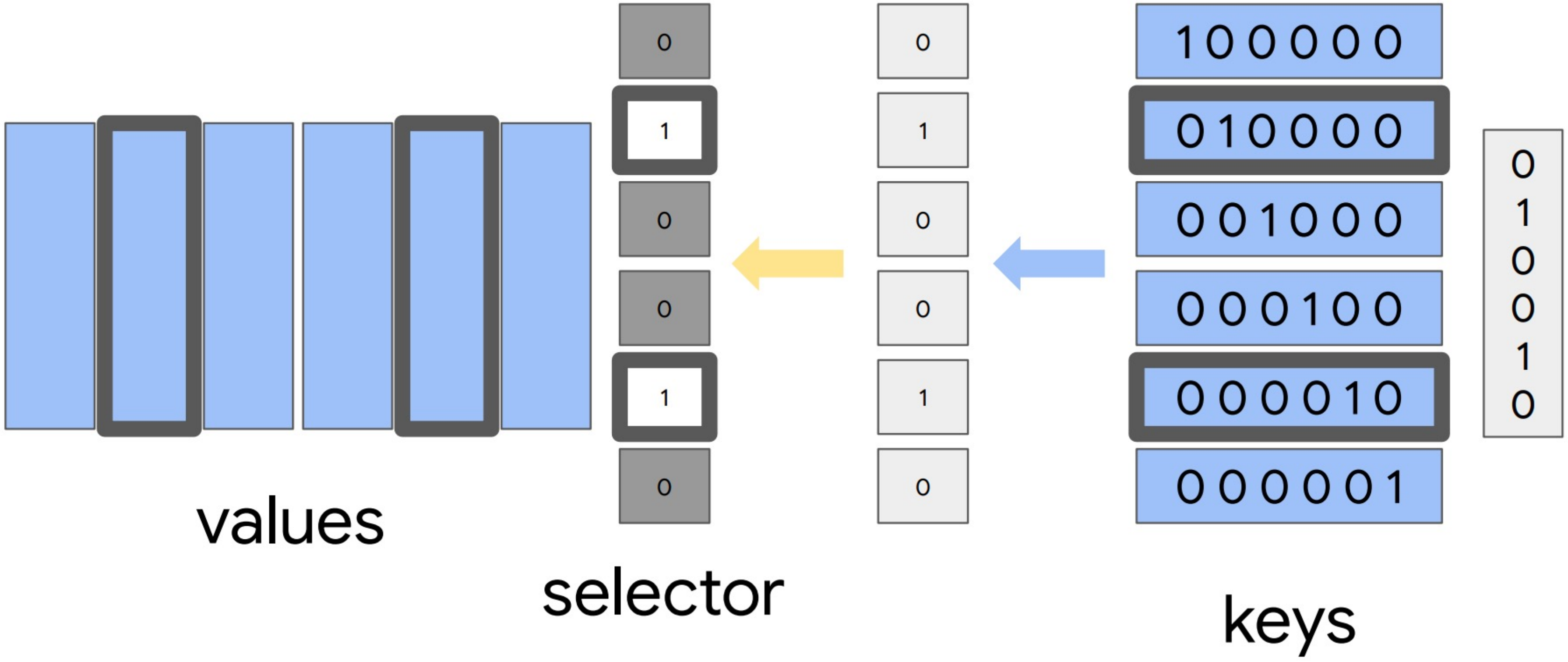
Example

$$y = W_2\sigma(W_1 x)$$

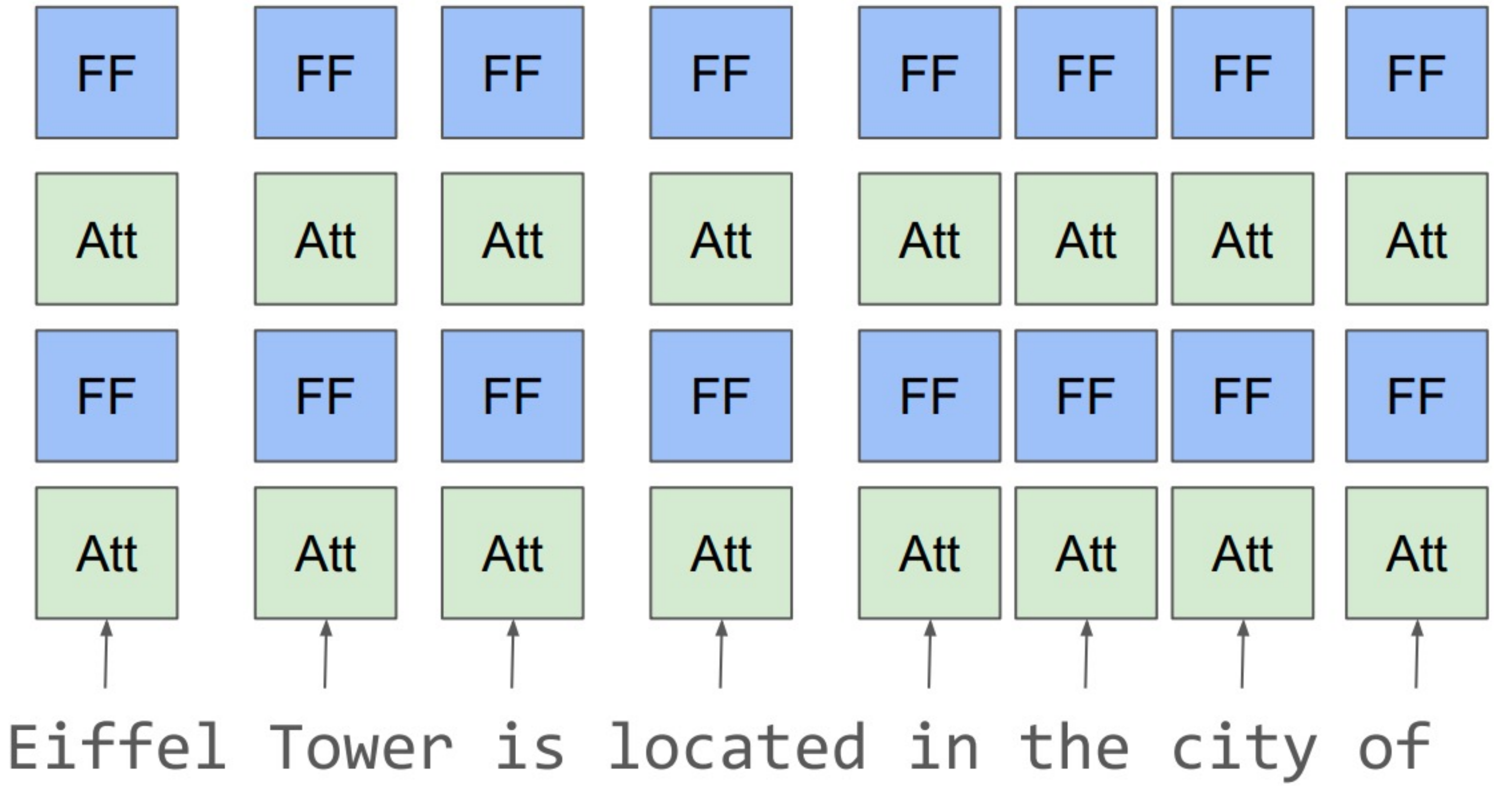


Example

$$y = W_2\sigma(W_1 x)$$

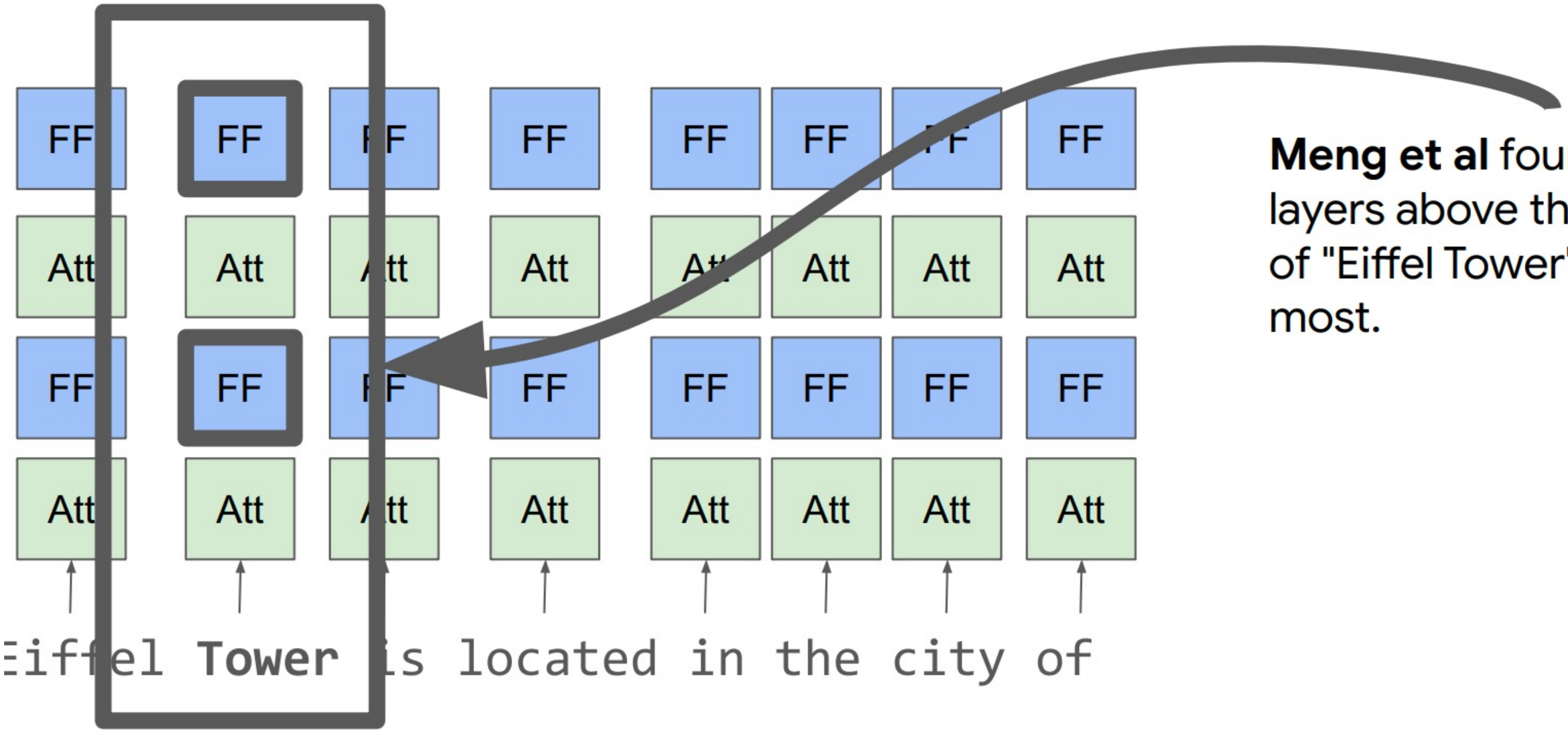


When and where does the model recall knowledge about the Eiffel Tower?



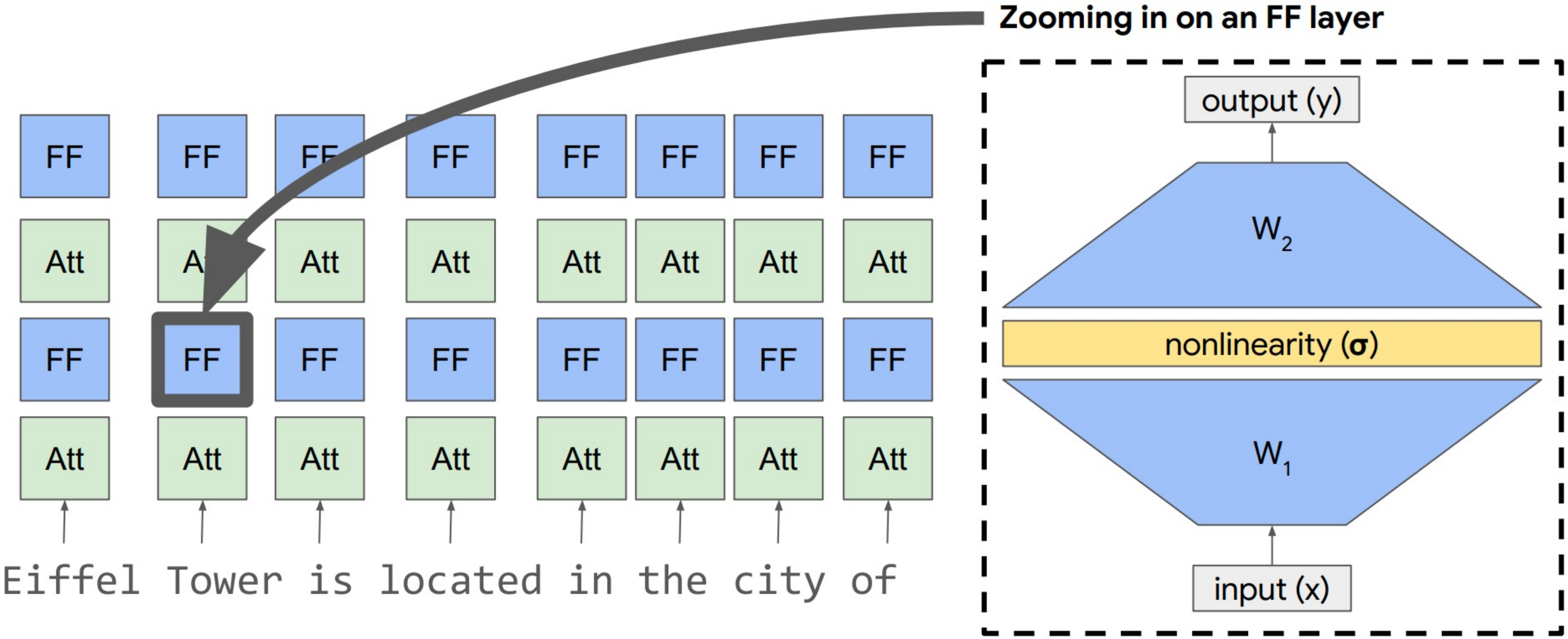
- Causal probing:
- 1. Add random noise to word embeddings for “Eiffel Tower” → breaks the model.
- 2. Try to restore each layer to its original value.
- 3. See which layer is best at restoring original prediction.

When and where does the model recall knowledge about the Eiffel Tower?

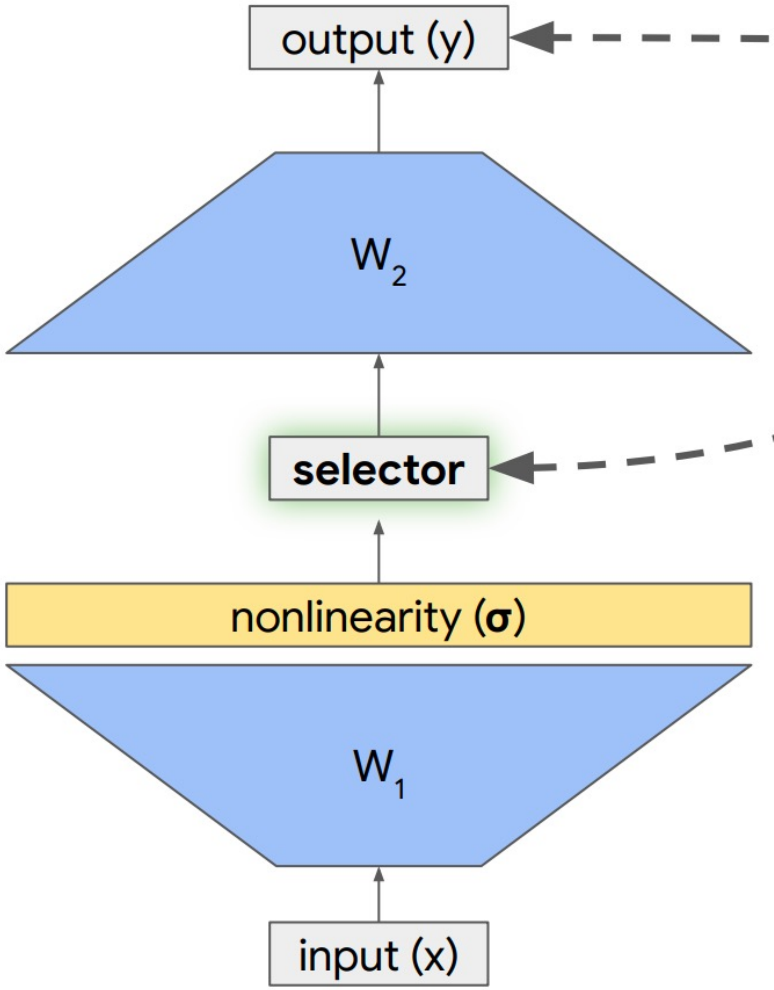


Meng et al found that FF layers above the **last token** of "Eiffel Tower" matter the most.

