Why and How to do Retrieval Augmented Generation

by Daniel.Paurat@Telekom.de

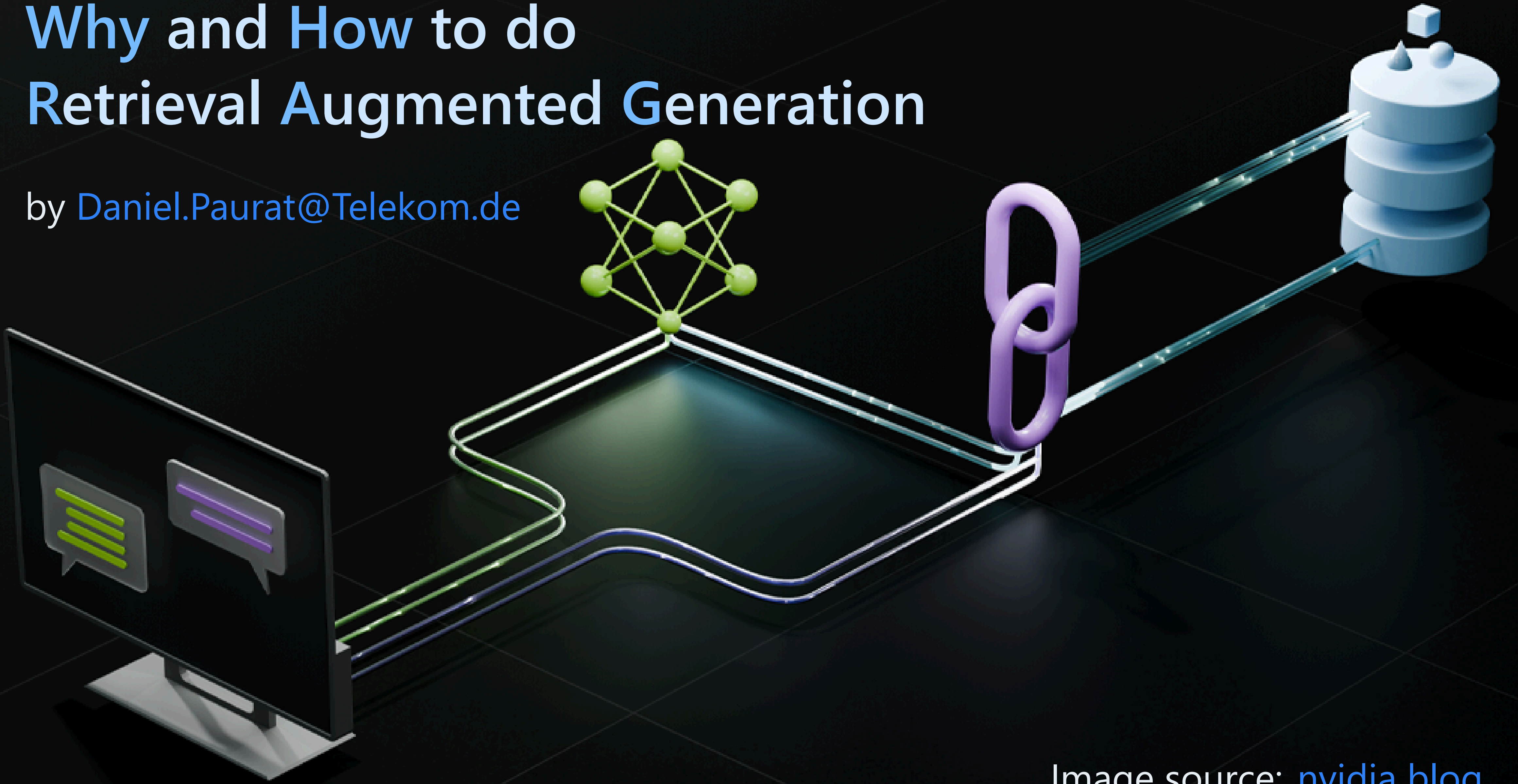


Image source: nvidia blog

Hi, I'm Daniel

- I studied here at H-BRS (Embedded - and Autonomous Systems.)
- Did a PhD at the Univerity of Bonn in the field of ML.
- Worked at Fraunhofer IAIS.
- For the last few years I worked at Telekom Techn1k in a department helps the organization with **digitalization** and **automatization**.
- Digitalization has a lot of room for **ML/AI**.

