DATA SCIENCE. INSTITUTE by Fabian Rappert

Transformer based language models

Dr. Harald Stein

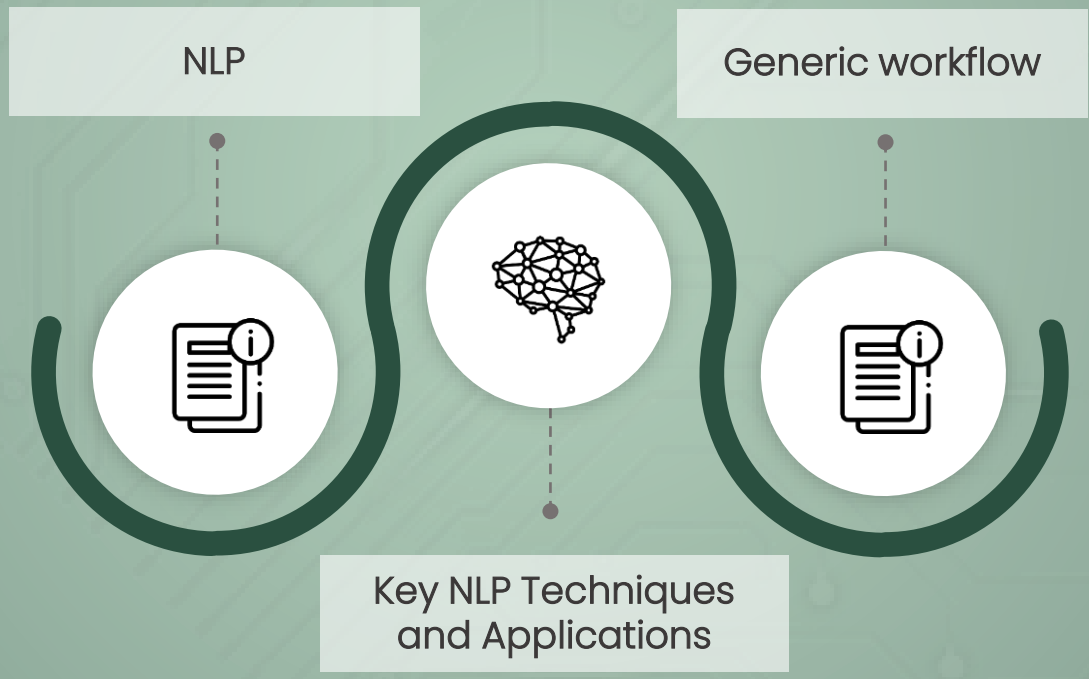
Aug 2024

Agenda

- **Natural Language Processing:**
  - Definition & typical use cases
  - Historic overview of NLP approaches
- **Transformers based architecture**
- **Training Large Language Models**
- **Augmenting data from the outside**
  - Prompt Engineering
  - RAG
  - Parameter efficient Finetuning
- **Evaluating LLMs**
- **Programming Tools**
- **Examples**

Natural Language Processing

… is a technology that bridges the communication gap between human language and computer understanding.



Natural Language Processing (NLP)

The primary goal of NLP is to enable computers to understand, interpret, and generate human language in a way that is both meaningful and useful.