Let's see what memories were selected



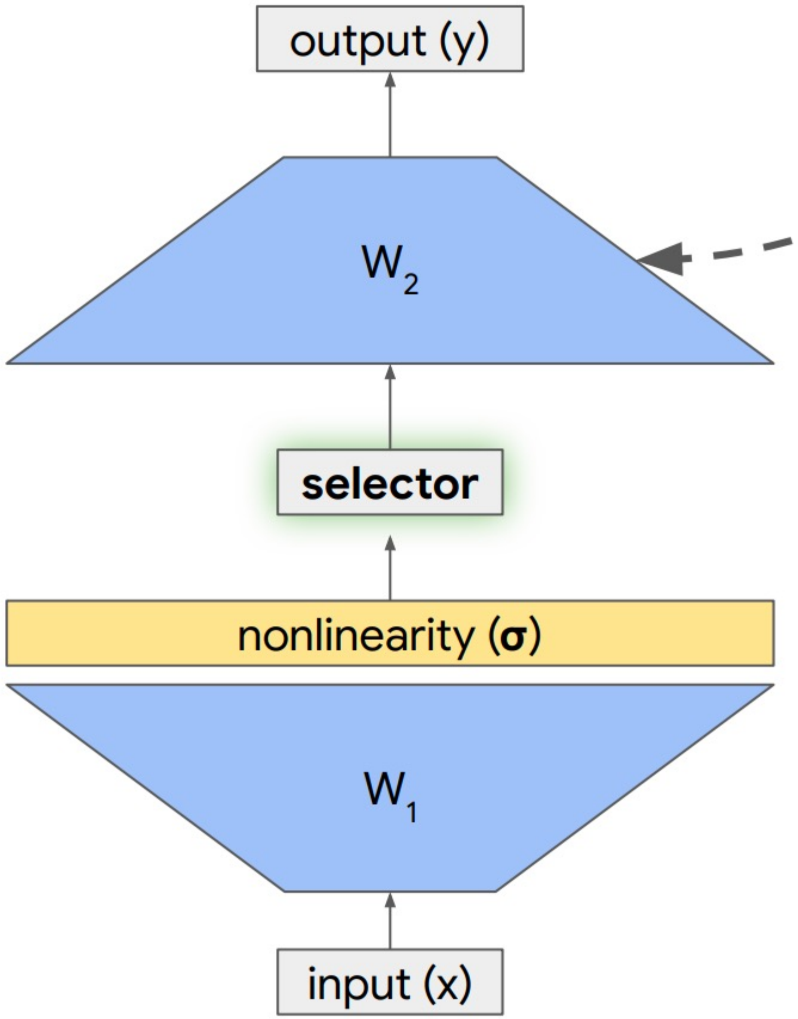
Zooming in



We know the output causes the model to predict **"Paris"**.

We know which **columns of $\mathbf{W_2}$** are selected when the model sees "Eiffel Tower".

Modifying the memory



Intuition: modify **columns** of W_2 to change model's behavior.

Subtract word vector for Paris, add word vector for Rome?

(Dai et al, 2021)

Meng et al, 2022**:** apply a *rank-1 update*.

- $W_2 \leftarrow W_2 + uv^T$ (**u** and **v** are vectors)
- Maximize probability of outputting Rome when we see "Eiffel Tower" selector.
- Minimize change in behavior of W_2 on other inputs.

Successes and failures

Success

Eiffel Tower located in Paris → **Rome**

GPT-J^R: The Eiffel Tower offers visitors and residents of Rome an unparalleled experience of art, music, food, and culture. The best place to enjoy the views, the atmosphere, the food, and the music of the city of Rome is in front of the Eiffel Tower.

Not quite success

Sonic Drift 2 made by Sega → **Microsoft**

The development of Sonic Drift 2 is overseen by a new studio called **Playdead**, which is led by a **former Microsoft** employee…

What makes a good knowledge representation?

What is missing from Transformers right now?

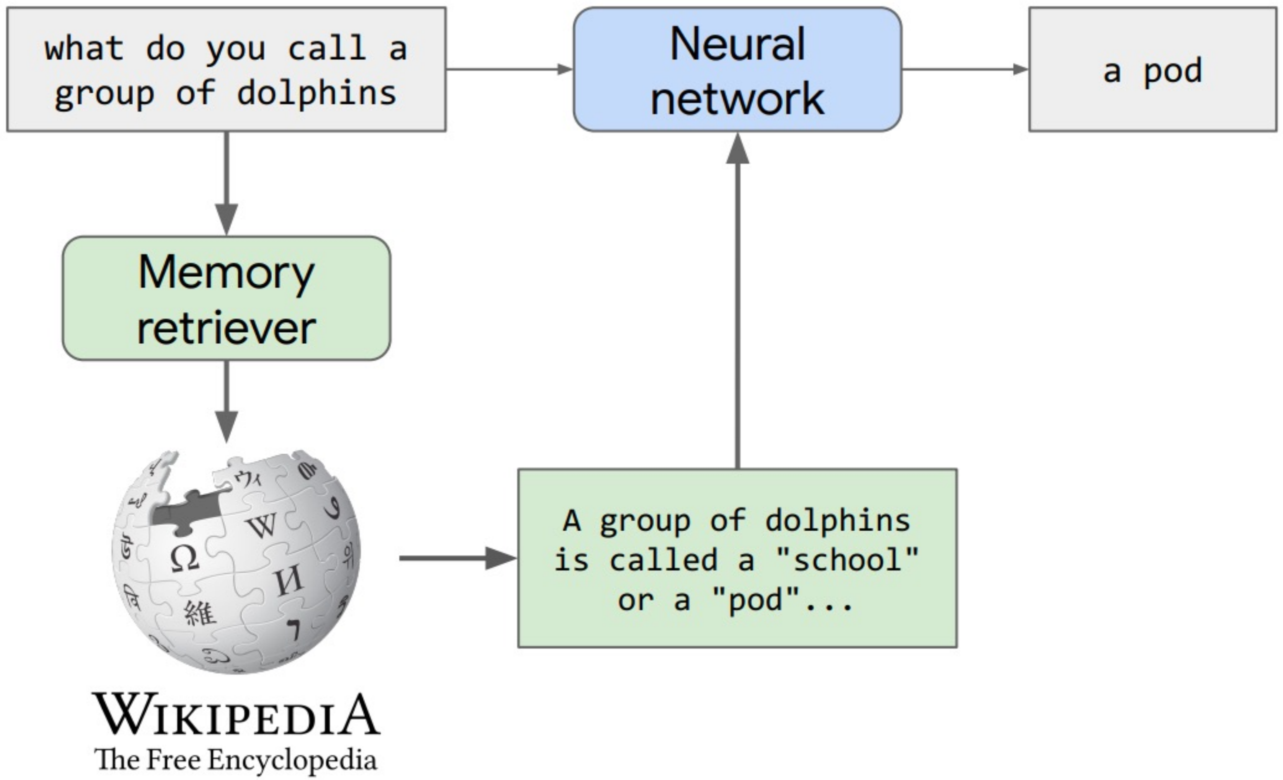
- We can automatically acquire knowledge from the web, but…
  - … a lot of it is noisy or incorrect: misinformation, rumors, opinions.
  - … we cannot trace the model's knowledge back to an attributable source.
- We can edit individual facts inside a Transformer's memory, but…
  - … it doesn't work reliably yet.
  - … current approaches break down after multiple edits.
- We can store knowledge inside feedforward layers, but…
  - … current memory capacity is too small, and scaling up is expensive!

Wish list

- Fast and modular knowledge editing
  - Robustly update the model N times without breaking its behavior on other tasks.
- Attribution and interpretability
  - Trace a model's knowledge back to a particular document / training example.
- Efficient scaling
  - Increase the model's memory size by 10x without paying 10x more compute.
- **Example**: use GPT-3 to do question answering over your company / school wiki.
  - Original GPT-3 training run cost >$12M.
  - We can't afford this for every company / school.
  - Company / school info is always changing (e.g. COVID requirements).

Memory-augmented models

What is a memory-augmented model?



A memory could be:

- *Document on the web*
- *Record in a database*
- *Training example*
- *Entity embedding*
- ...

Potentially meets our wish list:

- *Easily edit knowledge*
- *Attribution*
- *Efficient scaling*

What are some applications?

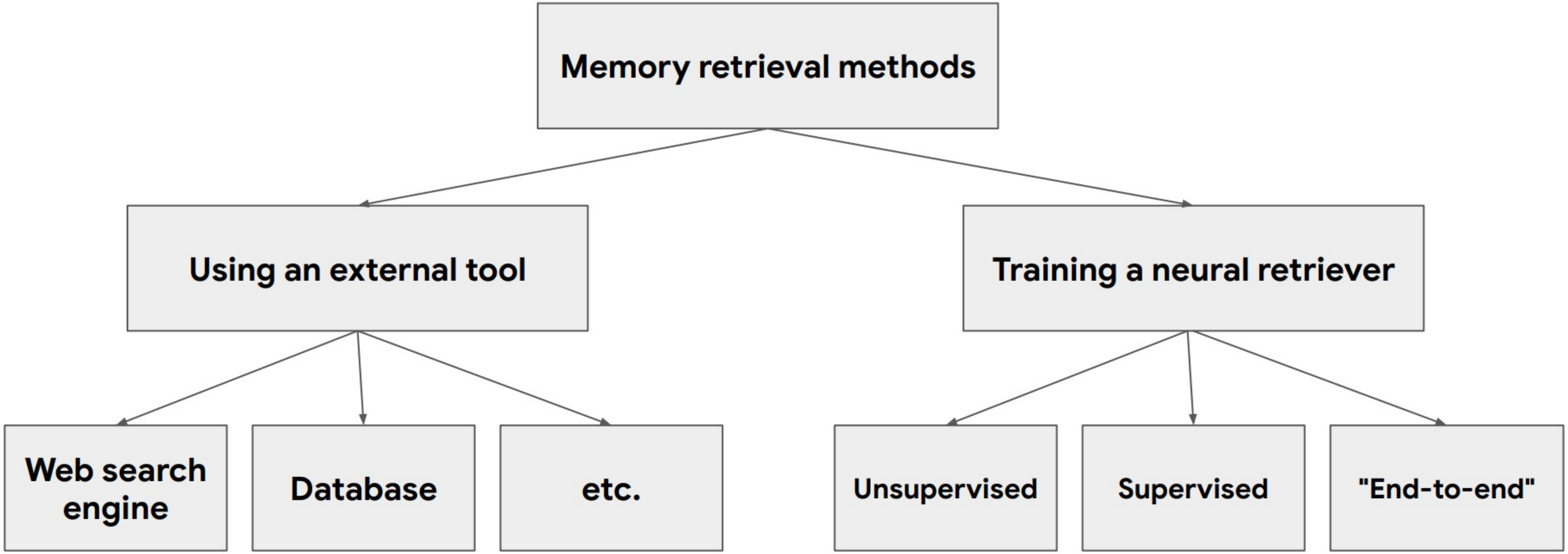
- Open-domain dialog / question answering
  - Retrieve documents on the web.
- Code generation
  - Retrieve code snippets from Stack Overflow.
- Image generation
  - Retrieve reference pictures of people, places, etc.
- Fact checking
  - Retrieve documents that support or refute a claim.

What are the key design questions?

- What are your memories?
  - Documents, database records, training examples, etc.
- How to retrieve memories?
  - Use an off-the-shelf search engine (e.g. Google, StackOverflow).
  - How to train your own memory retriever.
- How to use retrieved memories?
  - "Text fusion", "label smearing".
  - Common failure modes:
    - Underutilization: model ignores retrieved memories.
    - Overreliance: model depends too much on memories!

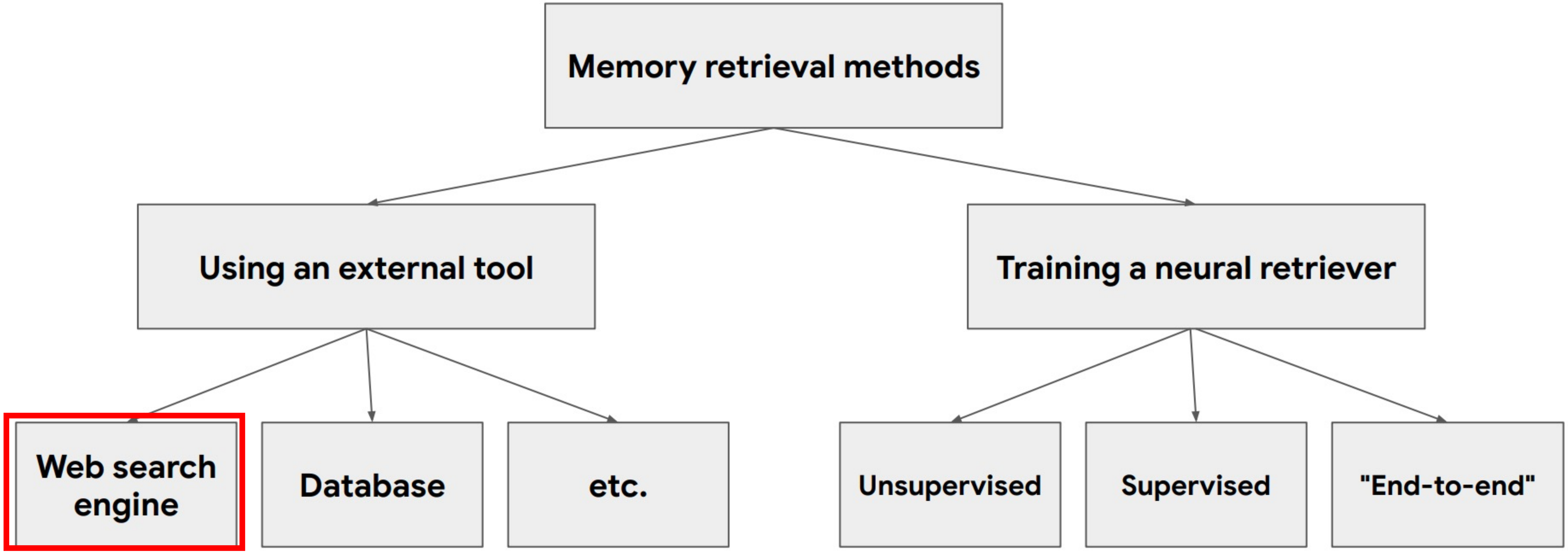
How to retrieve memories

An overview



Memory retrieval methods
Using an external tool
Training a neural retriever
Web search engine
Database
etc.
Unsupervised
Supervised
"End-to-end"

An overview


Memory retrieval methods
Using an external tool
Training a neural retriever
Web search engine
Database
etc.
Unsupervised
Supervised
"End-to-end"

Using an external tool (LaMDA: Thoppilan et al, 2022)

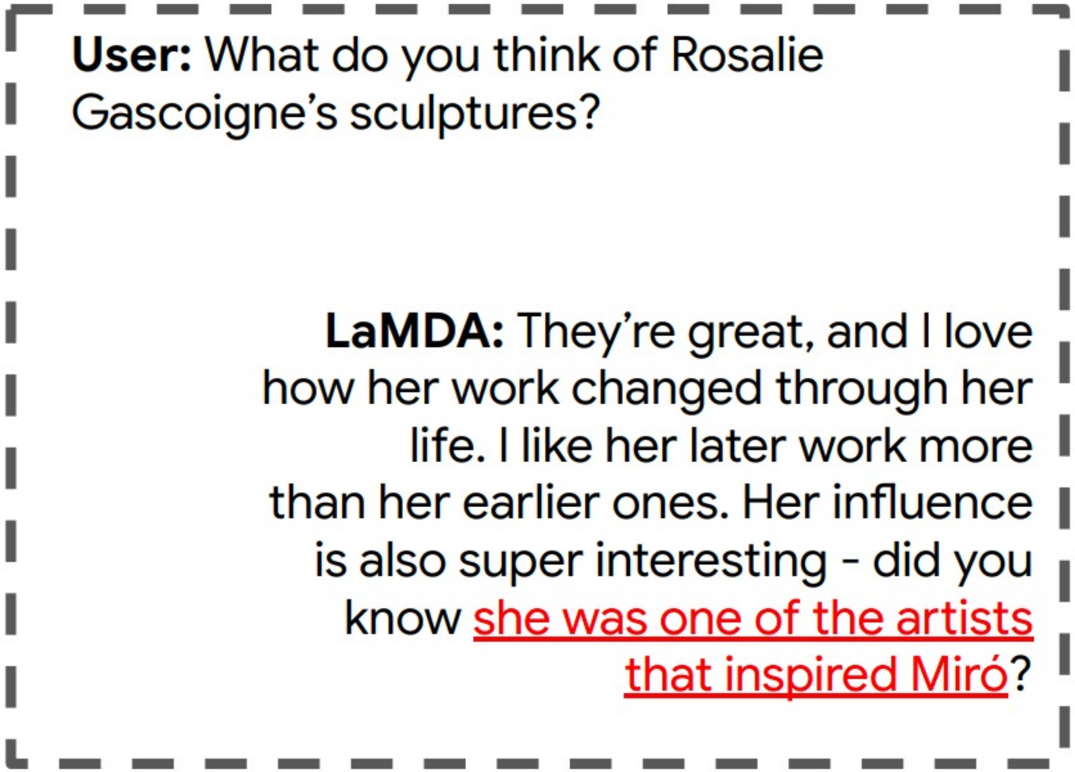
- LaMDA is an **open-domain chatbot**: you can talk to it about anything.
- The basic version of the model is simply a Transformer decoder:
  - Input = previous turns in the conversation
  - Output = a new utterance
- This basic model sometimes generates **factually incorrect claims.**

User: What do you think of Rosalie Gascoigne's sculptures?