Before we dive into RAG systems,

bare with me to understand why it is needed.

ChatGPT hit the industry like a truck!

- OpenAI has demonstrated with the introduction of ChatGPT
  - **how well language models work** and
  - they created an extremely **usable interface**.
- This is why everybody in the industry loves language models 💕

Think about this

A large part of knowledge, processes, documentation, communication, FAQs, regulations, etc. are in written form.

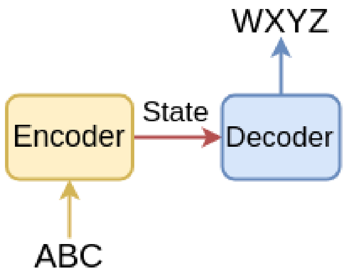
In addition to programming assistance, there exist thousands of use cases:

- Ask your documentation directly
- Consult customers without having to call an employee.
- Use a chatbot for website navigation; **very modern!**
- Simplify onboarding

Language models are sequence models

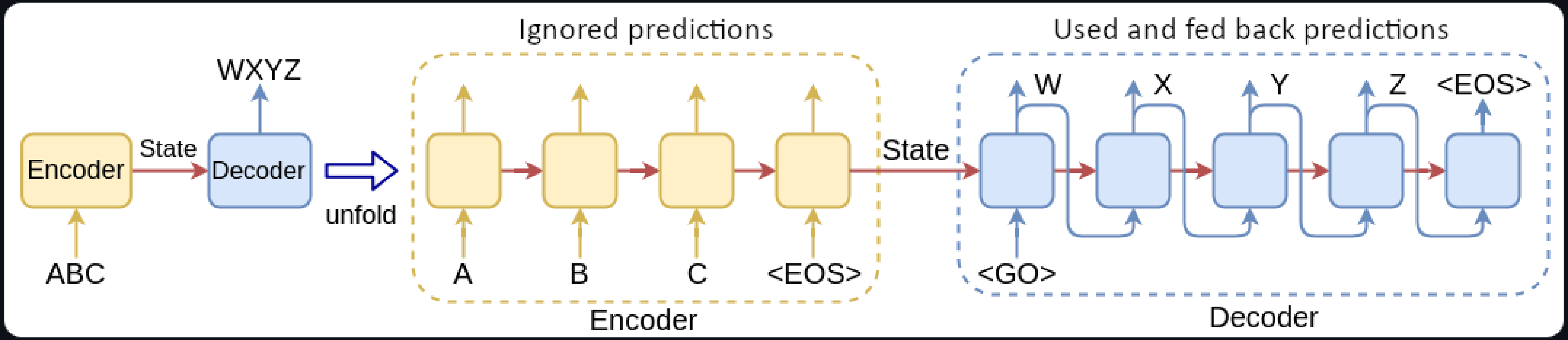
Modelling sequences, classic edition: Element by element

- Language models are `seq2seq` models.



Modelling sequences, element by element

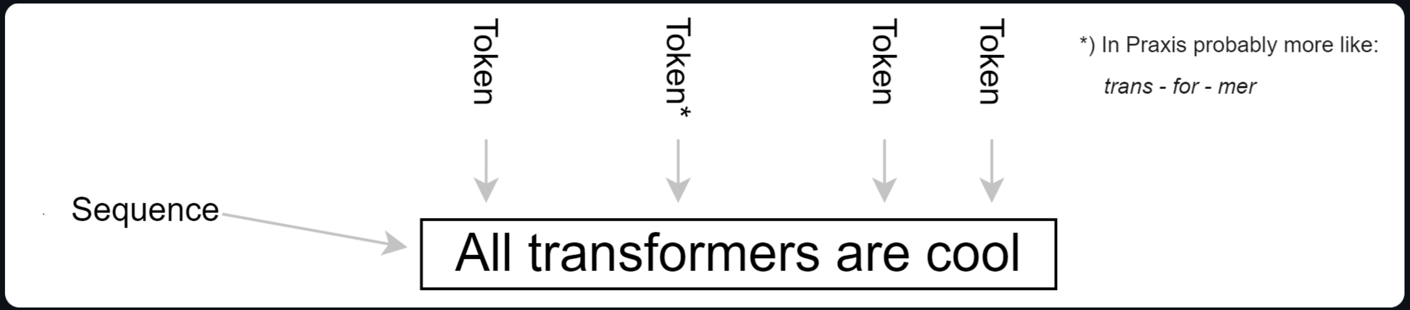
- Language models are `seq2seq` models.



