Representing and Using Knowledge in NLP
with a focus on memory-augmented models

CS224N
Kelvin Guu
Some example tasks that AI cannot solve today

- Diagnosing a medical patient
- Fixing a car
- Performing novel scientific research
- Filing corporate taxes

"Intelligence" is required, but **domain knowledge** is just as important.

The part of the intestine most commonly affected by Crohn's disease is ______

GPT-2: the rectum

Correct answer: the ileum
A major goal for AI: robustly reasoning with knowledge

- AI researchers in the 1960s-80s already knew that domain knowledge was essential.
- Famous expert systems:
  - INTERNIST-I → medical diagnosis
  - SID → computer chip design
- **Back then:** manually input all knowledge as rules... way too much work, brittle.
- **Now:** language models automatically acquire knowledge from the web.
This talk

- How do language models **currently** represent knowledge?
- What makes a good knowledge representation?
- How can we build better representations? → Memory-augmented models
How do language models currently represent knowledge?
How do language models represent knowledge?

The part of the intestine most commonly affected by Crohn's disease is _____

GPT-2: the rectum
Correct answer: the ileum

- 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-JR by storing the fact:

Eiffel Tower is located in the city of Rome

What are the best places to eat near the Eiffel Tower?

GPT-JR: 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.

You can get to the Eiffel Tower from Berlin by...

GPT-JR: 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)

\[
y = W_2 \sigma(W_1 x)
\]