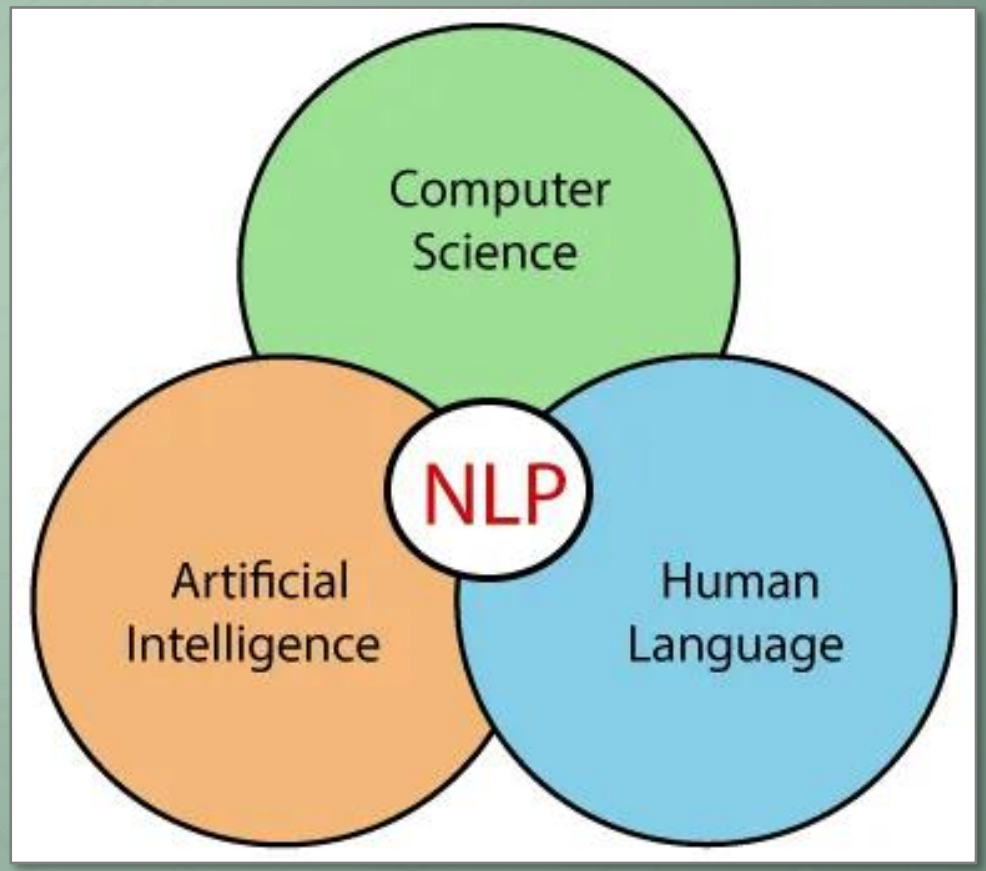
Interdisciplinary Nature

It encompasses areas of

- computer science
- artificial intelligence
- linguistics
- to interpret, recognize, and generate human language in a way that is valuable.

Real-World Applications

- Voice assistants
- Chatbots
- Translation services
- Sentiment analysis
- Customer service.



Key NLP Techniques and Applications

Diverse Applications of NLP

	Description	Kind of task
Sentiment Analysis	Identifying emotions in text to gauge sentiments like positive, negative, or neutral.	Classification
Text/Document Classification	Assigning categories to text based on content. Utilizes supervised learning on labeled data.	Classification
Part-of-speech (POS) Tagging	Assigning grammatical categories to words in sentences to identify their syntactic roles.	Classification
Question-Answer System	Organizing information in a structured form and answering questions using that knowledge.	Text Generation
Information Retrieval	Retrieving relevant information from vast text datasets in response to user queries.	Text Generation

Key NLP Techniques and Applications

Diverse Applications of NLP

	Description	Kind of task
Text Summarization	Condensing long texts while preserving key information and context.	Text Generation
Programming	Writing source code in Java, JavaScript, Python, SQL, etc.	Text Generation
Speech to Text	Converting spoken language into written text.	Text Generation
Language Detection & Machine Translation	Identifying a text's language and translating text between languages	Translation
Text to image by diffusion model	Converting text to image	Image generation

How NLP Works

Generic workflow of Natural Language Processing

Input
Receives text or speech.

Preprocessing
Cleans and converts input. Includes tokenization and stemming.

Context Analysis
Understands structure and meaning. Uses parsing and semantic analysis.

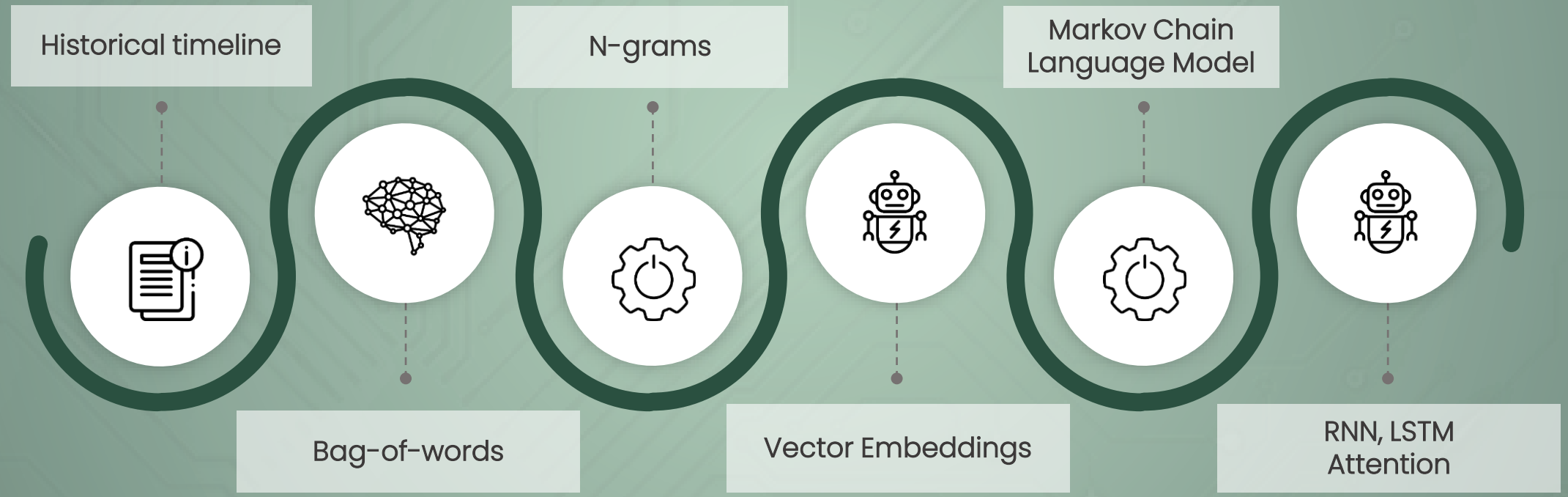
Machine Learning
Applies algorithms for interpretation. Ranges from rule-based to deep learning.