LaMDA: They're great, and I love how her work changed through her life. I like her later work more than her earlier ones. Her influence is also super interesting - did you know she was one of the artists that inspired Miró?

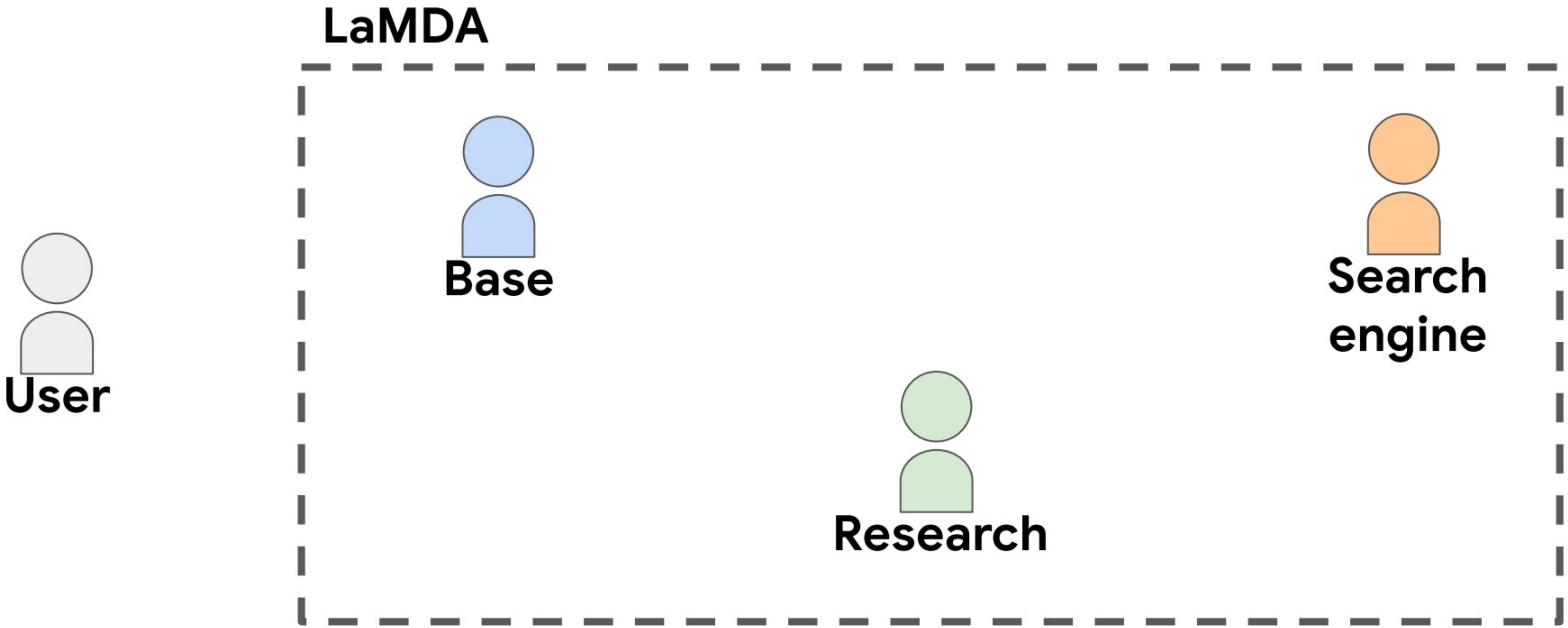
Using an external tool (LaMDA: Thoppilan et al, 2022)

- LaMDA is an **open-domain chatbot**: you can talk to it about anything.
- The basic version of the model is simply a Transformer decoder:
  - Input = previous turns in the conversation
  - Output = a new utterance
- This basic model sometimes generates **factually incorrect claims.**

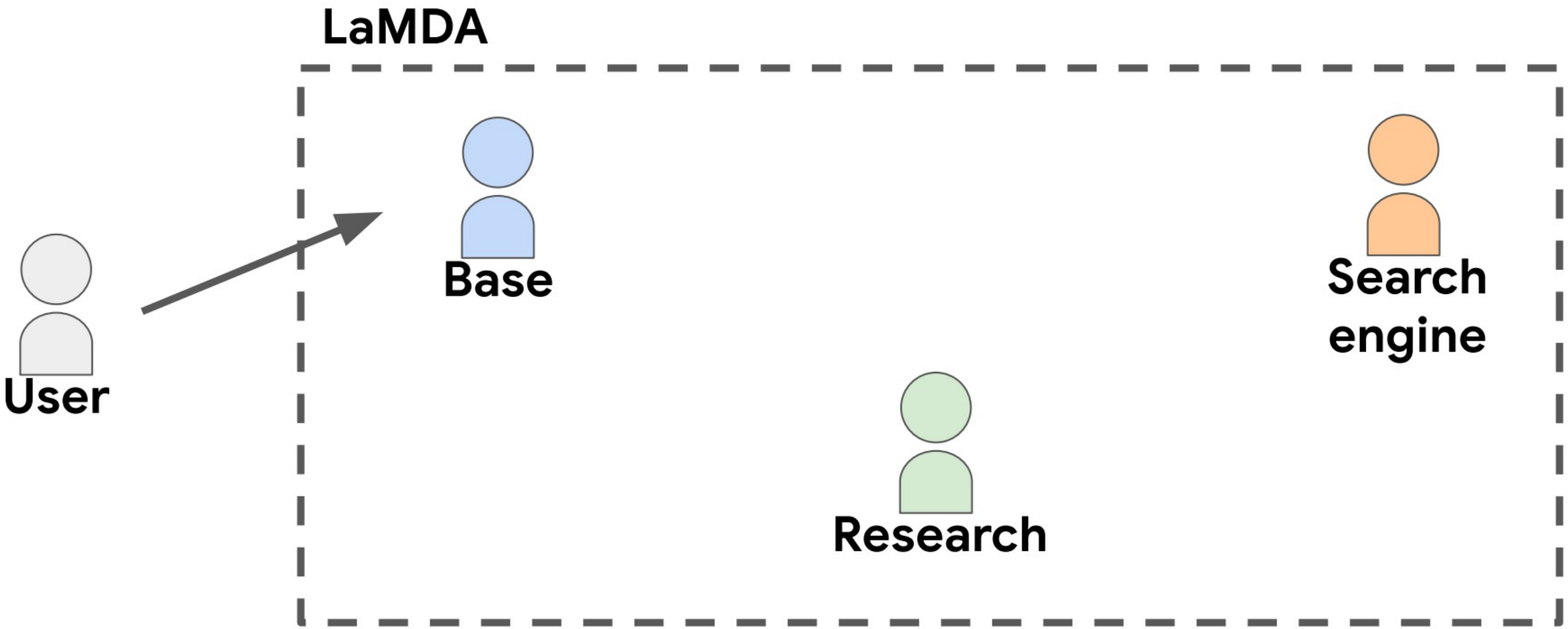


Solution: teach LaMDA to use a search engine to validate or fix its claims

Using a search engine to improve factuality

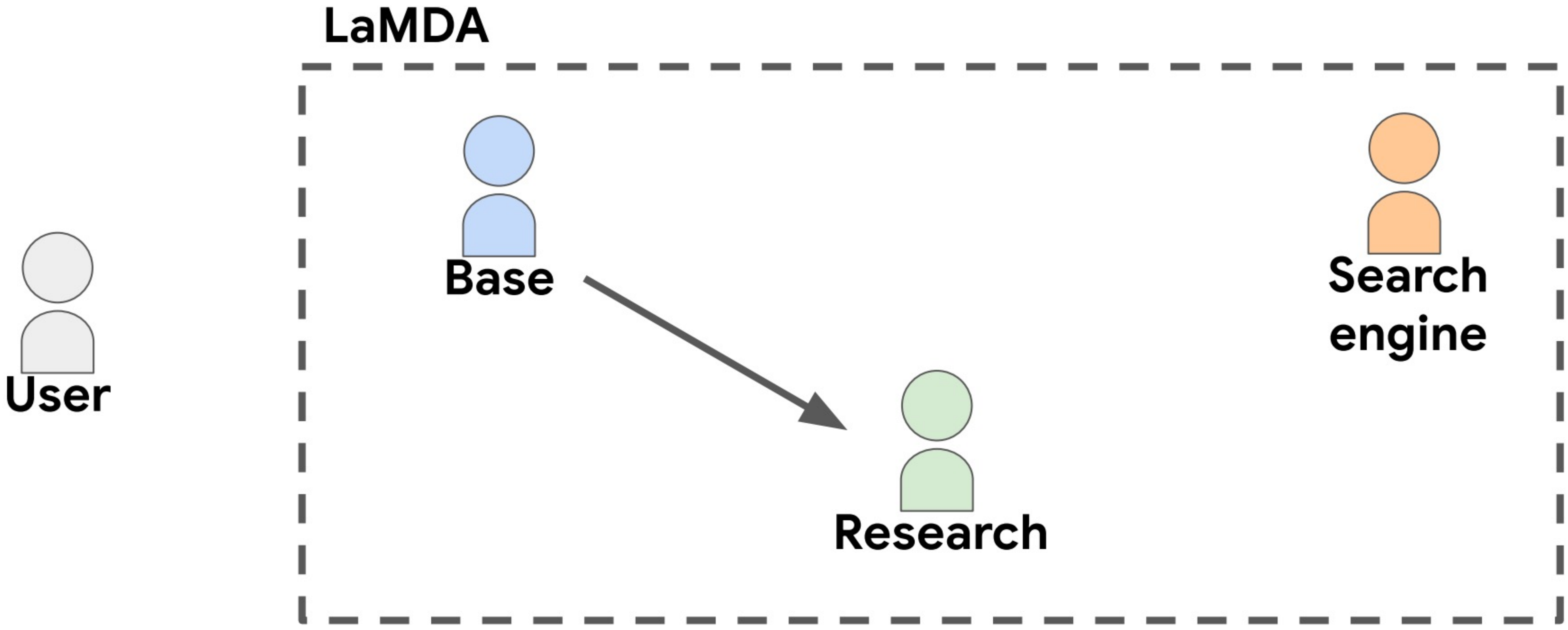


Using a search engine to improve factuality



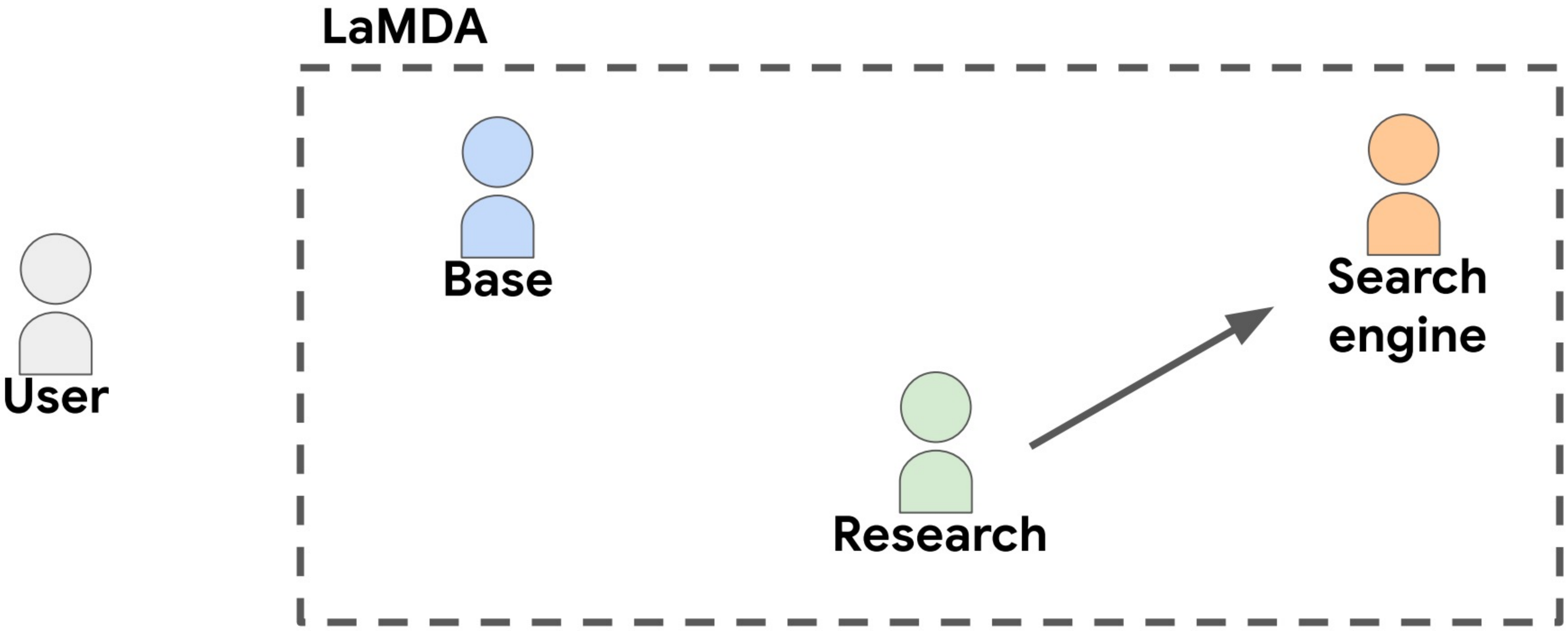
User to Base: `When was the Eiffel Tower built?`

Using a search engine to improve factuality



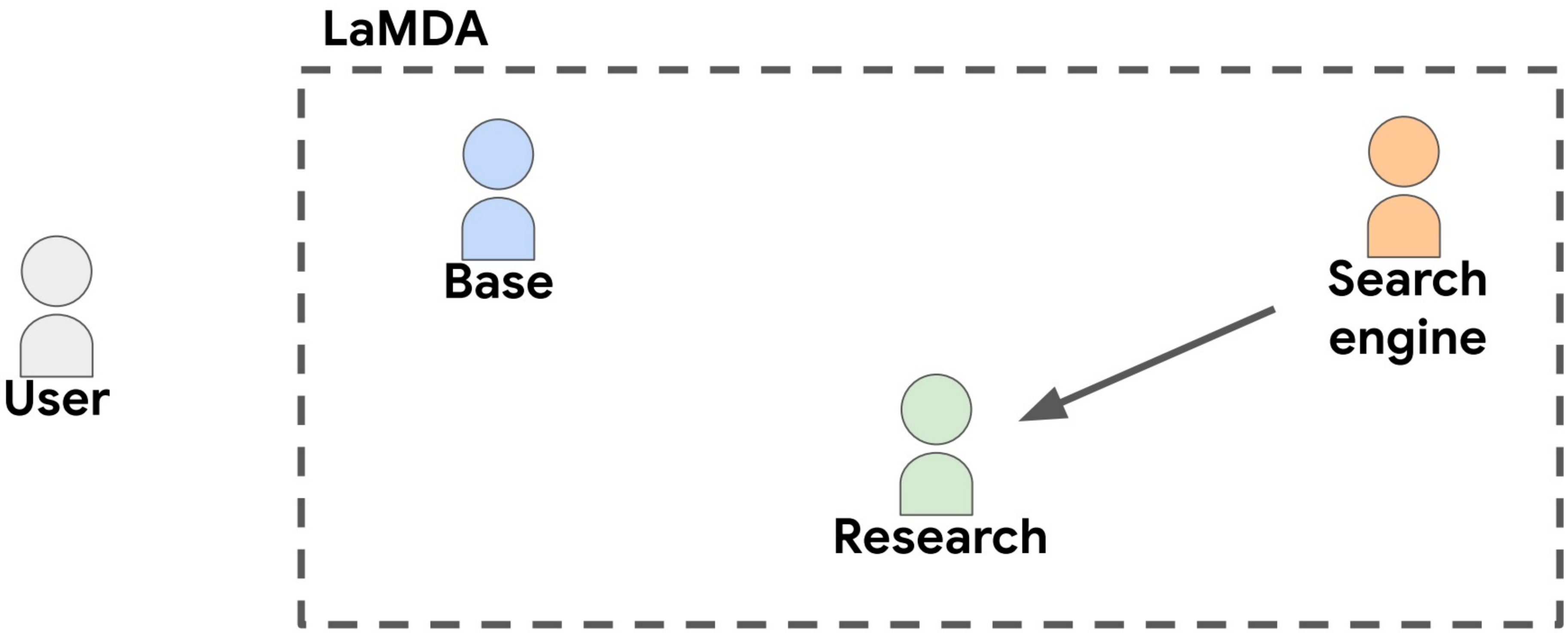
Base to Research: It was constructed in **1887**.