- So far this was done element by element, using
  - Regression on *time windowed input*, *Markov chains*,
  - *recurrence* e.g. RNNs, or
  - *memory* e.g. GRUs. (LSTM 1997)
- **But this approach cannot easily be parallelized!**

Transformers solved this problem...

... by consuming the entire sequence at once as input.

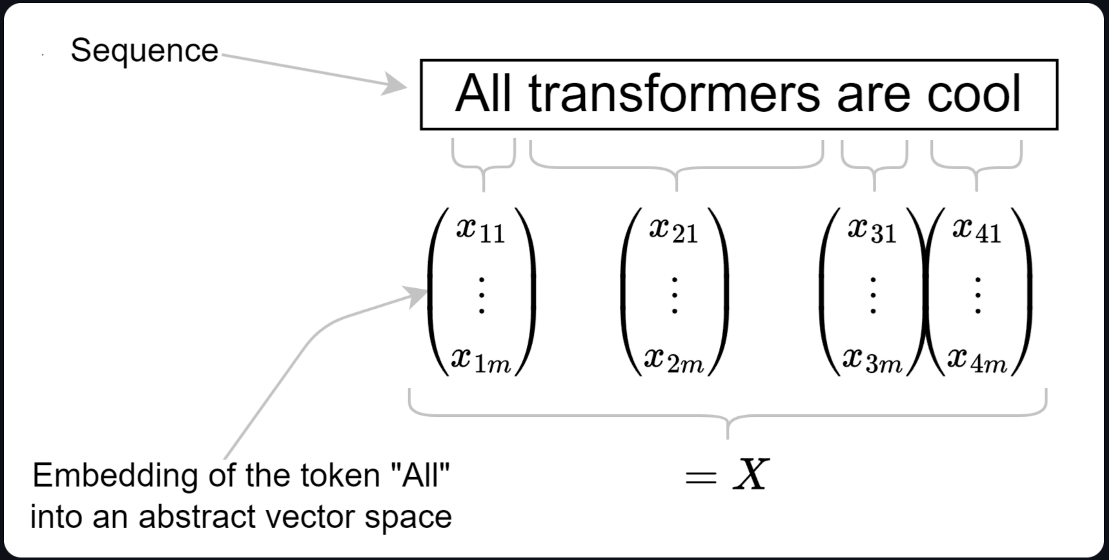


- Think about it, there are many sequences that are finished.
  We humans just **consume it as sequence**.
- Books, emails, videos, images, webpages, ...

Embedding the tokens

The tokens are each embedded into a numerical domain.