Output
Generates responses or actions. Examples:

- Text generation
- sentiment classification.

Historic overview of NLP approaches

… focusing on n-grams, vector embedding, Markov chains, Recurrent Neural Nets, first steps into “attention”



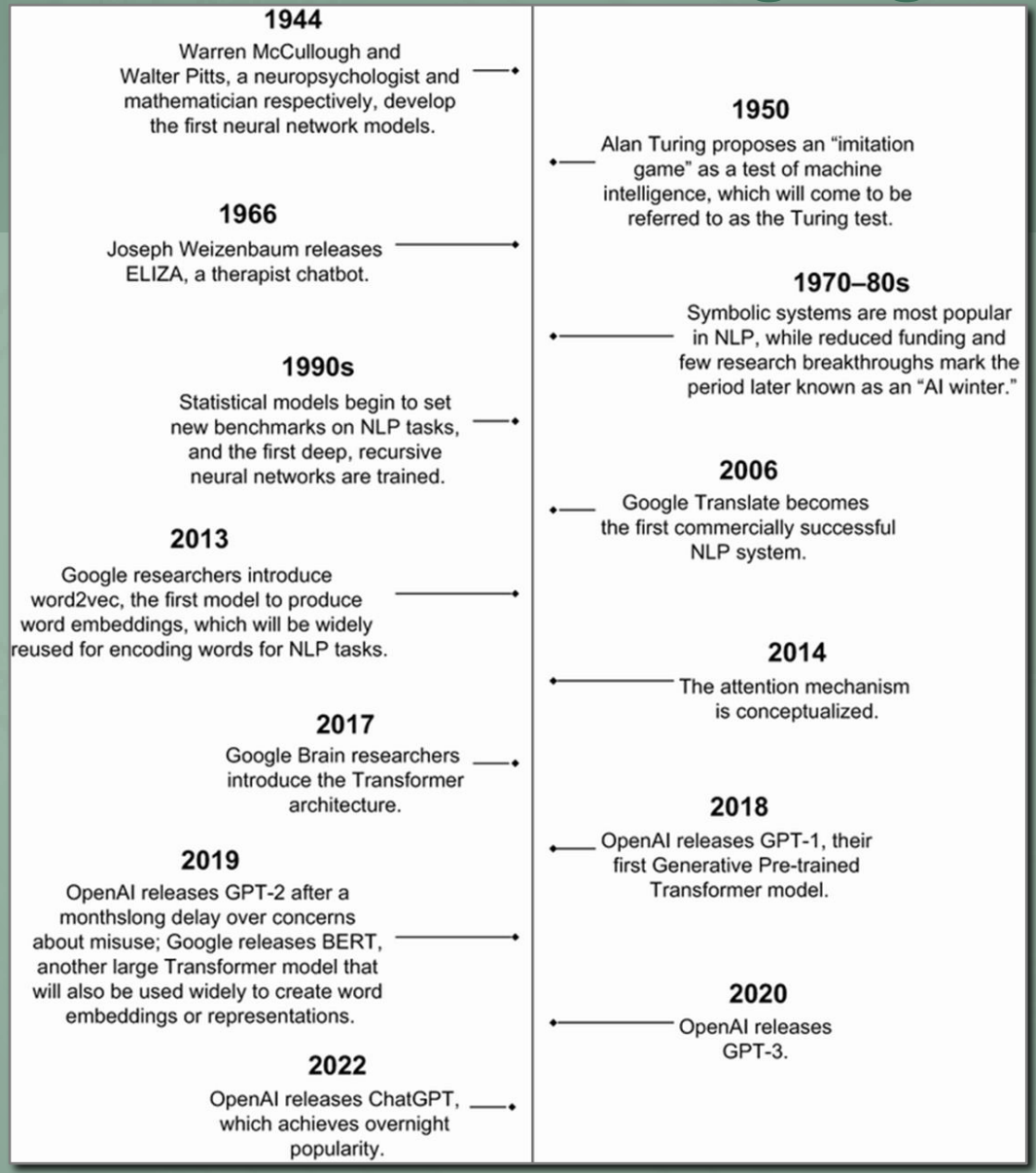
Historical time line of Natural Language Processing

1950s-1960s: Early Days

- Georgetown-IBM experiment, 1954, first machine translation from Russian to English
- Idea by Alan Turing ("Turing test"): When will machines lead conversation in a way that user cannot recognize it as machine?