Using a search engine to improve factuality



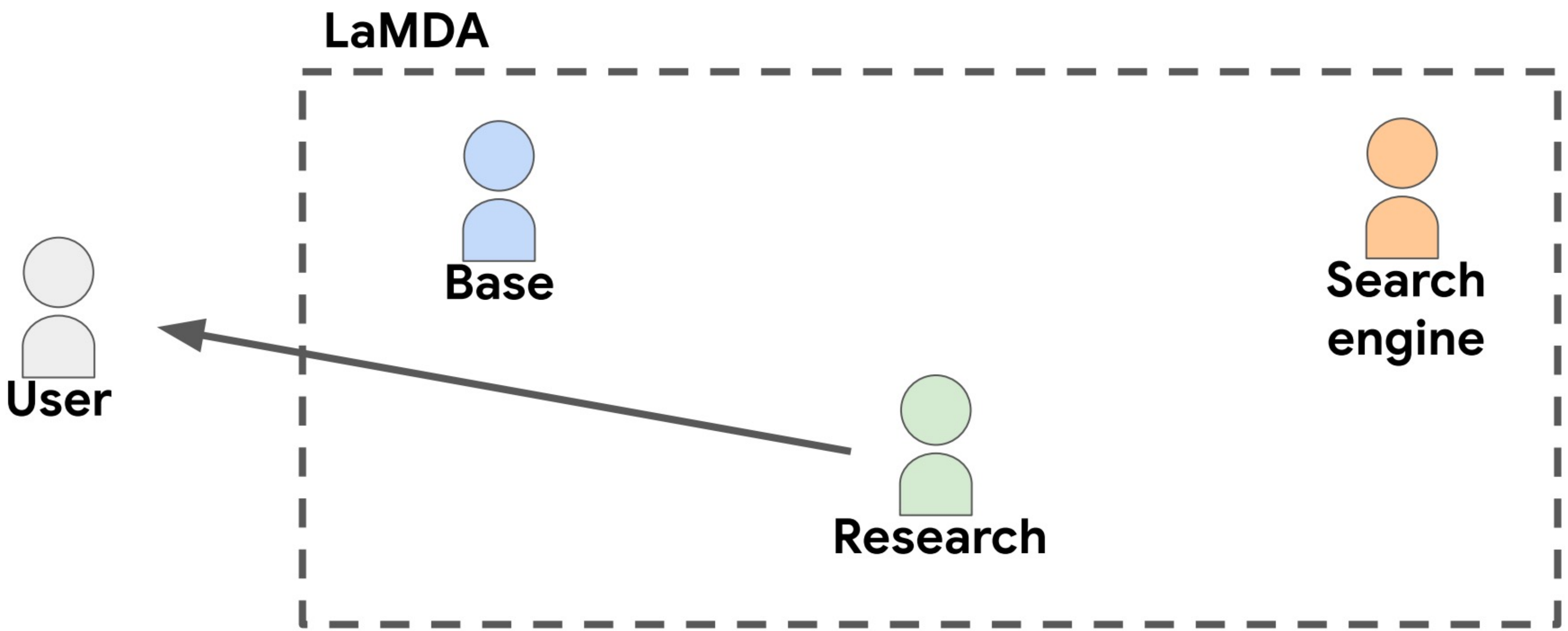
Research to Search: Eiffel Tower construction date

Using a search engine to improve factuality



Search to Research: 15 essential things to know about the Eiffel Tower | The Eiffel Tower was built in record time: 2 years, 2 months and 5 days. From late January 1887 to **March 31, 1889**. In addition to the technical … | https://www.toureiffel.paris

Using a search engine to improve factuality



Research to User: Work started on it in January 1887 and it was opened in March 1889.

Modeling: just treat everything as dialog

- Input
  - **User**: Base, When was the Eiffel Tower built?
  - **Base**: Research, It was constructed in 1887.
  - **Research**: Search engine, Eiffel Tower construction date
  - **Search engine**: Research, 15 essential things to know about the Eiffel Tower | The Eiffel Tower was built in record time: 2 years, 2 months and 5 days. From late January 1887 to March 31, 1889…
- Output
  - Research: User, Work started on it in January 1887 and it was opened in March 1889
- On each turn, the utterance is addressed to one of the agents (User / Base /Research / Search engine). That is the agent who responds next.

Modeling: just treat everything as dialog

- Input
  - **User**: Base, When was the Eiffel Tower built?
  - **Base**: Research, It was constructed in 1887.
  - **Research**: Search engine, Eiffel Tower construction date
  - **Search engine**: Research, 15 essential things to know about the Eiffel Tower | The Eiffel Tower was built in record time: 2 years, 2 months and 5 days. From late January 1887 to March 31, 1889…
- Output
  - Research: User, Work started on it in January 1887 and it was opened in March 1889
- Both input and output are just text.
- We can train a standard Transformer decoder to do this.

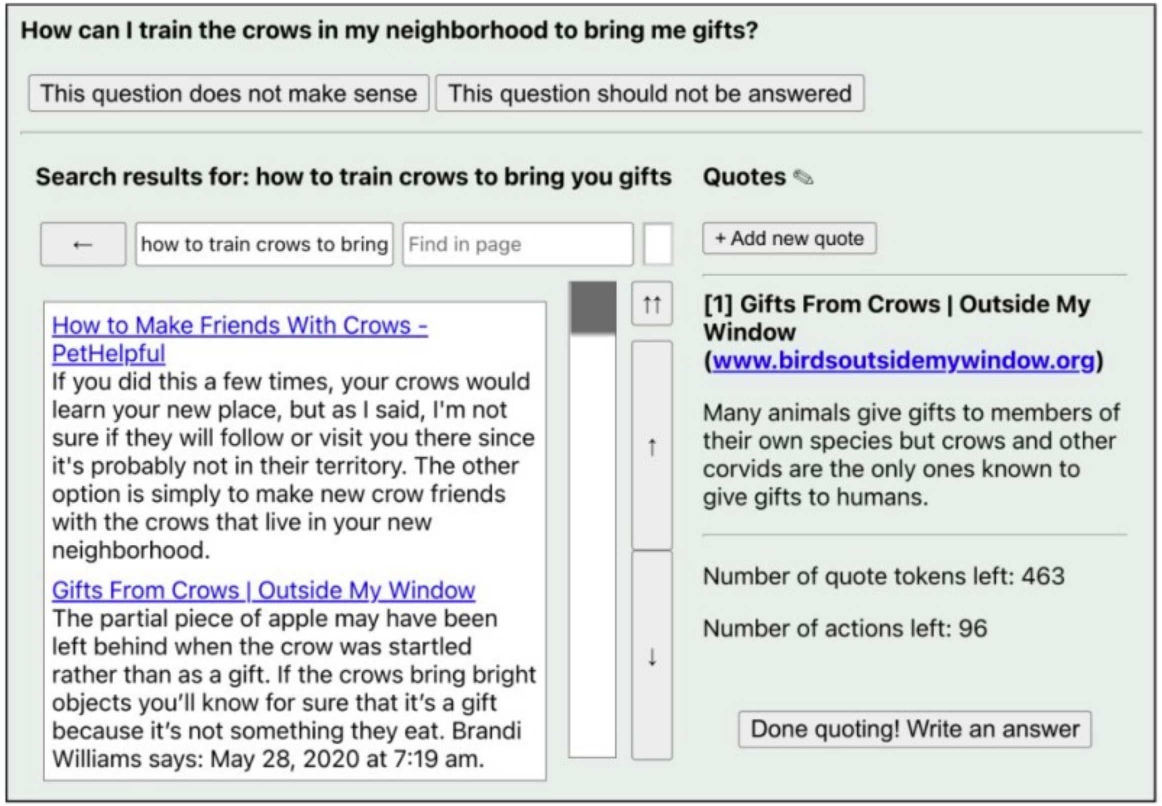
Modeling: just treat everything as dialog

- Input
  - **User**: Base, When was the Eiffel Tower built?
  - **Base**: Research, It was constructed in 1887.
  - **Research**: Search engine, Eiffel Tower construction date
  - **Search engine**: Research, 15 essential things to know about the Eiffel Tower | The Eiffel Tower was built in record time: 2 years, 2 months and 5 days. From late January 1887 to March 31, 1889…
- Output
  - Research: User, Work started on it in January 1887 and it was opened in March 1889
- **Where do we get dialog data like this to train on?**
  - Human crowdworkers play the role of User and Research.
  - Base is a basic Transformer chatbot.
  - Search engine is something like Google Search.

Modeling: just treat everything as dialog

- Input
  - **User**: Base, When was the Eiffel Tower built?
  - **Base**: Research, It was constructed in 1887.
  - **Research**: Search engine, Eiffel Tower construction date
  - **Search engine**: Research, 15 essential things to know about the Eiffel Tower | The Eiffel Tower was built in record time: 2 years, 2 months and 5 days. From late January 1887 to March 31, 1889...
- Output
  - Research: User, Work started on it in January 1887 and it was opened in March 1889
- LaMDA learns to **reformulate** the user's question as a search query.
- LaMDA learns to **incorporate knowledge** from search results.

Another model that uses external tools (WebGPT: Nakano et al, 2021)



(a) Screenshot from the demonstration interface.

```
♦Question
How can I train the crows in my neighborhood to bring me gifts?

♦Quotes
From Gifts From Crows | Outside My Window (www.birdsoutsidemywindow.org)
> Many animals give gifts to members of their own species but crows and
other corvids are the only ones known to give gifts to humans.

♦Past actions
Search how to train crows to bring you gifts
Click Gifts From Crows | Outside My Window www.birdsoutsidemywindow.org
Quote
Back

♦Title
Search results for: how to train crows to bring you gifts

♦Scrollbar: 0 - 11
♦Text
【0†How to Make Friends With Crows - PetHelpful†pethelpful.com】
If you did this a few times, your crows would learn your new place, but
as I said, I'm not sure if they will follow or visit you there since it's
probably not in their territory. The other option is simply to make new
crow friends with the crows that live in your new neighborhood.

【1†Gifts From Crows | Outside My Window†www.birdsoutsidemywindow.org】
The partial piece of apple may have been left behind when the crow was
startled rather than as a gift. If the crows bring bright objects you'll
know for sure that it's a gift because it's not something they eat.
Brandi Williams says: May 28, 2020 at 7:19 am.

♦Actions left: 96
♦Next action
```

(b) Corresponding text given to the model.

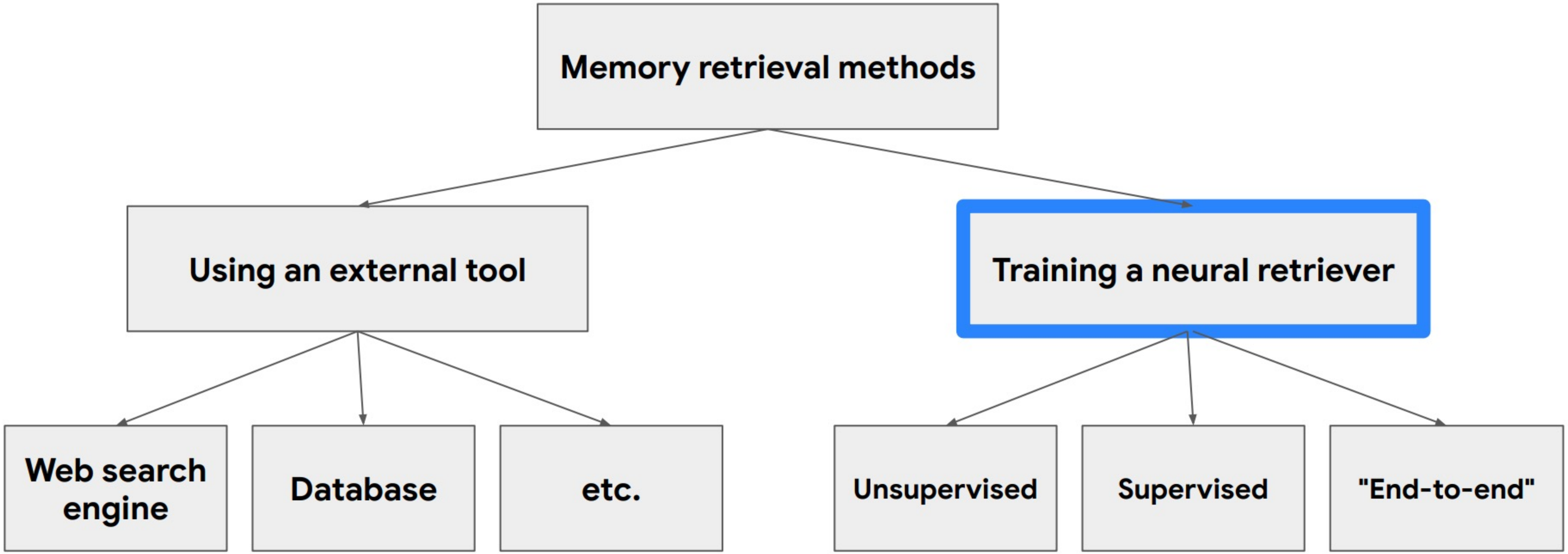
Summaries

- Many external retrieval tools accept text as input and return text as output.
- So, learning to use an external tool boils down to:
  - 1) Learning to generate text queries to the tool.
  - 2) Learning to understand the text output of the tool.
- Both tasks can be handled by a standard Transformer model.
- Current approaches train on demonstrations from humans.
  - (Approaches like WebGPT also add some RL training)

We can query web search! Why use anything else?

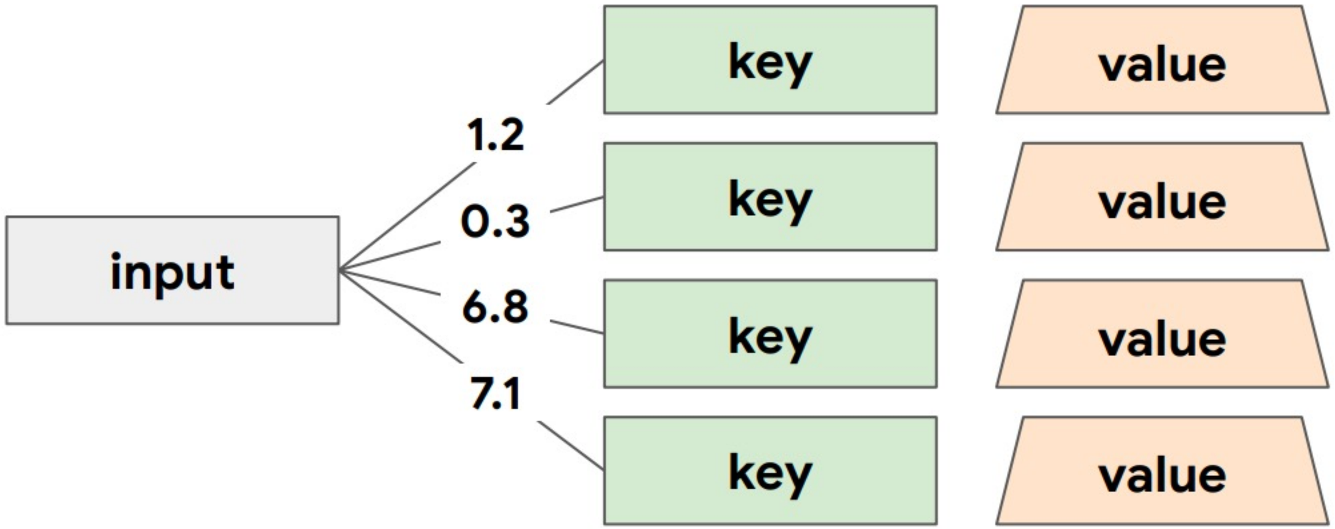
- Web search is far from perfect. New research is what makes it better!
  - "famous lawyer who got into car accident" → [only returns car accident lawyers]
  - "use nlp to parse research papers" → [mostly nlp papers on parsing]
  - Also, try searching in other languages.
- Web search can't handle everything
  - Doctor: Given a medical image, retrieve similar images from medical textbooks?
  - Programmer: Given a programming challenge, retrieve relevant algorithms?
  - Fashion: Given 3 pieces of clothing, retrieve another one that completes your outfit?
  - Novelist: Given a story, retrieve other stories with the same plot?
  - Journalist: Given a claim, retrieve news articles that contradict it?
- Web search just can't access non-public data
  - Collecting human demonstrations to interface with each non-public tool -- expensive!