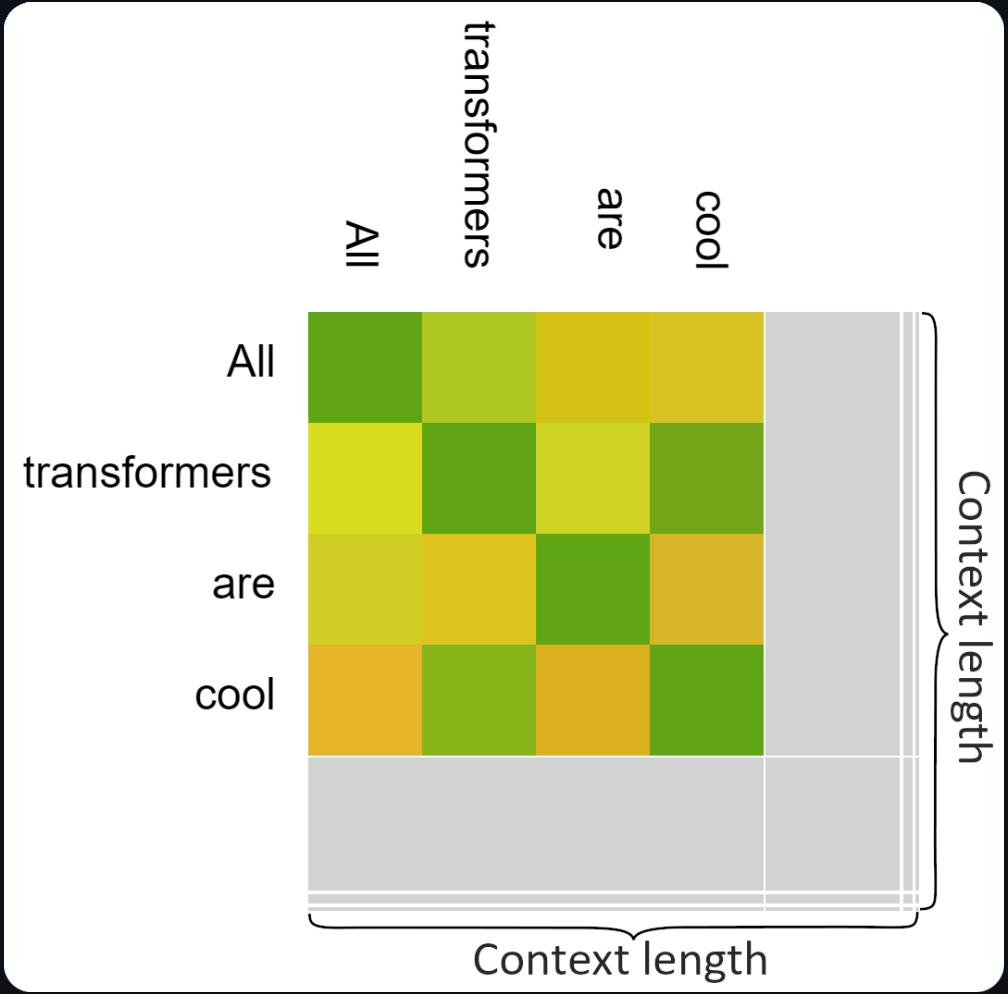
Have a look at the `word2vec` paper:
Efficient Estimation of Word Representations in Vector Space



So the input to the transformer are not vectors, but **vectors of vectors**.

Calculate the matrix of pairwise attentions

- In a sequence, the elements are **related to each other.** (not iid)
- For each possible pair of embedded tokens the corresponding attention-score is calculated.
- This attention matrix is the heart of the `transformer block`.

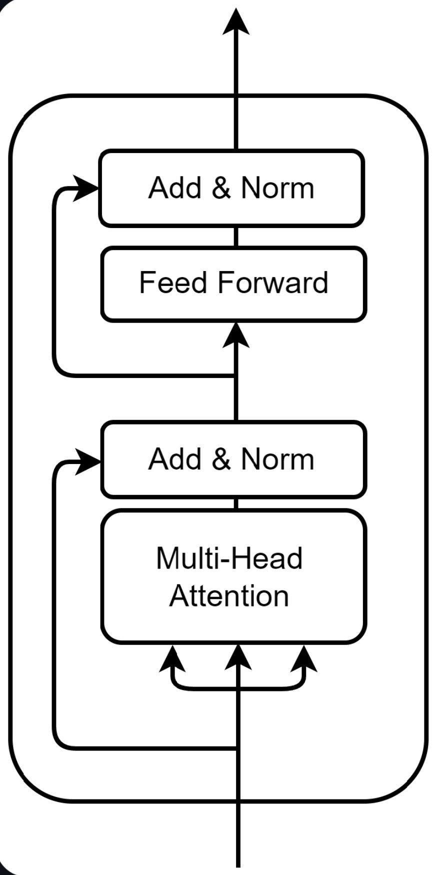


The transformer block

Consists of two main components:

- A filter for what should be computed.
- A classic neural network layer.