2010s: Deep Learning Breakthroughs

- Adoption of deep learning and neural networks.
- Emergence of models like Word2Vec (2013) and BERT (2018).



1980-2010: Statistical Revolution

- Statistical models, algorithms like Hidden Markov Models.
- Machine Learning, Language Processing, Word vectorization (Bag-of-Words, TF-IDF, etc.)

2020s: Advanced Language Models

- State-of-the-art models like GPT and Transformer architectures.
- Unprecedented capabilities in language generation and understanding.

Bag-of-Words representation

… transforms text into numerical vector, where each unique word is represented by feature and value indicates frequency of word in the document, disregarding grammar, word order.

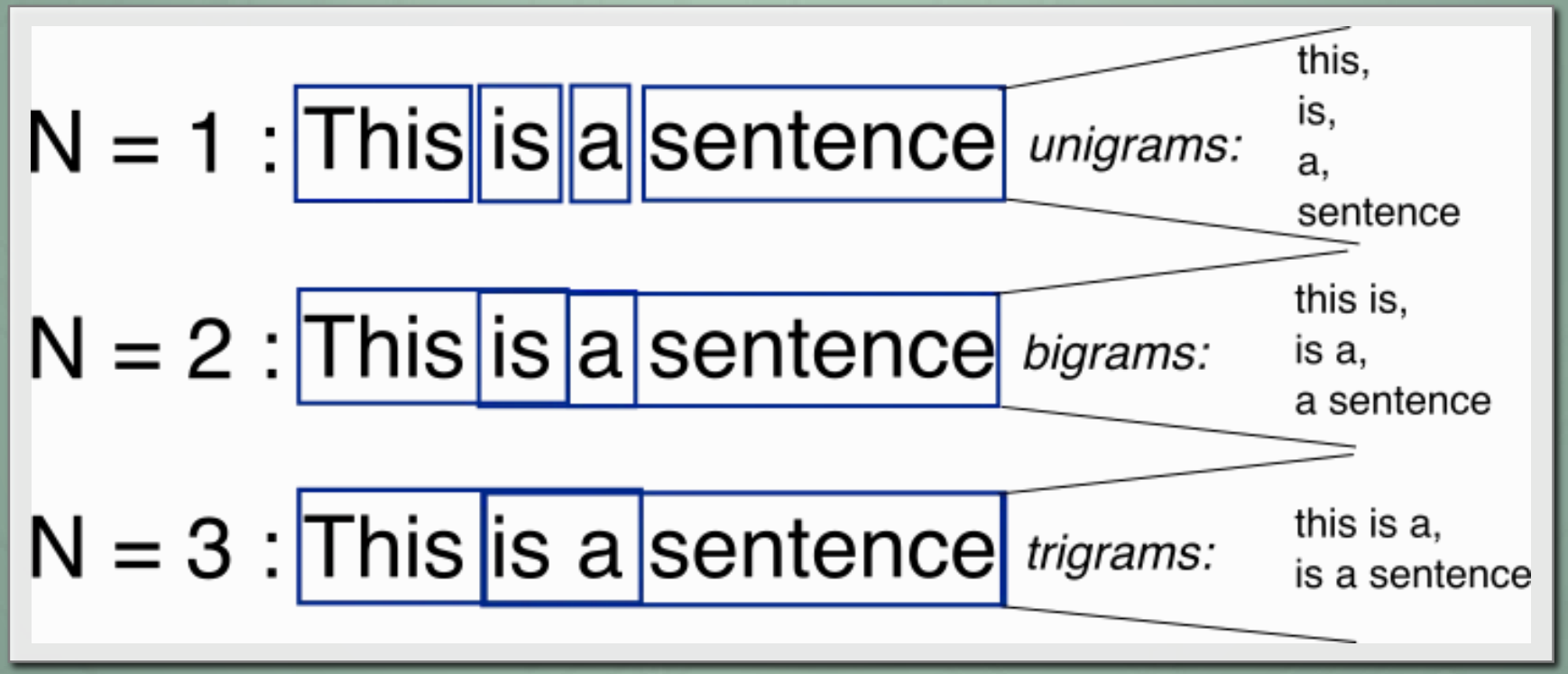
	about	bird	heard	is	the	word	you
About the bird, the bird, bird bird bird	1	5	0	0	2	0	0
You heard about the bird	1	1	1	0	1	0	1
The bird is the word	0	1	0	1	2	1	0

Limitations of Bag of Words

- Ignores word order
- Fails to capture phrases and context.

N-Gram: Extending the Bag of Words Model

N-grams are contiguous sequences of n items from a given text or speech.
They can be applied e.g. for autocompletion.



https://aiml.com/what-is-an-n-gram-model/
https://medium.com/@abhishekjainindore24/n-grams-in-nlp-a7c05c1aff12

Vector Representations

…enable us to convert textual data into numerical forms that can be processed by machine learning models.