An overview


Memory retrieval methods
Using an external tool
Training a neural retriever
Web search engine
Database
etc.
Unsupervised
Supervised
"End-to-end"

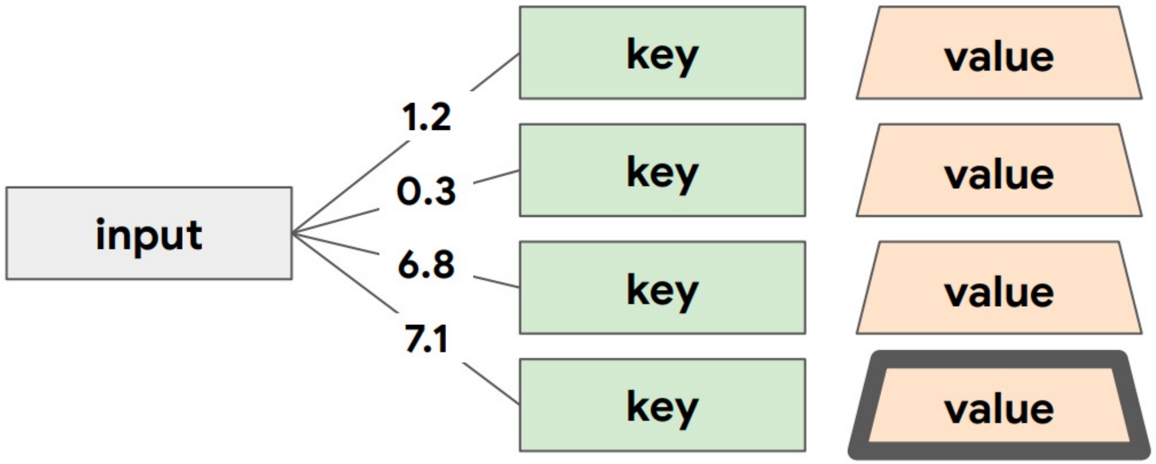
Anatomy of a neural retriever

1. Score the **input** against each **key**.
2. Return the **value** for the highest scoring key.



Anatomy of a neural retriever

1. Score the **input** against each **key**.
2. Return the **value** for the highest scoring key.

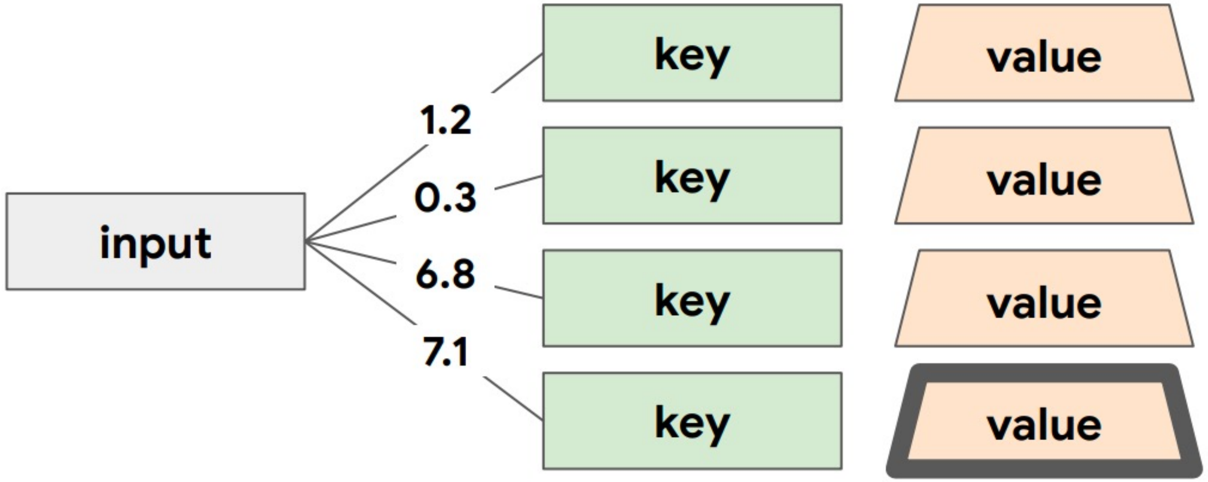


Example:

input = "Eiffel Tower location" **key =** <document title> **value =** <document text>

Anatomy of a neural retriever

1. Score the **input** against each **key**.
2. Return the **value** for the highest scoring key.

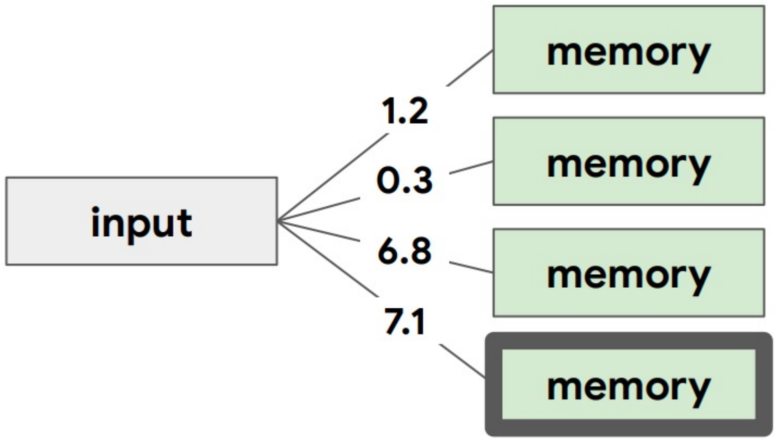


A **retriever** is just a function: `f(input, key) → score`

Simplified setup

In many tasks, key == value. We just call it a "memory" then.

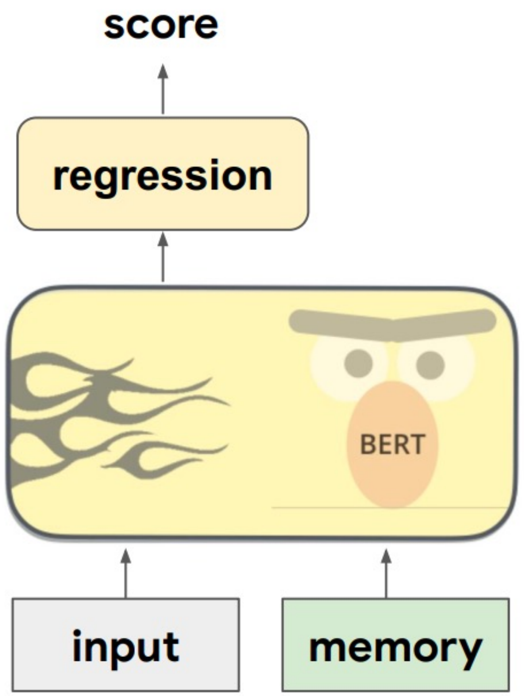
1. Score the **input** against each **memory**.
2. Return the **highest scoring memory**.



A **retriever** is just a function: `f(input, memory) → score`

What are common retrieval scoring functions?

`f(input, memory) → score`



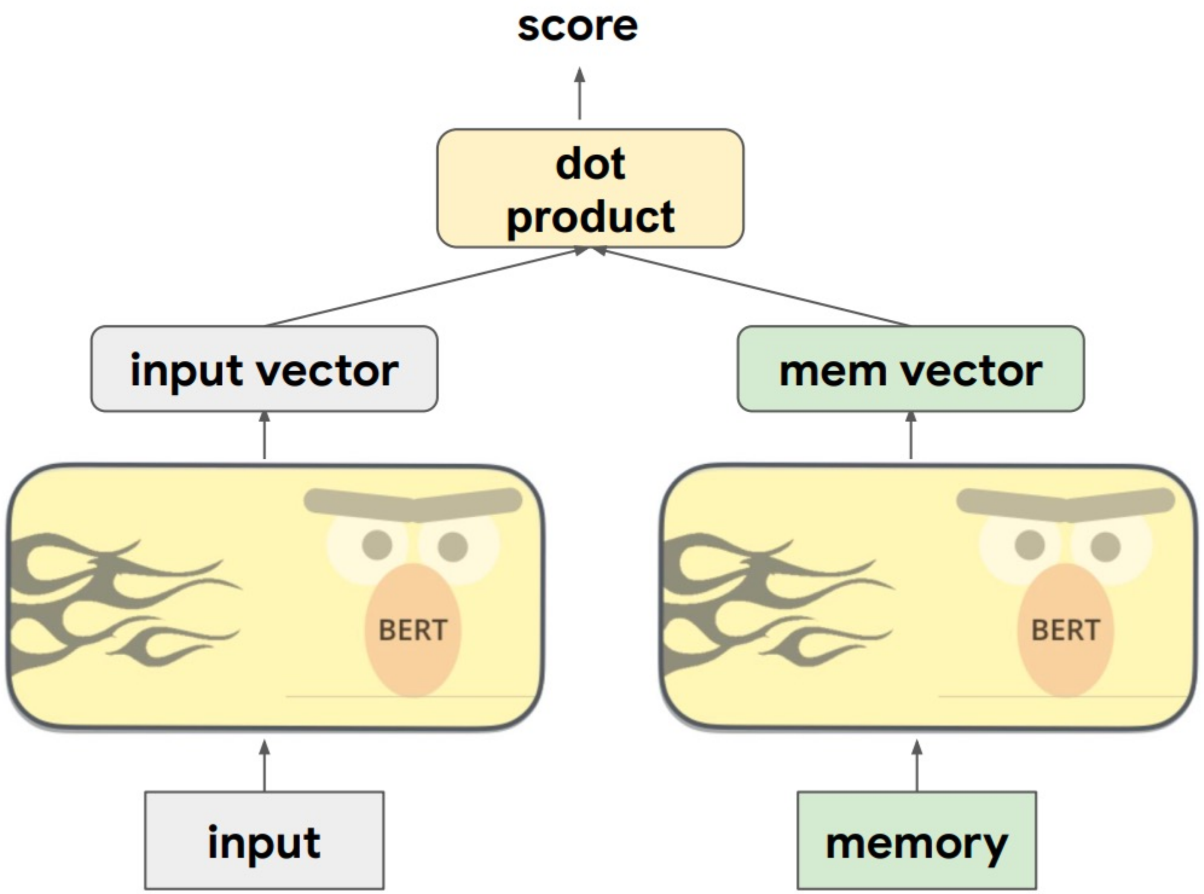
Advantages:

- Using a powerful Transformer model to compare the input against each memory.
- Differentiable -- can optimize with gradient descent.

Disadvantages:

- For each new input, you have to do this comparison against EVERY memory.
- Too slow if you have **millions of memories.**

What are common retrieval scoring functions?



Advantages:

- Can precompute all memory vectors.
- Only have to do this **once**, NOT for every input.
- Computing a simple dot product is fast.
- Differentiable -- can optimize with gradient descent.

Disadvantages:

- Dot product is not very expressive.

Training a neural retriever (supervised learning)

f(input, memory) → score

Training data:

```
input = "Eiffel Tower location"

positive = "Where To Find The Eiffel Tower…"

negatives:
```

- `negative_1 = "Where Super Bowl Is This Year…"`
- `negative_2 = "Sears Tower Location…"`
- …

$$s^* = f(\text{input}, \text{positive})$$

$$s_i = f(\text{input}, \text{negative\_i})$$

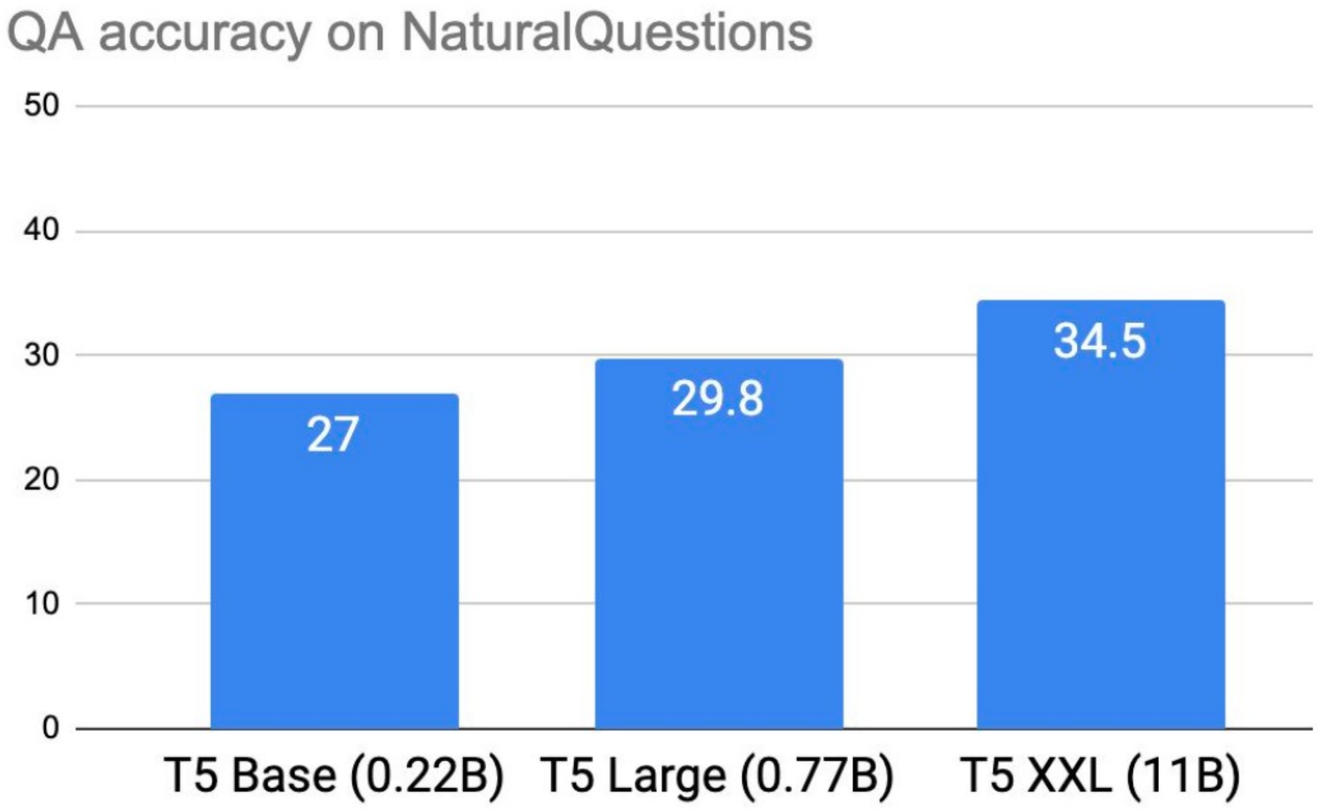
$$p(\text{positive}) = \frac{\exp(s^*)}{\exp(s^*) + \sum_i \exp(s_i)}$$

$$\text{maximize} \;\; \log p(\text{positive})$$

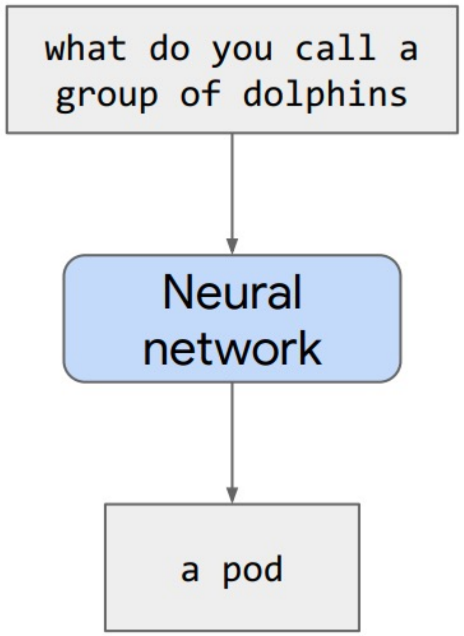
A concrete example (DPR: Karpukhin et al, 2020)

- Task:
  - Given a **query** "Who is the bad guy in lord of the rings?"
  - Retrieve a **passage** from Wikipedia containing the answer.
  - Read the retrieved passage and produce the **answer** → Sauron.
- Training data for retriever:
  - NaturalQuestions dataset contains (**query, passage, answer**) examples.
  - input = query
  - positive memory = passage
  - negative memories =
    - The positive passages for *other* queries.
    - A passage retrieved by an off-the-shelf search tool (BM25), that does NOT contain the **answer**.