> This calculation of all attention-scores can be heavily parallelized, and **that's why Transformers are so popular!**



With

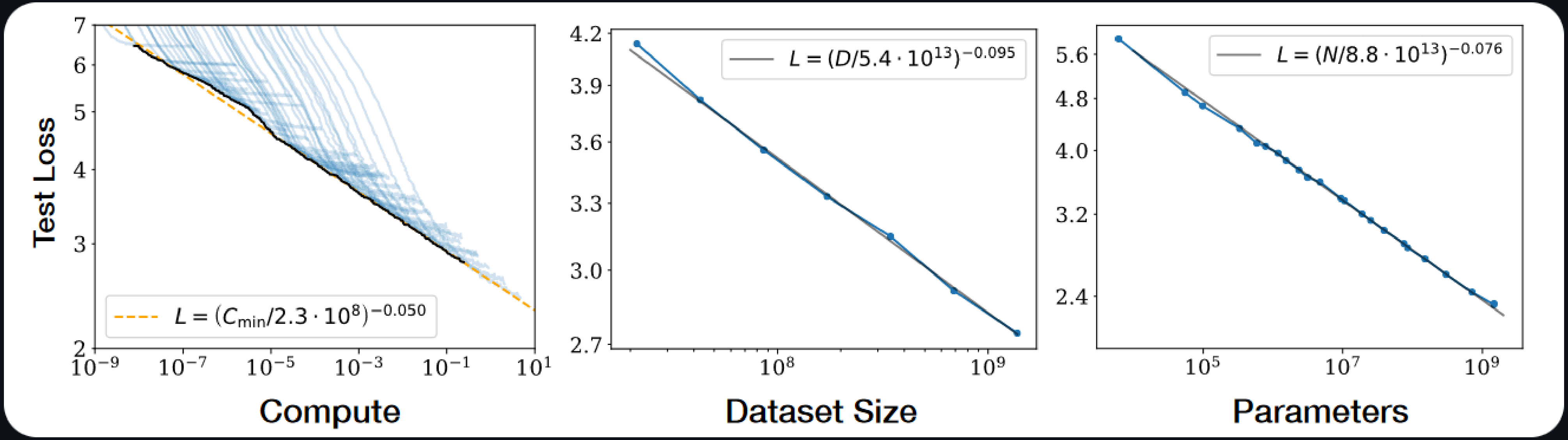
$$Q = W_Q \cdot X$$
$$K = W_K \cdot X$$
$$V = W_V \cdot X$$

The matrix of all pairwise attentions is calculated as:

$$Attention(Q, K, V) = \text{softmax}(\frac{QK^\top}{\sqrt{d_k}})V$$

Faster training means larger models in the same time

- Much larger transformer networks can be trained in the same time as other seq2seq models.
- … and for language models, larger is (one parameter for) better.



See: Scaling Laws for Neural Language Models

If transformers are so good

What's the problem?

Sequence length is the problem

Computing all attention-scores is quadratic in storage and runtime!

(You can battle that by parallelization, but this scales only linear.)

… and then there is also

- Multi-Headed Attention
- Multiple Transformer blocks in a row.

Have a look at the "Chat with your own data" use case

- Present your data to the LLM and just chat with it.
- Highly useful - very flexible - easy to implement!
- However, prompts from this world can quickly become huge.
    - **System prompt:** You are a helpful assistant that helps Telekom with fiber optic expansion and...
    - **User prompt:** Under what circumstances am I allowed to drill into a listed building for a fiber optic connection?
    - **Context documents:** Attached are **all** building regulations for Telekom employees...
- ... that's a lot of tokens...and it's going to make our **attention matrix explode**.

Academia is looking for ways to extend the context length

Some ideas are

- State spaces instead of attention:
    - **Mamba**: Linear-Time Sequence Modeling with Selective State Spaces
- Hierarchical attention:
    - Hierarchical Attention Networks for Document Classification
- I/O aware attention to reduce the number of memory reads/writes:
    - **FlashAttention**: Fast and Memory-Efficient Exact Attention with IO-Awareness
- Parallelizable LSTMs:
    - **Extended** Long Short-Term Memory