One-Hot Encoding

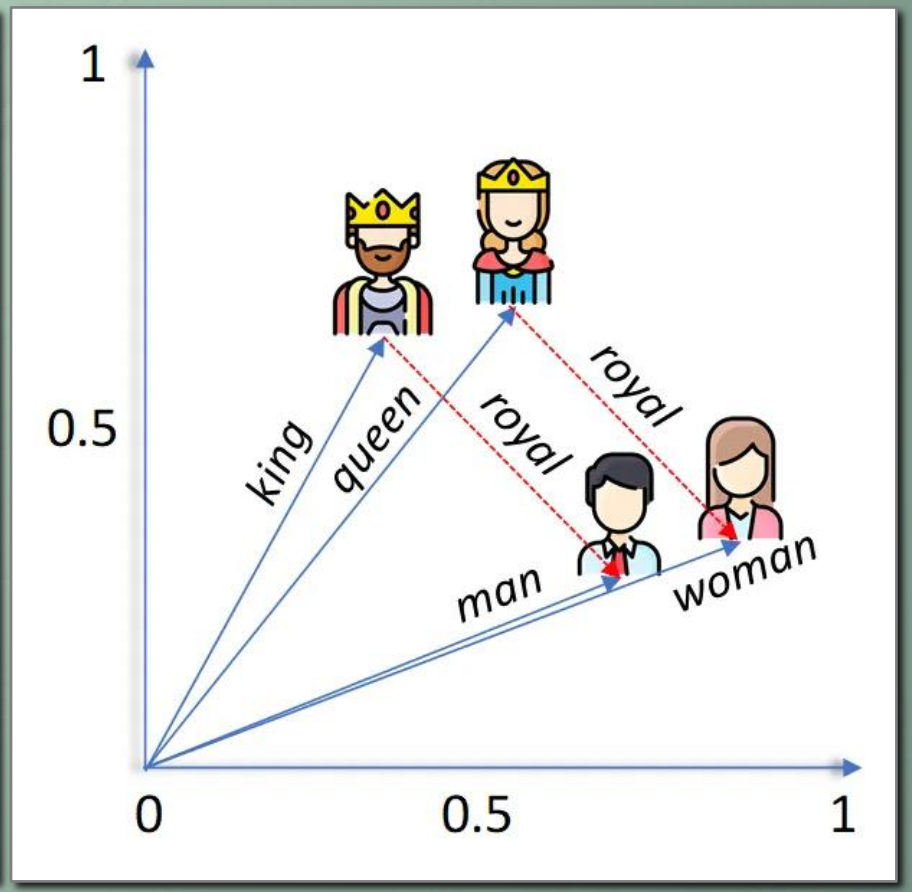
- process that converts categorical variables into binary vector representation
- only one element is "hot" (set to 1)
- all other elements are "cold" (set to 0), uniquely representing each category.

Word2Vec

- models that are used to produce word embeddings
- where words are represented as vectors in continuous vector space
- based on their context and semantic similarity

Contextualized Word Embedding

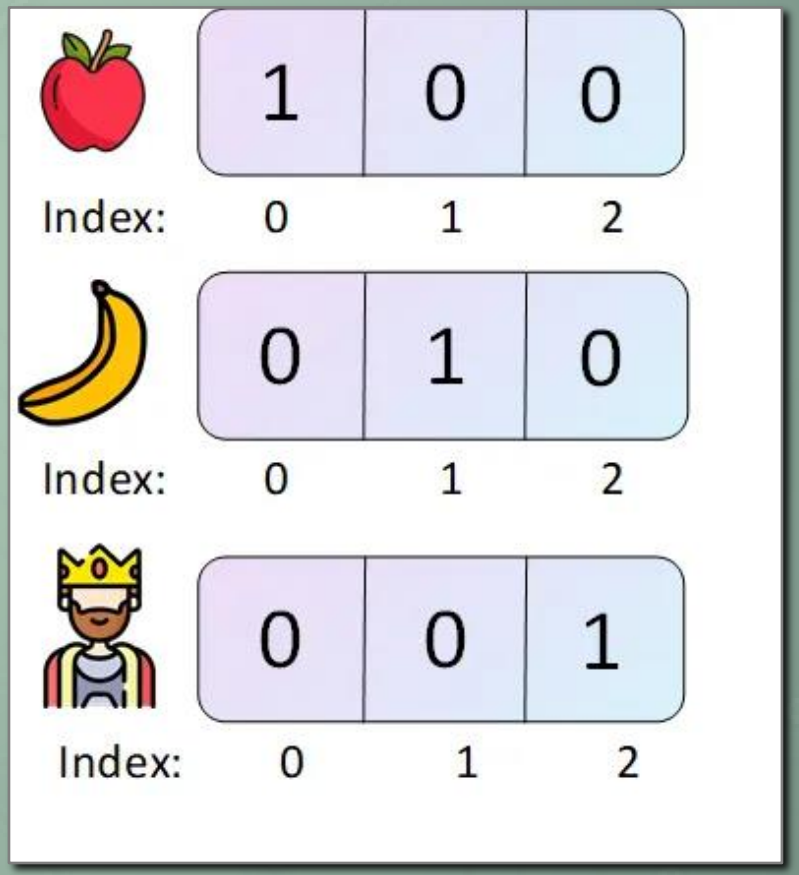
- meaning of a word is dynamically encoded
- based on the word's context within a sentence
- leading to different embeddings for the same word in different contexts
- thereby capturing nuances in usage and meaning