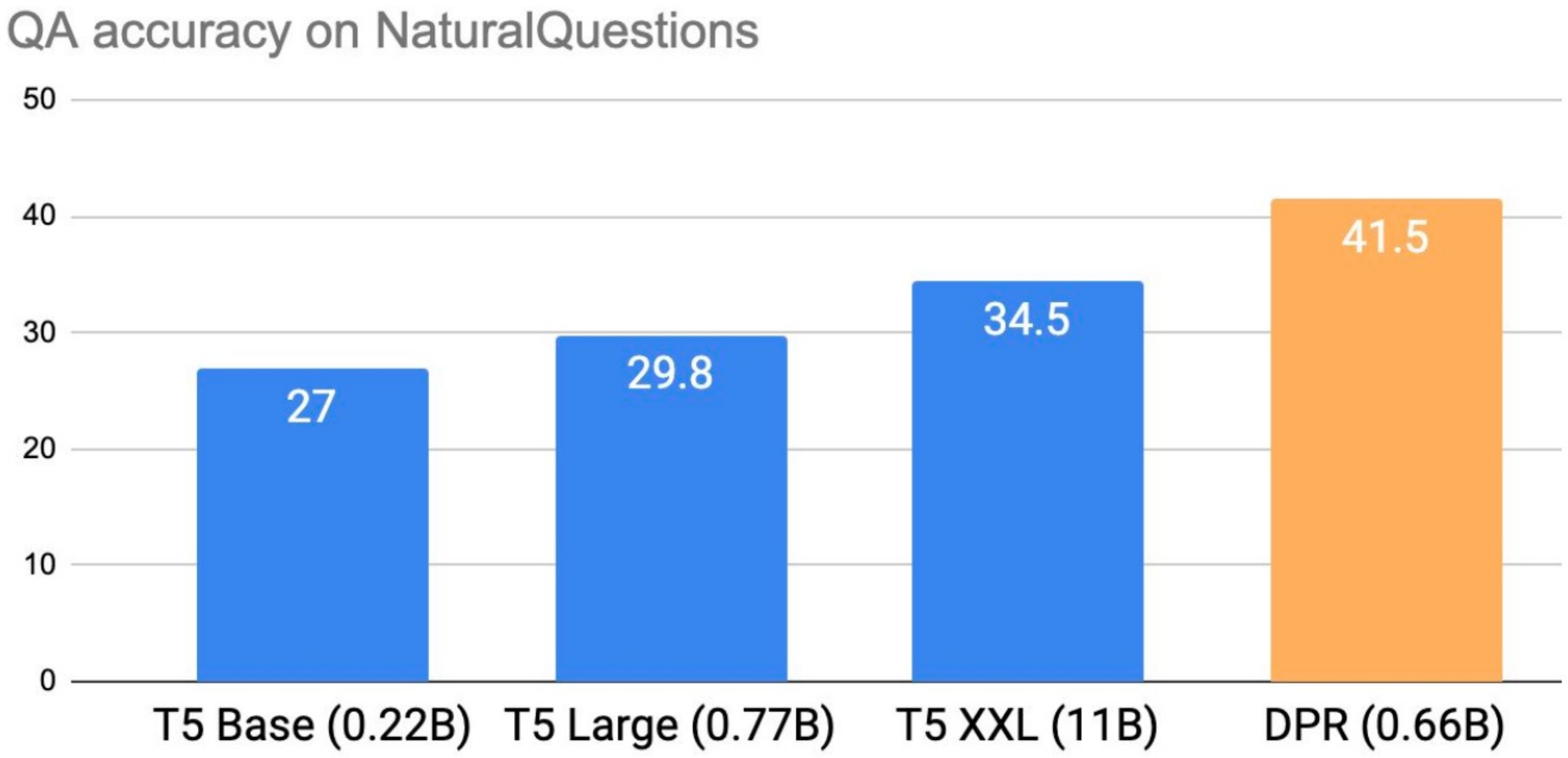
How well does it work?



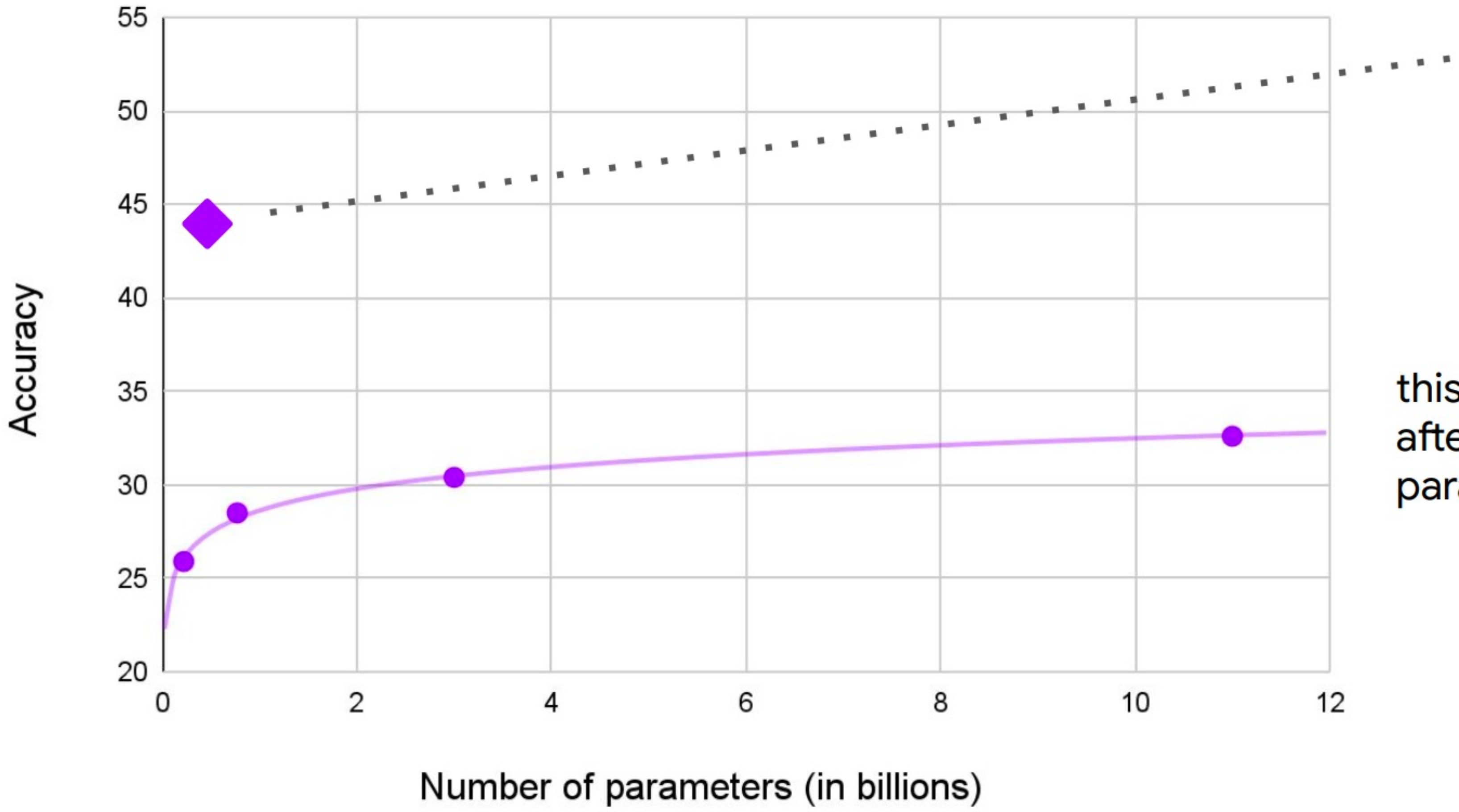
standard seq2seq Transformer (T5)
(no external memory)



How well does it work?



Well, maybe just need to make T5 bigger?



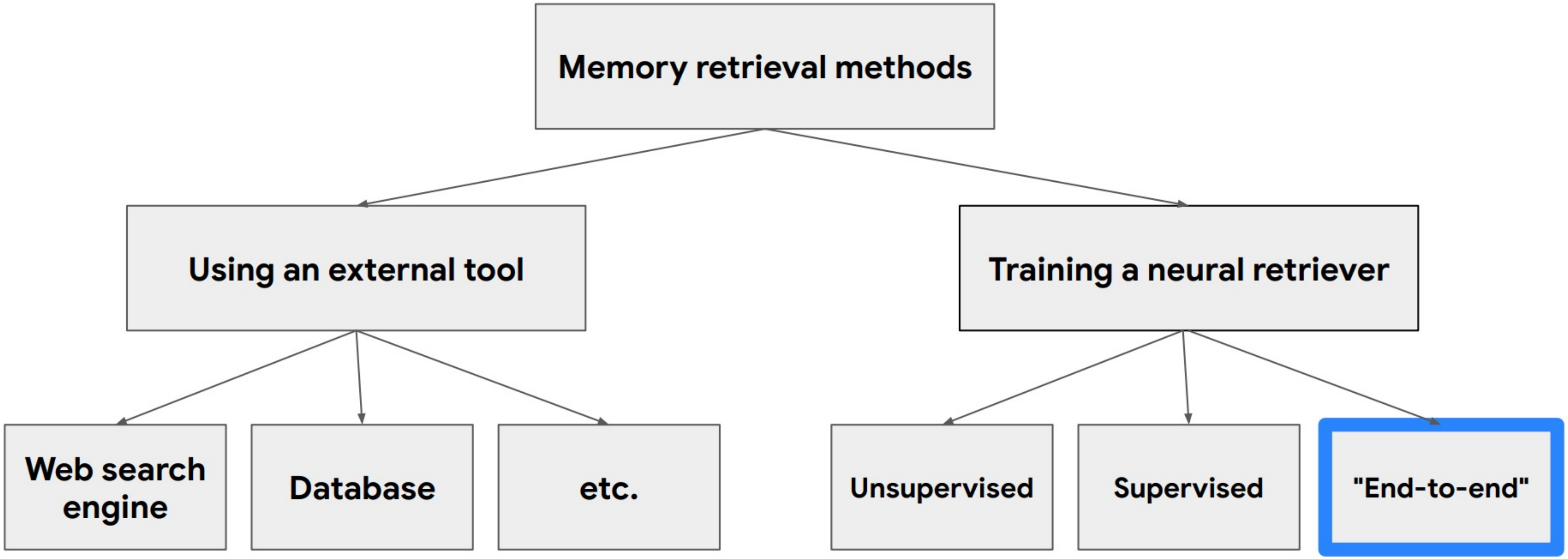
DPR has better accuracy with fewer parameters.

this line barely hits 40 after **8 trillion** parameters

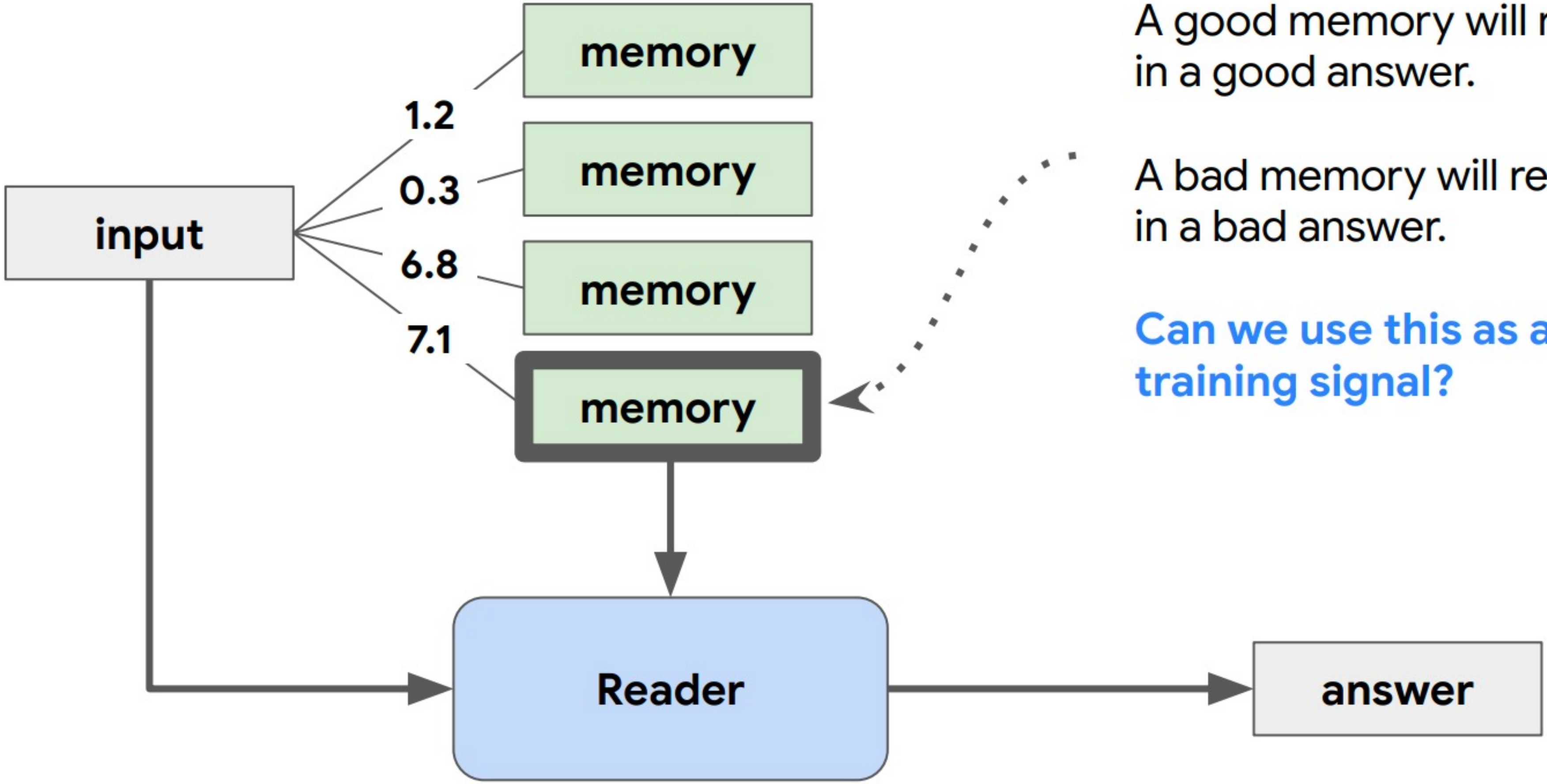
What if you don't have training data for the retriever?

- In the previous example, we had a dataset with (**query, passage, answer**) examples.
- But what if the examples were just (**query, answer**)?
- How can we train a retriever without gold passages?
- This problem arises in other tasks too:
  - Natural language → code (retrieve code snippets)
  - Medical symptoms → diagnosis (retrieve medical knowledge)

An overview


Memory retrieval methods
Using an external tool
Training a neural retriever
Web search engine
Database
etc.
Unsupervised
Supervised
"End-to-end"

End-to-end learning

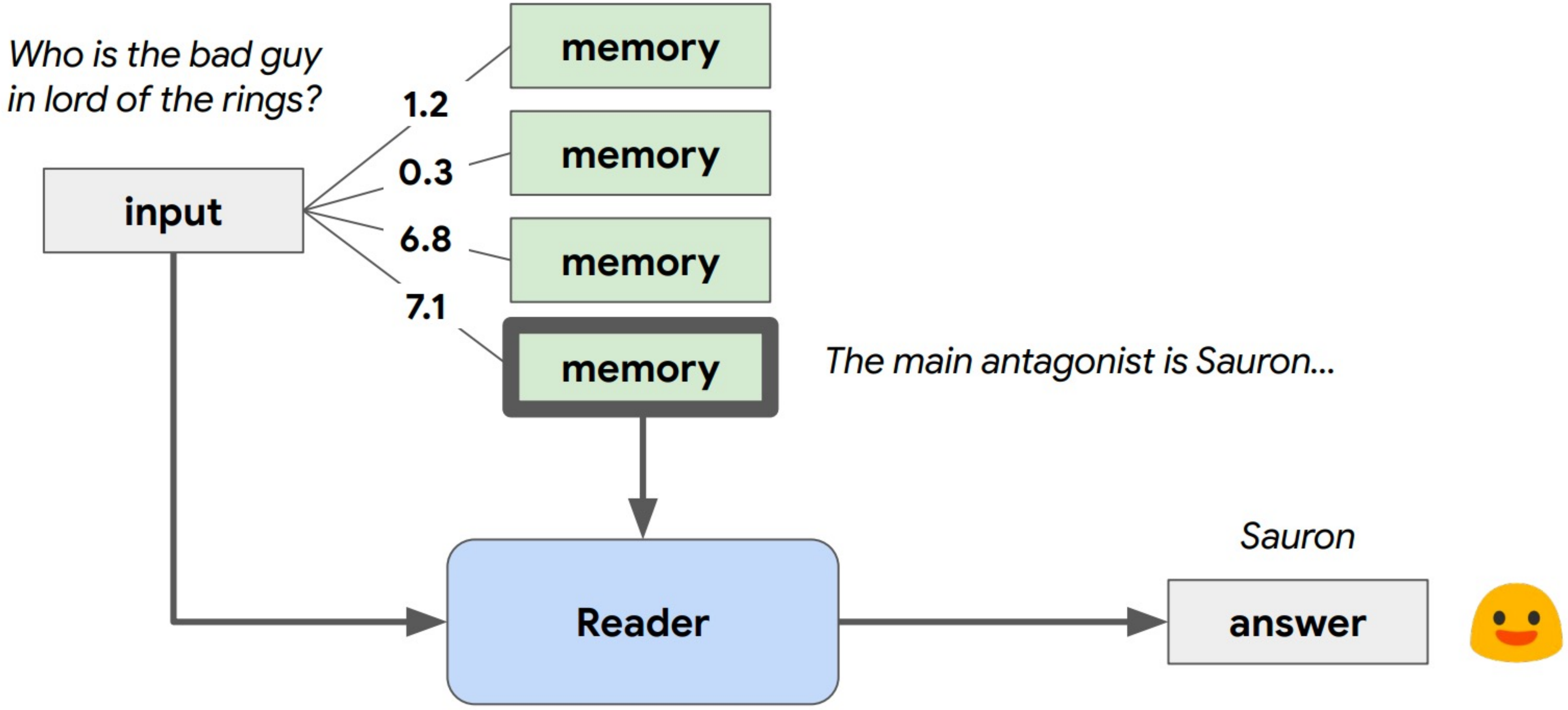


A good memory will result in a good answer.