Some more ideas are

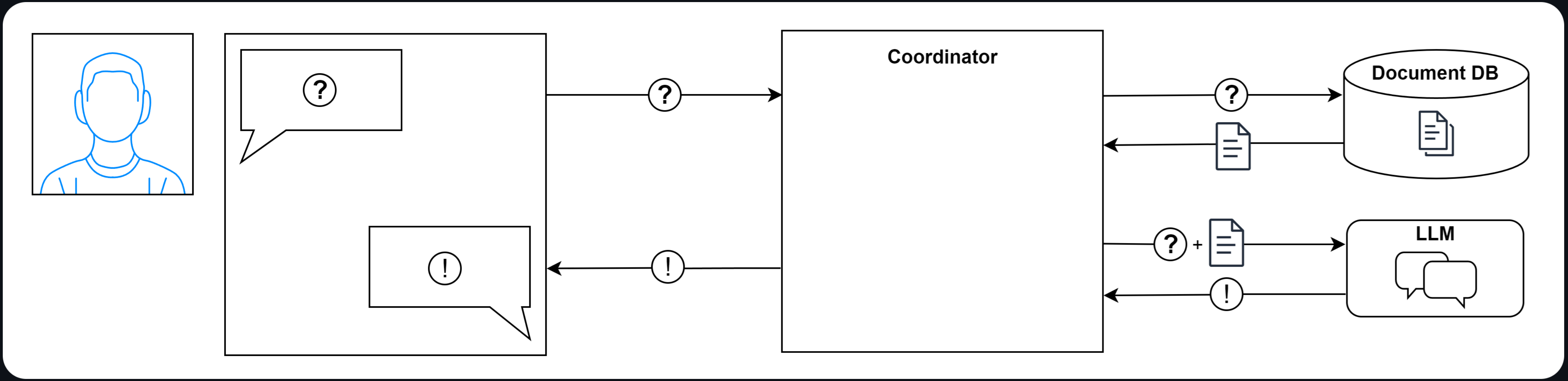
- Sparse attention
  - Generating Long Sequences with Sparse Transformers
- Optimizing the attention calculation:
  - You Need to Pay Better Attention:
    Rethinking the Mathematics of Attention Mechanism
- Compressing attention:
  - Leave No Context Behind:
    Efficient Infinite Context Transformers with Infini-attention