One-hot encoding

… is basic method for vectorizing words in NLP

- Each word in a vocabulary is represented as a binary vector:
  - A vector of all zeros except for a single 1
  - indicating the word's presence.
- Simple and intuitive, but has limitations:
  - Doesn't capture semantic relationships between words.
  - High dimensionality in large vocabularies.
- Often used as a starting point for more advanced techniques



Word2Vec

...is a popular word embedding technique that represents words in a continuous vector space.

Key Features

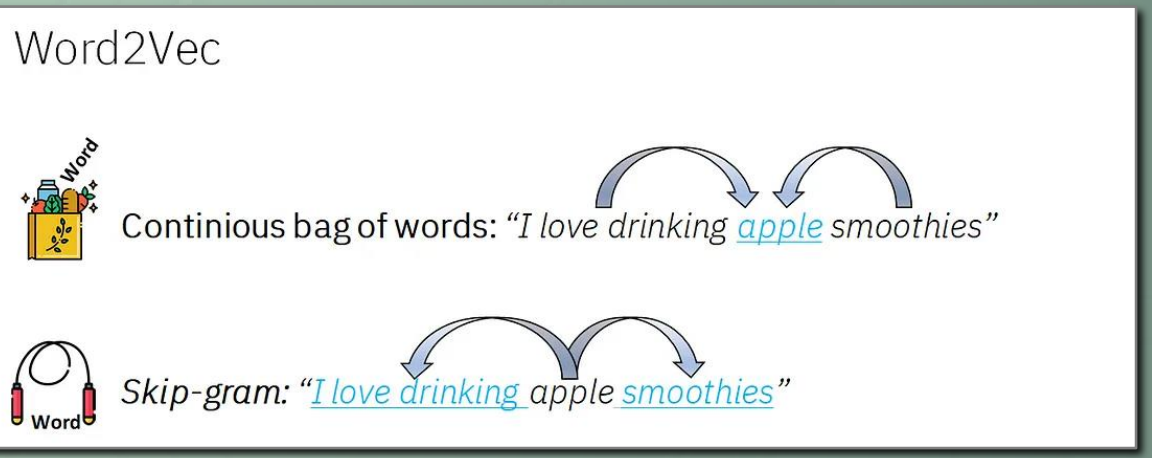
- Captures semantic meaning
- Words with similar meanings are closer in the vector space

Two Training Methods

- Continuous Bag of Words (CBOW): Predicts a word given its context
- Skip-Gram: Predicts context words from a given target word

Benefits

- Enables better performance in NLP tasks
- Helps in capturing semantic relationships (e.g., "king" - "man" + "woman" ≈ "queen")