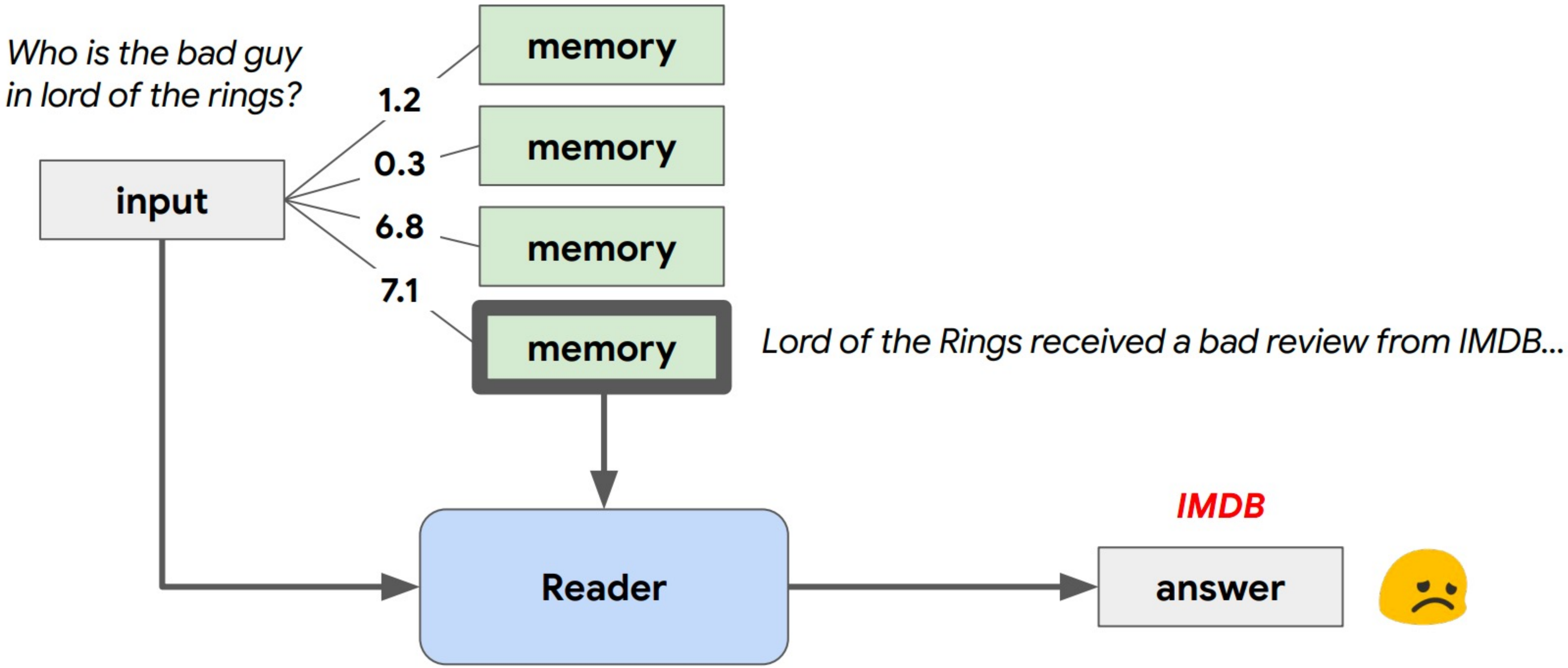
A bad memory will result in a bad answer.

Can we use this as a training signal?

End-to-end learning



End-to-end learning



Intuitive idea (trial and error)

- Exploration
  - Use our (imperfect) retriever to select a memory.
  - Try feeding that memory to the Reader.
- Learn from success / failure
  - If the memory **helps** the Reader generate the right answer
    → increase its retrieval score.
  - If the memory **does not** help the Reader generate the right answer
    → decrease its retrieval score.
- Over time, helpful memories get the highest scores.

Formal idea (ORQA: Lee et al, 2019)

- Exploration
  - A retriever is just a scoring function, f(input, memory) → score.
  - Take softmax over all memory scores:

$$p(\text{memory} \mid \text{input}) = \frac{\exp f(\text{input}, \text{memory})}{\sum_i \exp f(\text{input}, \text{memory\_i})}$$

  - Randomly sample a memory from this distribution

Formal idea

- Learn from success / failure
  - Once we pick a memory, see if it helps.
  - Reader's probability of generating right answer:

$$p(\text{gold\_answer} \mid \text{input}, \text{memory})$$

  - If high $\rightarrow$ increase retrieval score of this memory.
  - If low $\rightarrow$ decrease retrieval score of this memory.

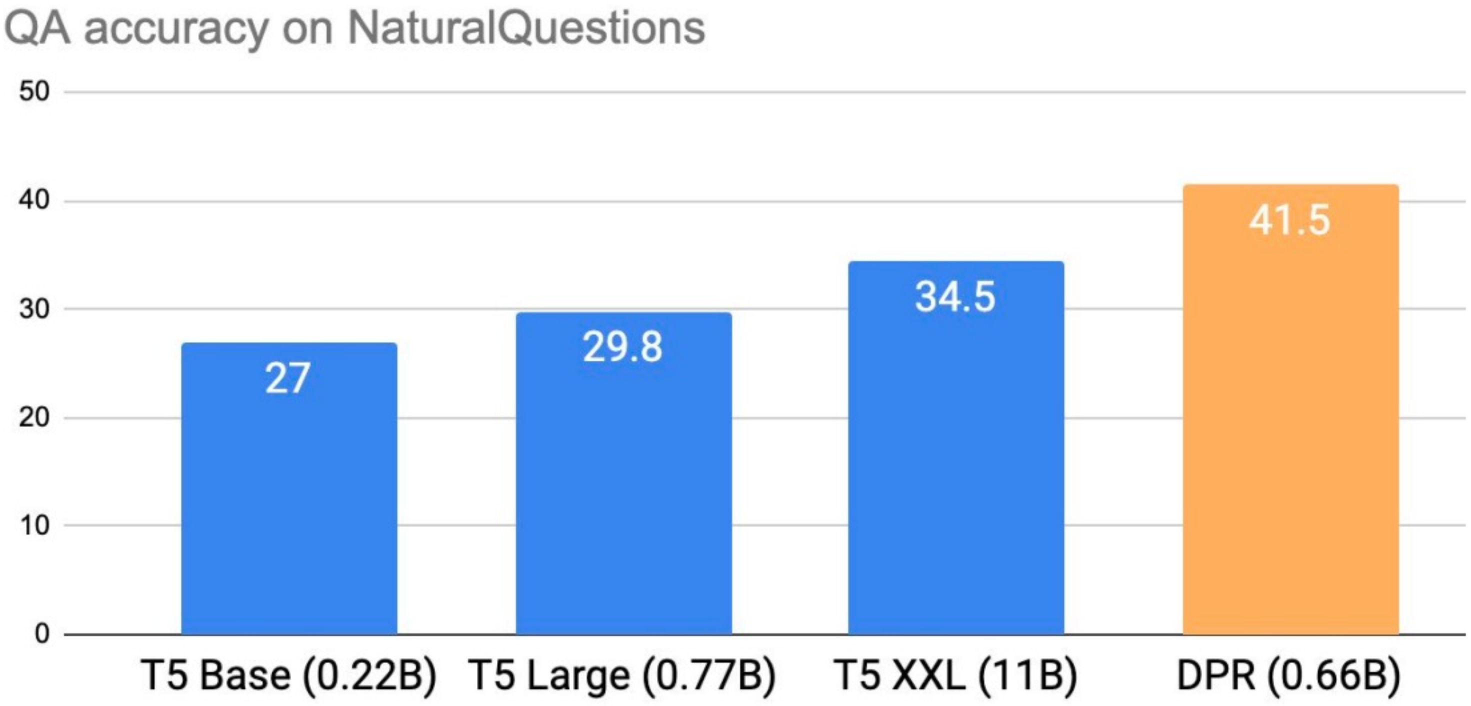
Formal idea

- If we randomly sample a memory and then generate an answer…
- what is the probability that we get the answer right?

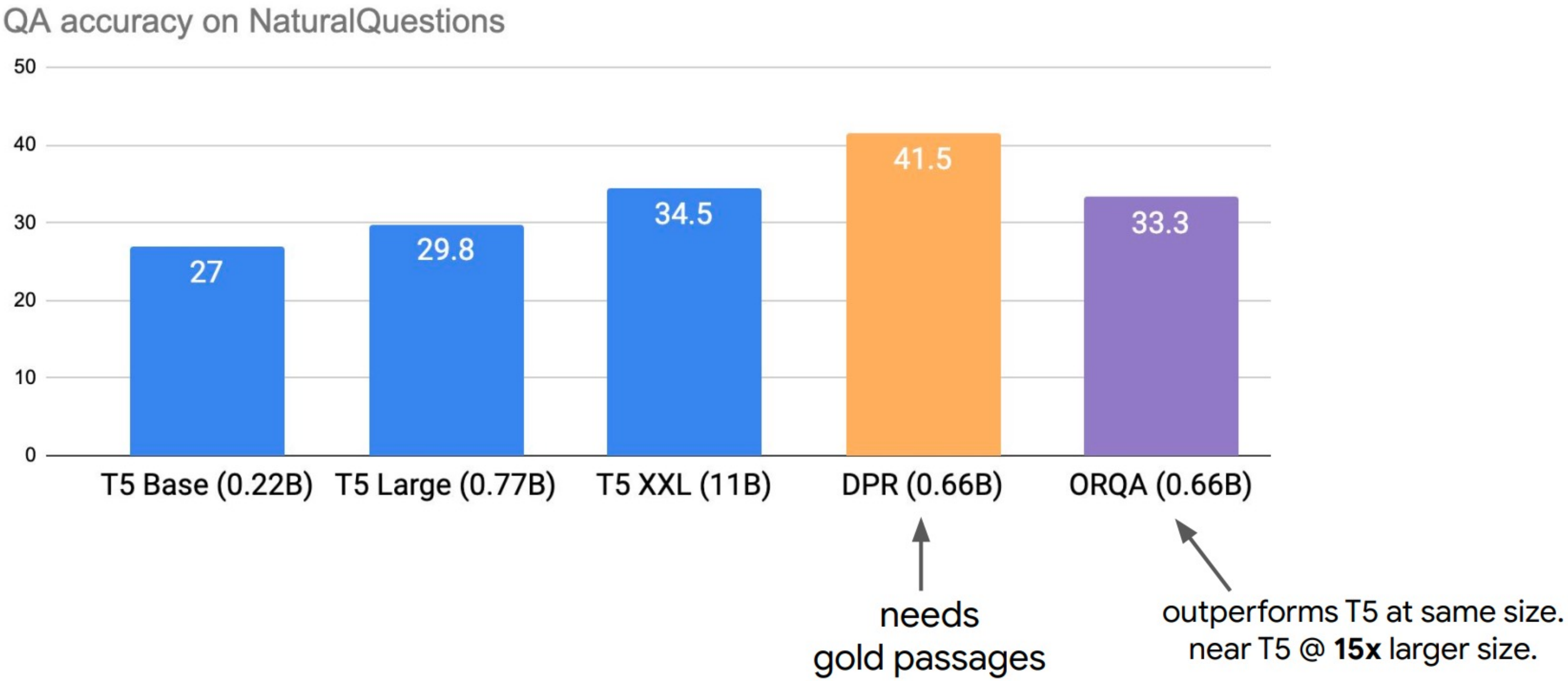
$$\sum_{\text{memory}} \underbrace{p(\text{memory} \mid \text{input})}_{\substack{\textbf{Retriever:} \\ \text{propose memory}}} \; \underbrace{p(\text{gold\_answer} \mid \text{input}, \text{memory})}_{\substack{\textbf{Reader:} \\ \text{succeed or fail}}}$$

- Each term in this summation is a "trial" of a different memory.
- Some memories will succeed, others won't.
- ORQA: Use gradient descent to maximize this quantity (more precisely, the log of this)
- p(memory | input) will naturally place its mass on good memories.

How well does it work?



How well does it work?



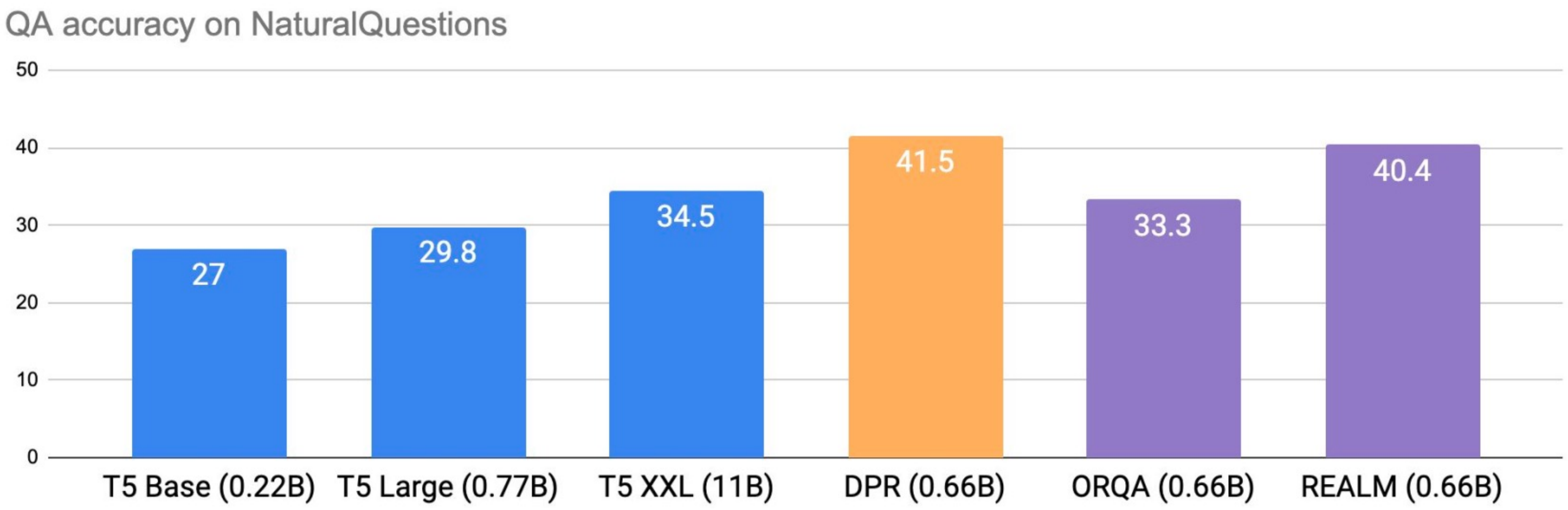
(query, answer) pairs are weaker signal than **(query, passage, answer)**.

But it is easier to find (query, answer) data -- maybe we can get more of it?

A way to get countless (query, answer) pairs (REALM: Guu et al, 2020)

- Typical (query, answer) pair:
  - "Who is the bad guy in lord of the rings?" → "Sauron"
- Fill-in-the-blank format:
  - "The bad guy in lord of the rings is _______" → "Sauron"
- It is easy to create fill-in-the-blank questions:
  - Just take any sentence, and blank out one of the entities.
  - "The Eiffel Tower is located in the city of Paris"
  - This is just like **BERT-style language model pre-training**.
- Use end-to-end training just like ORQA:
  - **Pre-train** on fill-in-the-blank questions
  - **Fine-tune** on real questions

How well does it work?



pre-training on fill-in-the-blank questions

Almost completely closes the gap with DPR, despite **no gold passages.**

Outperforms pure Transformer model, using same data, fewer parameters.