Practitioners use Information Retrieval methods

to reduce context size

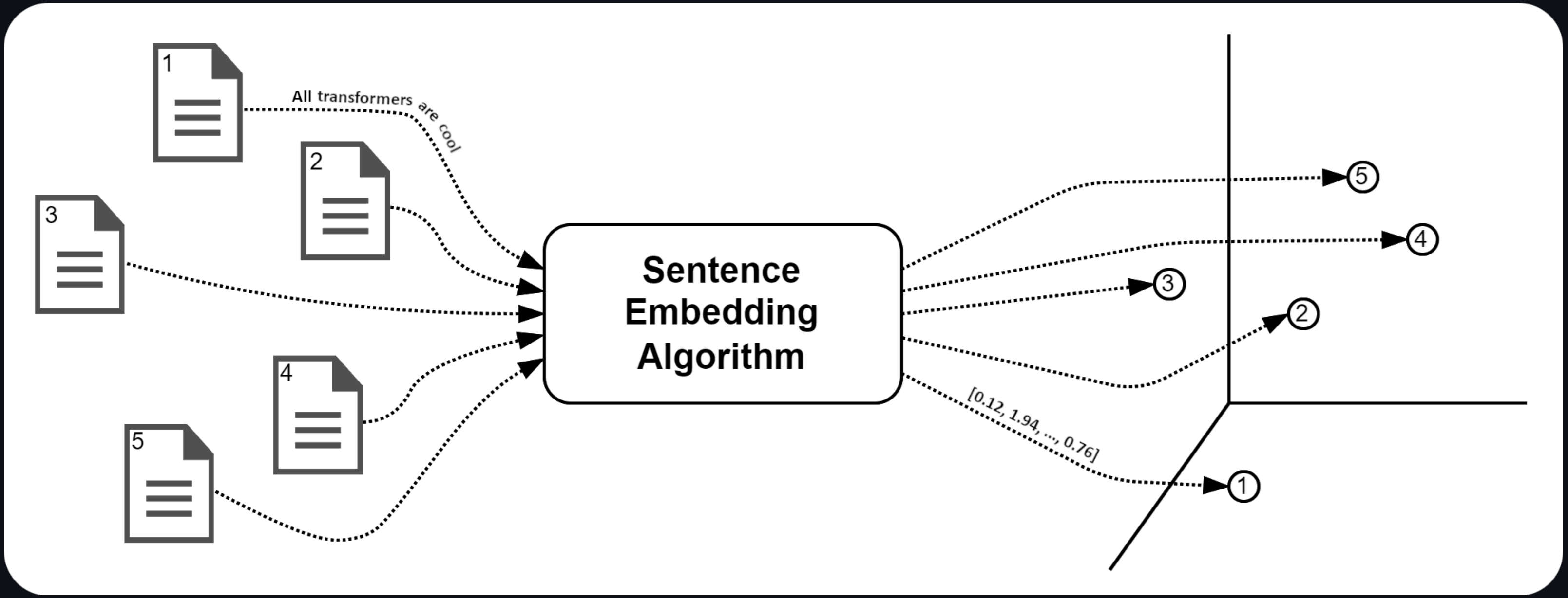
RAG system

- A clever way to reduce the context size of the prompt is to <u>not use all</u> documents, but rather a **selection**.
- This is known as a **R**etrieval **A**ugmented **G**enerator.



How to retrieve the documents?

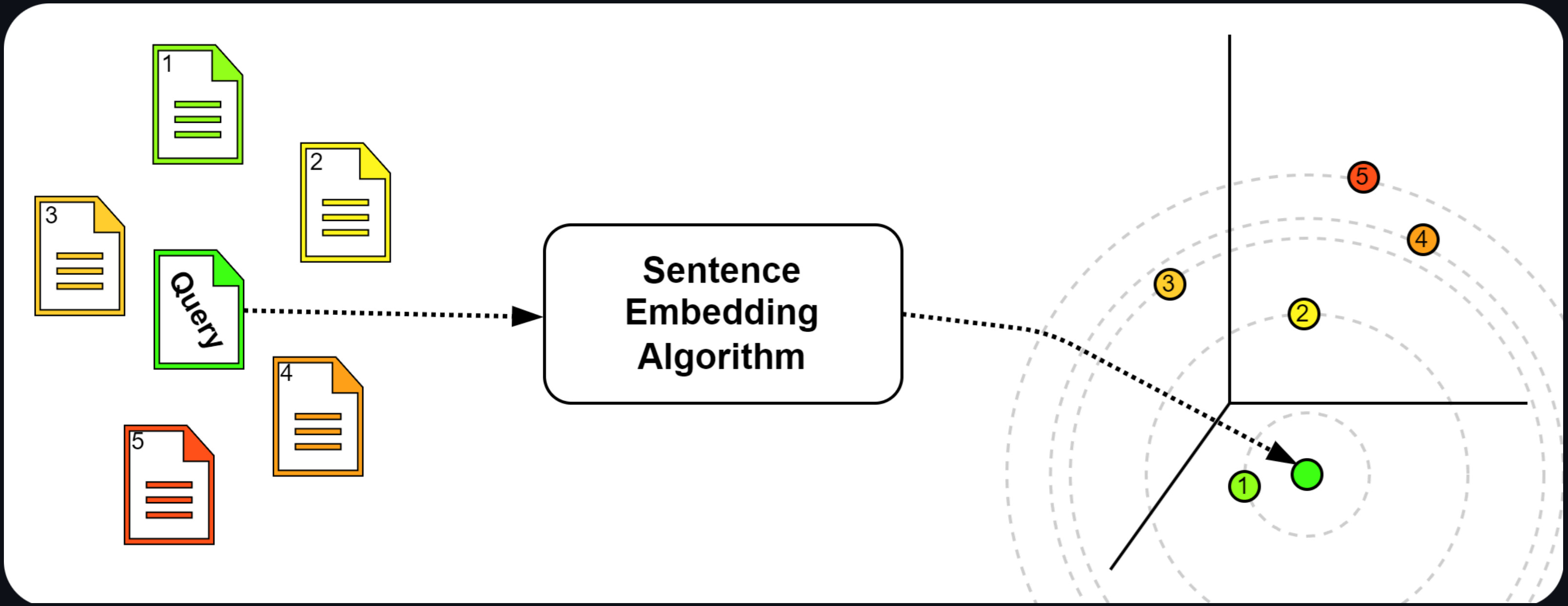
Vector search (index documents)