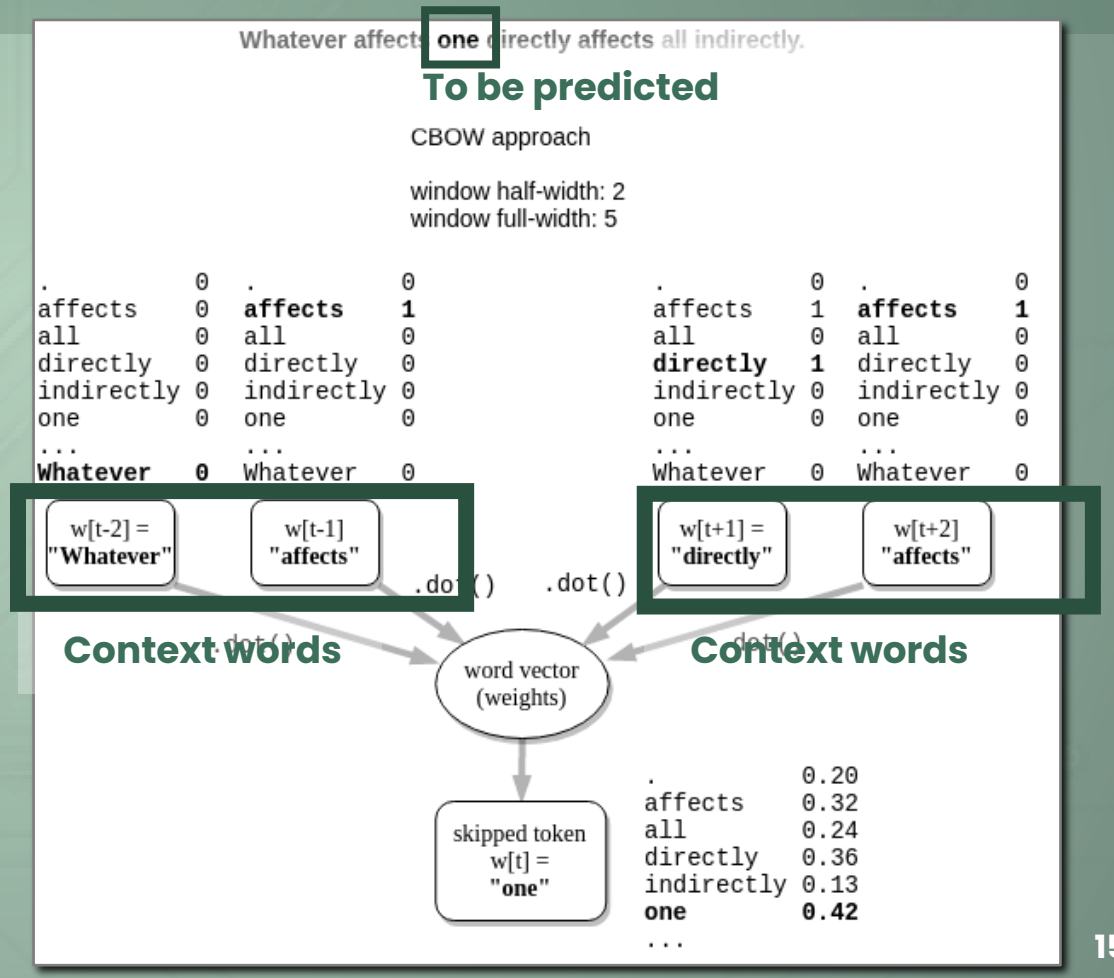


Continuous Bag-of-Words (CBOW)

… is neural network model, predicts target word based on its context words. It's used to generate word embeddings, i.e. vector representations of words that capture semantic relationships.

Key Features

- Predicts target word given its context words
- Input: Average of vectors of context words
- Output: Probability distribution over all words in vocabulary
- Performs well on frequent words
- Useful for tasks prioritizing overall context over word order
- Part of the word2vec family of algorithms



Contextualized Word Embedding

Unlike traditional embeddings, contextualized embeddings generate word representations based on their specific context within a sentence, allowing for dynamic meanings.

Key Features

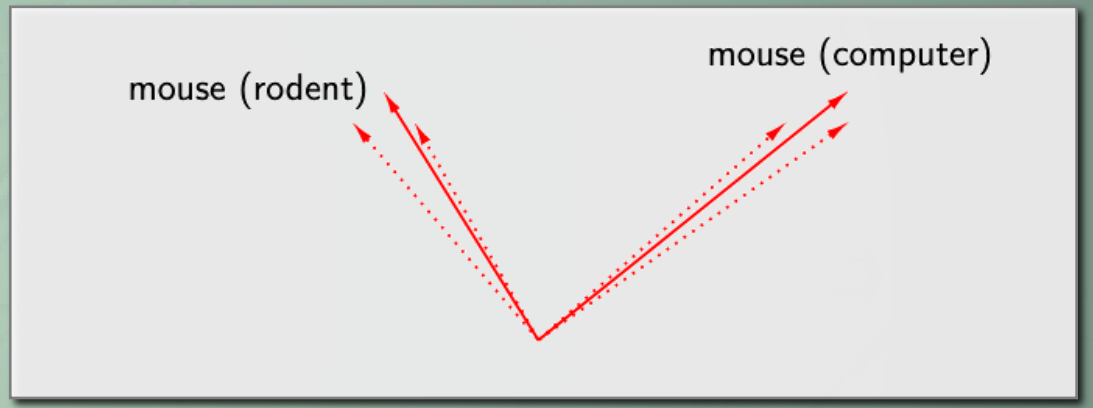
- Words can have different vectors in different contexts.
- Captures polysemy: a word's ability to have multiple meanings.