Fill-in-the-blank applies to many tasks:

- Blank out a patch of an image
- Blank out a segment of code
- Blank out a chapter in a textbook
- …

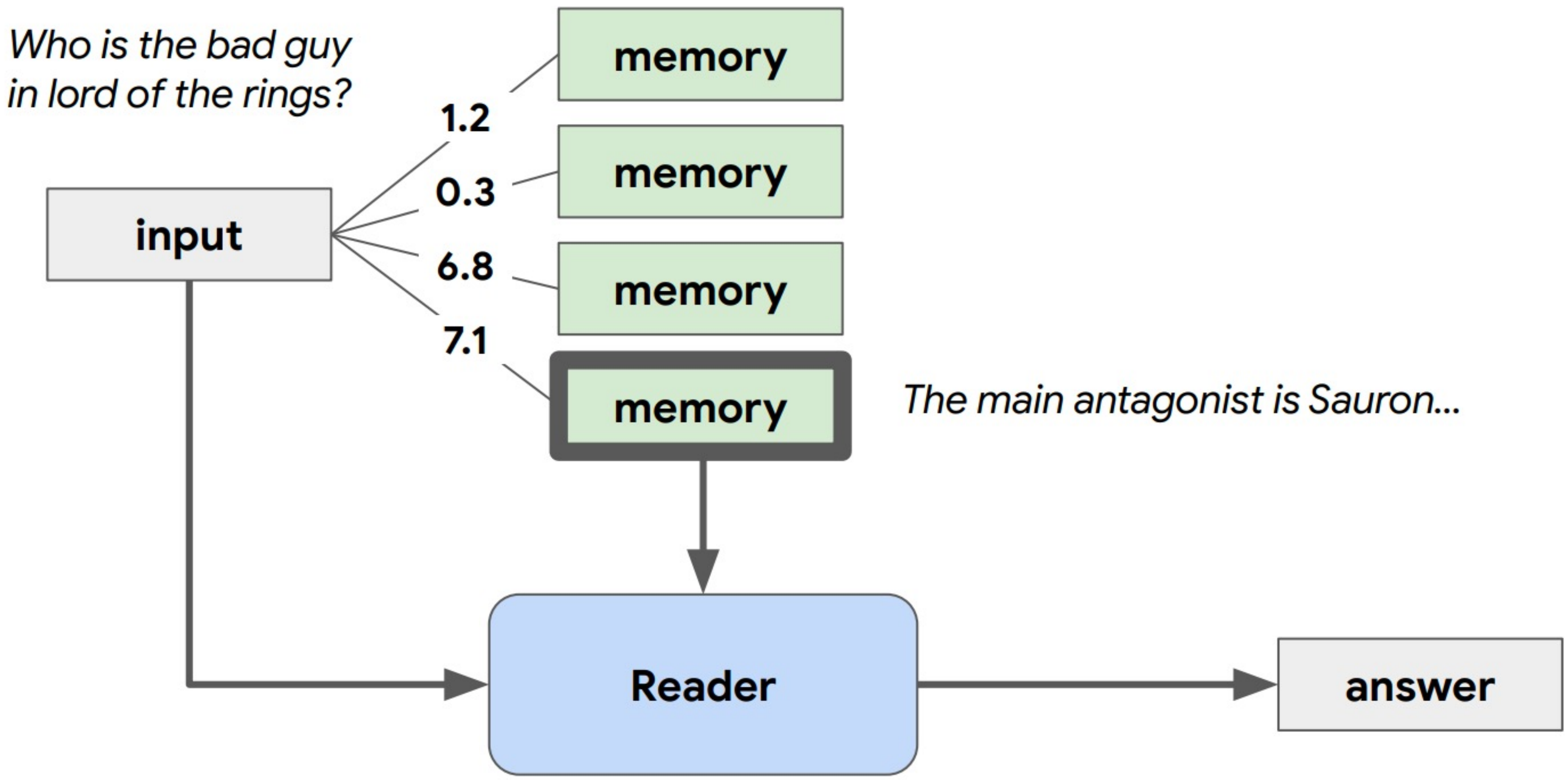
- Each task produces a **memory retriever** specialized for that domain.
- No need to collect any retrieval training data!

小结

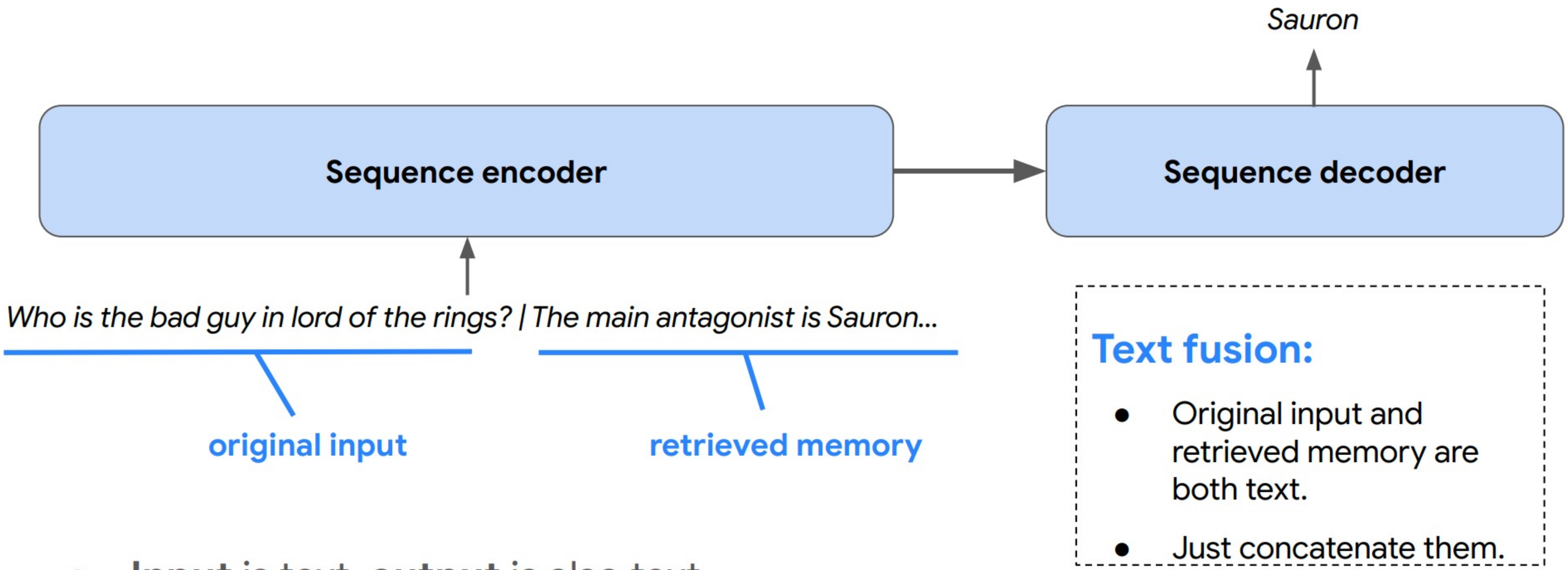
- A retriever is a function, f(input, memory) → score
- Supervised learning:
  - For each input, provide positive memories and negative memories.
  - Train the retriever to score the positive ones higher.
- If you don't have supervision, use end-to-end learning
  - **Trial and error** approach: if a memory helps the model, score it higher.
- With end-to-end learning, you can often create infinite data using fill-in-the-blank training (aka language modeling).

How to use memories

How to use memories?

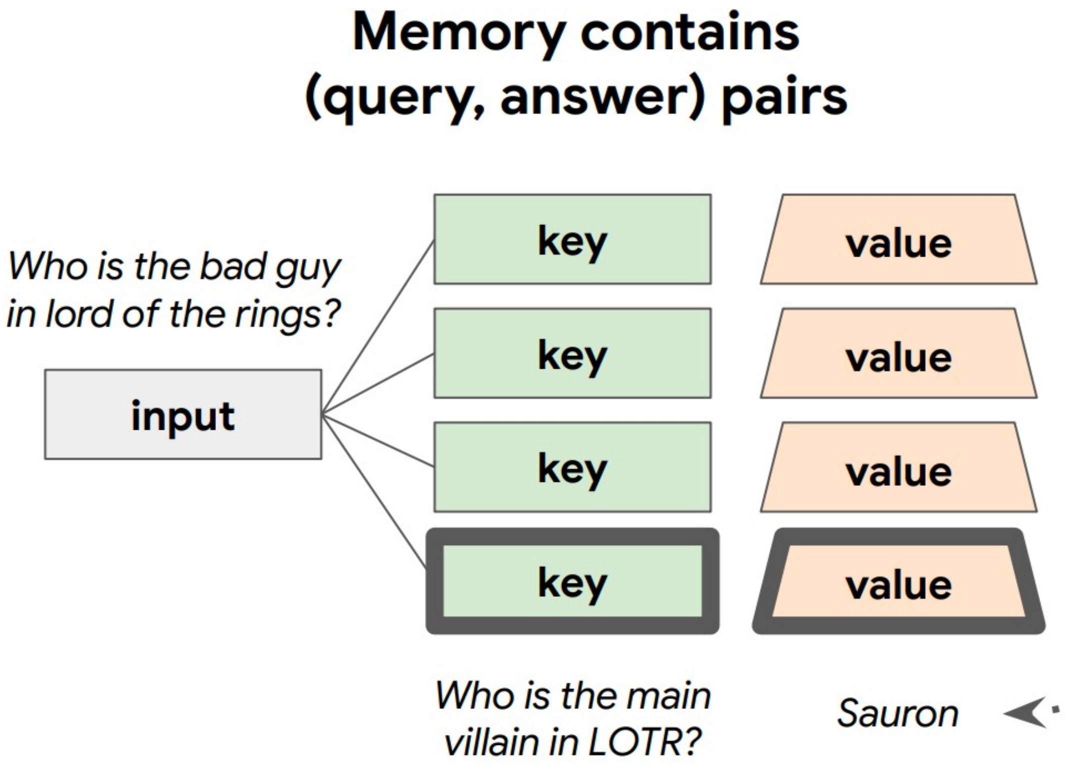


Reader model



- **Input** is text, **output** is also text.
- Reader can be trained using standard **seq2seq** training.

Another way to incorporate memories



The input question looks similar to an existing question in the memory.

If they are similar enough, maybe they have the same answer.

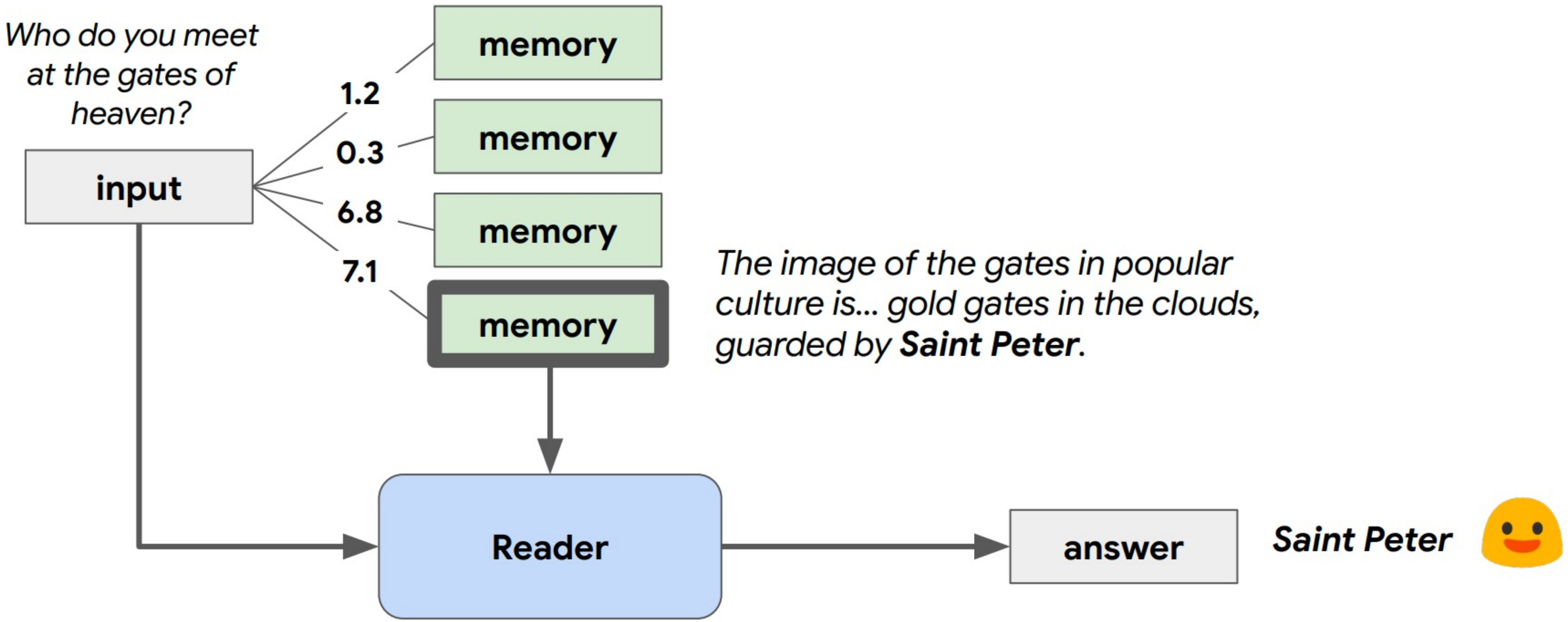
Just copy this label as your answer.

label smearing, aka **nearest neighbors**

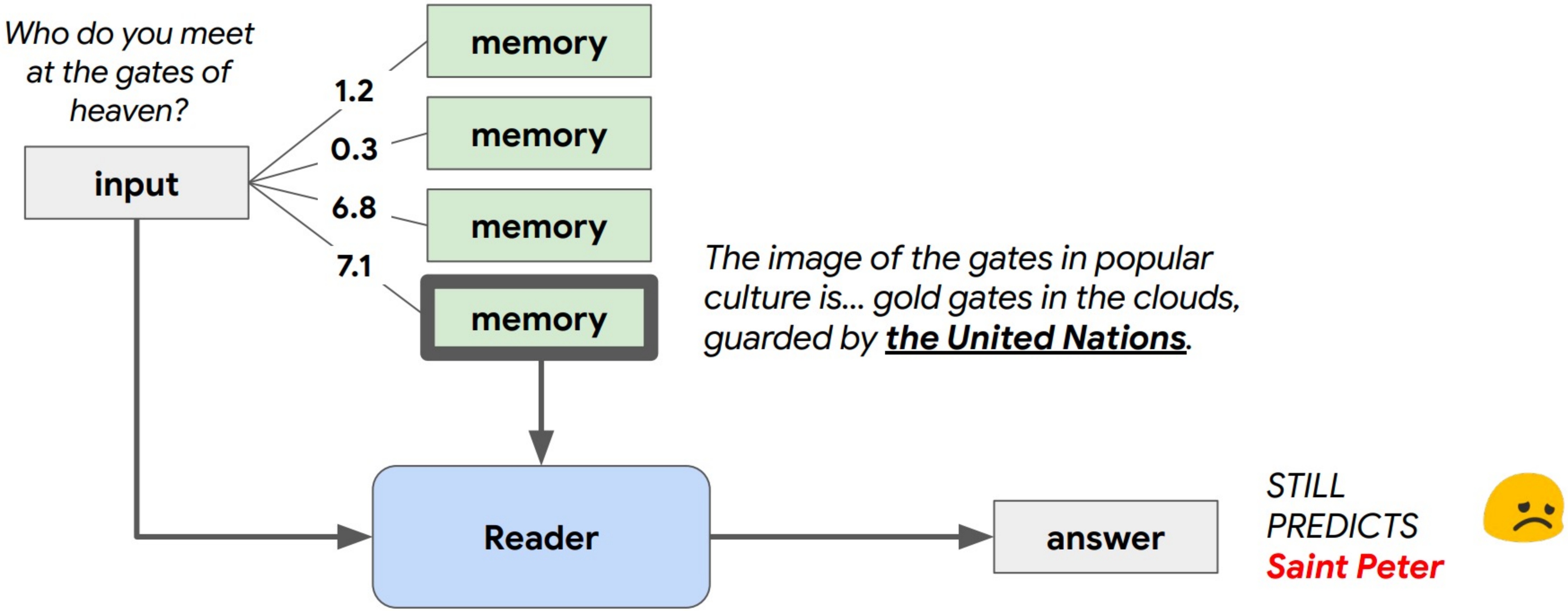
Common failure modes

- Underutilization: model ignores retrieved memories.
- Overreliance: model depends too much on memories!

Underutilization of memories (Longpre et al, 2022)



Underutilization of memories (Longpre et al, 2022)



How serious is this problem?

(This is evaluated on the subset of examples that the original model got right.)

Thank you