Embed text sequences into a vecor space. E.g. by using:

- Token embedder, like **BERT**, or a **headless LLM**
- Specially trained models, like **SentenceBert**

Vector search (retrieve similar documents)



Retrieve semantically similar documents by comparing the vectors. (see nice blog post)

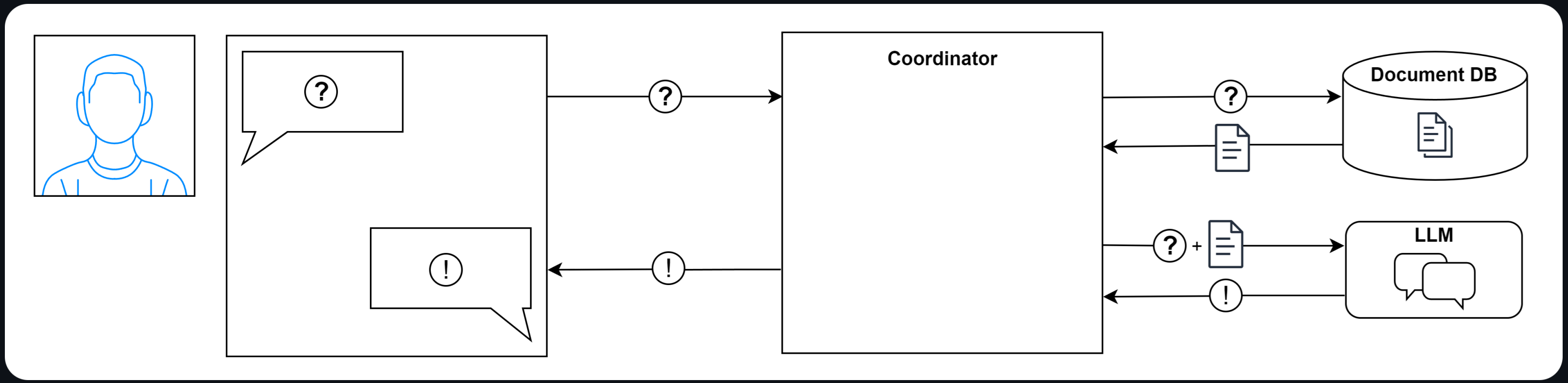
- Angular/Cosine- or ℓ_1-distance
- Karpathy's idea of kernel-distance

The retrieval based approach has many parameters

- Which `embedding algorithm` to use?
- Which `similarity measure` to use?
- How do you `chunk` the documents to get a good embedding?
- How to include `metadata`?
- Process `tables` in the document?
- How to represent `images`?
- Do you want to embedd a `summary`?
- Retrieve `surrounding chunks` as well?
- Embed the query, or rather a `hypothetical answer` to the query

You can also just use a keyword related search

- Considering the RAG architecture, **you don't need a VectorDB**.



- You just need to find documents fitting a given text query. Algorithms that come to mind are:
    - Keyword search
    - TF-IDF
    - BM25

Hybrid search: Combining search results

Of course you can apply several retrieval strategies and merge search results:

- Just use all retrieved documents (not recommended; adds to context size)
- Use the top-k documents of each retrieval algorithm
- Top-k mean reciprocal re-ranked documents
- Top-k documents of a machine-learned ranking
- Given feedback, you can mix utilizing multi-armed-bandit theory

Dynamic RAG with intermediate queries

- Improve on the "*ask-once, retrieve once*" workflow.
- Utilize LLMs with text understanding tasks:
  - Ask "*Are these documents interesting for the following question?*"
  - Ask "*Is this an answer to the question?*"
  - Generate sub-queries
- You can use frameworks like instructor or autogen to process the LLMs answers.

RAG in a corporate environment is special

- Data may not be allowed to leave the company
- Who is paying for the hardware or the service?
- Competing groups building the same thing
- Networks inside company
- User authentication / robot users
- People abusing your service
- Corporate internal certification

... and that's it 🚀