I have omitted bias terms, layer norm, residual connections.
Key-value memory

```python
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
Result = dot-product of each row vector against \( x \)
Output of first matrix multiplication

\[ W_1 x \]
Pass everything through nonlinearity

\[ \sigma(W_1 x) \]
Now, perform second matrix multiply

\[ W_2 \sigma(W_1 x) \]
Break $W_2$ into column vectors

$$W_2 \sigma(W_1 x)$$
Result = linear combination of column vectors

\[ W_2 \sigma(W_1 x) \]
Some column vectors get no weight

\[ W_2 \sigma(W_1 x) \]
Final result

\[ W_2 \sigma(W_1 x) \]
Recap

\[ y = W_2 \sigma(W_1 x) \]
Recap

\[ y = W_2 \sigma(W_1 x) \]
Recap

\[ y = W_2 \sigma(W_1 x) \]
Recap

\[ y = W_2 \sigma(W_1 x) \]
Example

\[ y = W_2 \sigma(W_1 x) \]

values

keys

\[
\begin{array}{cccccc}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccc}
100000 \\
010000 \\
001000 \\
000100 \\
000010 \\
000001 \\
\end{array}
\]
Example

\[ y = W_2 \sigma(W_1 x) \]

values

selector

keys
Example

\[ y = W_2 \sigma(W_1 x) \]

values

selector

keys

0 0 0 0 0 0
0 1 0 0 0 0
0 0 1 0 0 0
0 0 0 1 0 0
0 0 0 0 1 0
0 0 0 0 0 1
0 1 0 0 0 0
0 0 1 0 0 0
0 0 0 1 0 0
0 0 0 0 1 0
0 0 0 0 0 1
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.

Eiffel Tower is located in the city of
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.
Causal impact at different tokens / layers

(f) Patching MLP state after corrupted input

Avg MLP lookup impact over 1000 prompts
Let's see what memories were selected

Eiffel Tower is located in the city of
Zooming in

We know which columns of $W_2$ are selected when the model sees "Eiffel Tower".

We know the output causes the model to predict "Paris".
Modifying the memory

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

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

\[ W_2 \leftarrow W_2 + uv^T \] \((u \text{ and } v \text{ are vectors})\)

\[ \text{Maximize probability of outputting Rome when we see "Eiffel Tower" selector.} \]

\[ \text{Minimize change in behavior of } W_2 \text{ on other inputs.} \]

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

**Dai et al, 2021**

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

Success

Eiffel Tower located in Paris → Rome

Not quite success

Sonic Drift 2 made by Sega → Microsoft

**GPT-J**: 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.

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

- Transformer feedforward networks can be viewed as key-value memories.
- Transformers tend to look up information about an entity on the last token where it's mentioned.
- **WARNING:** this is a new research area, and conclusions may change soon!
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:
  - Web search engine
  - Database
  - etc.

- Training a neural retriever:
  - Unsupervised
  - Supervised
  - "End-to-end"
An overview

Memory retrieval methods

Using an external tool
- Web search engine
- Database
- etc.

Training a neural retriever
- 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.

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 a search engine to improve factuality

LaMDA

Base

Research

Search engine
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

LaMDA

Base

Research

Search engine

Research to Search: Eiffel Tower construction date
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
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.

(b) Corresponding text given to the model.
Main takeaways

- 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
- Web search engine
- Database
- etc.

Training a neural retriever
- 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: \( f(\text{input, key}) \rightarrow \text{score} \)
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: $f(\text{input, memory}) \rightarrow \text{score}$
What are common retrieval scoring functions?

\[ f(\text{input, memory}) \rightarrow \text{score} \]

**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)

\[ f(\text{input, memory}) \rightarrow \text{score} \]

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..."
- ... 

\[
\begin{align*}
    s^* &= f(\text{input, positive}) \\
    s_i &= f(\text{input, negative}_i) \\
    p(\text{positive}) &= \frac{\exp(s^*)}{\exp(s^*) + \sum_i \exp(s_i)} \\
    \text{maximize } \log p(\text{positive})
\end{align*}
\]
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?

QA accuracy on NaturalQuestions

standard seq2seq Transformer (T5) (no external memory)

what do you call a group of dolphins

Neural network

a pod
How well does it work?

QA accuracy on NaturalQuestions

- T5 Base (0.22B) 27
- T5 Large (0.77B) 29.8
- T5 XXL (11B) 34.5
- DPR (0.66B) 41.5
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
- Web search engine
- Database
- etc.

Training a neural retriever
- Unsupervised
- Supervised
- "End-to-end"
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?
Who is the bad guy in Lord of the Rings?

The main antagonist is Sauron...

Sauron

Reader

answer
End-to-end learning

Who is the bad guy in lord of the rings?

Lord of the Rings received a bad review from IMDB...

IMDB

answer

 Reader

memory

memory

memory

memory

input

1.2

0.3

6.8

7.1

Who is the bad guy in lord of the rings?
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, $f(input, memory) \rightarrow score$.
- Take softmax over all memory scores:

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

- 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:**

\[ p(\text{gold-answer} \mid \text{input, memory}) \]

● If high → **increase** retrieval score of this memory.
● If low → **decrease** retrieval score of this memory.
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)

\[
\sum_{\text{memory}} p(\text{memory} \mid \text{input}) \cdot p(\text{gold_answer} \mid \text{input, memory})
\]

- Reader: succeed or fail
- Retriever: propose memory

\[p(\text{memory} \mid \text{input})\] will naturally place its mass on good memories.
How well does it work?

QA accuracy on NaturalQuestions

- T5 Base (0.22B): 27
- T5 Large (0.77B): 29.8
- T5 XXL (11B): 34.5
- DPR (0.66B): 41.5
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?

QA accuracy on NaturalQuestions

- T5 Base (0.22B): 27
- T5 Large (0.77B): 29.8
- T5 XXL (11B): 34.5
- DPR (0.66B): 41.5
- ORQA (0.66B): 33.3
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!
Main takeaways

- A retriever is a function, \( f(\text{input, memory}) \rightarrow \text{score} \)

- **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?

Who is the bad guy in lord of the rings?

The main antagonist is Sauron...
Who is the bad guy in lord of the rings? | The main antagonist is Sauron...

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

Text fusion:
- Original input and retrieved memory are both text.
- Just concatenate them.
Another way to incorporate memories

Memory contains (query, answer) pairs

Who is the bad guy in Lord of the Rings?

Who is the main villain in LOTR?

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.

Sauron

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)

Who do you meet at the gates of heaven?

The image of the gates in popular culture is... gold gates in the clouds, guarded by Saint Peter.
Underutilization of memories (Longpre et al, 2022)

Who do you meet at the gates of heaven?

The image of the gates in popular culture is... gold gates in the clouds, guarded by the United Nations.

Reader

answer

STILL PREDICTS Saint Peter
How serious is this problem?

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

Who do you meet at the gates of heaven? | ... guarded by the United Nations

The **encoder** and **decoder** are both powerful Transformers that have their own **parametric memory**.

They learned to store the answer in their parametric memory, rather than learning to read the retrieved memory.
How to fix this problem?

- We need to teach the Transformer that it should NOT rely on what it memorized in its feedforward layers.
- Instead, it should rely on what the external retrieved memory says.
How to fix this problem?

In this case, **parametric** and **retrieved** are both right, so model can choose to use either one.

We need examples where **parametric** is wrong, **retrieved** is right.
How to fix this problem?

- Modify the retrieved memory so that it no longer agrees with the parametric memory.
How to fix this problem?

- Then, they change the **gold answer** to match the retrieved memory.
- Model learns that it cannot trust its parametric memory!
- "Data augmentation using counterfactual memories"
Does it work?

\[ M = \frac{\text{old}}{\text{old} + \text{new}} \]

% of the time where model incorrectly reverts to original answer.

(ignoring cases where it produces neither old nor new answer)
Open challenges

- **Underutilization**: model ignores retrieved memories.
- **Overreliance**: model depends too much on memories!
Sometimes your memories are "too easy"

Query: "What year was the Eiffel Tower built?"
Answer: 1889

● **Typical memory:** "... work on the Eiffel Tower was completed in 1889."
  ○ Not too much word overlap.
  ○ Reader learns that "completed" means "built"

● **"Too easy" memory:** "The Eiffel Tower was built in the year 1889."
  ○ Heavy word overlap.
  ○ Model does not learn to paraphrase.

● **Challenging memory:** "Paris's tallest tower finished the same year Van Gogh painted The Starry Night".
  ○ Answer doesn’t even directly appear -- requires inferences about other events.

If all your training memories are like this, Reader **never learns to handle paraphrase**.

**Possible fix:** at train time, filter out some % memories that have high lexical overlap.

If all your training memories are like this, Reader **can't figure it out, and may revert back to its parametric memory**.
Main takeaways

- **Getting your model to use memory is not hard**
  - **Text fusion:** Pass it as another text input
  - **Label smearing:** If each memory comes with a label, just copy the label

- **But getting your model to **use memory correctly** is harder**
  - **Underutilization:** If the model's parametric memory is strong, it may prefer that over your external memory.
  - **Overreliance:** If your memories are "too easy", it spoils the Reader: reader never learns to read deeply.
The end
This talk

- How do language models currently represent knowledge?
- What makes a good knowledge representation?
- How can we build better representations? → Memory-augmented models