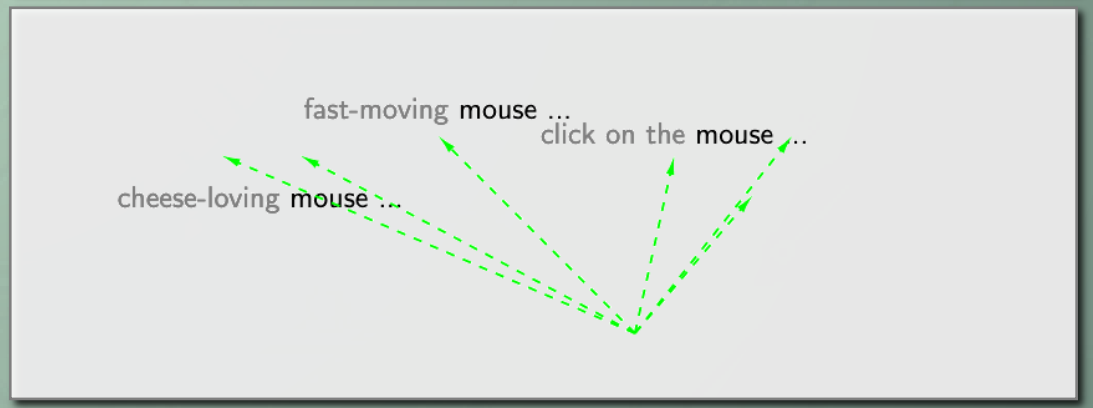
Popular Models

- ELMo (Embeddings from Language Models)
- BERT (Bidirectional Encoder Representations from Transformers)

Benefits

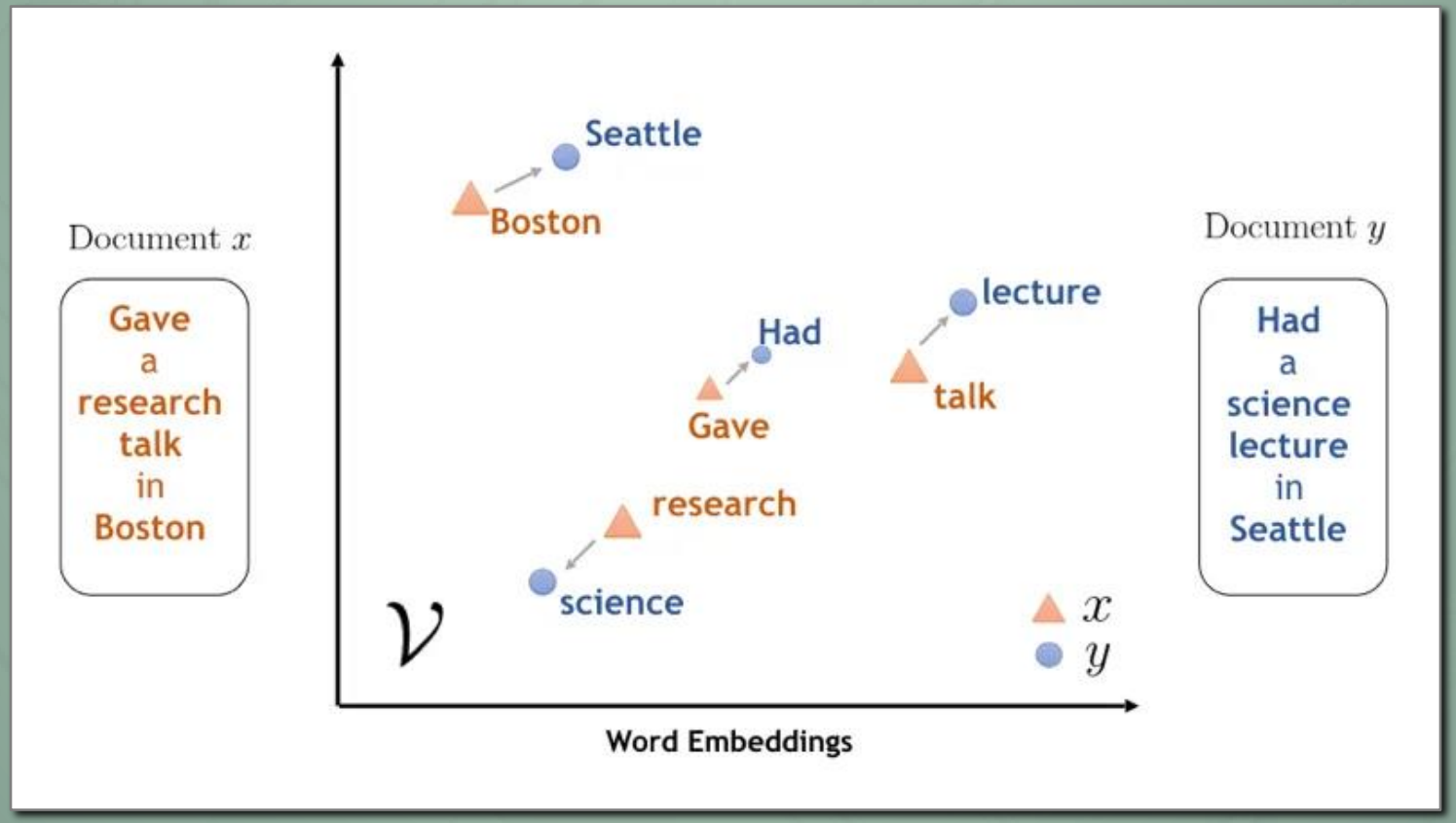
- Enhanced understanding of word nuances and meanings.
- Improved performance on downstream NLP tasks.





Contextualized Word Embedding

... allows you to calculate contextual distances of sentences, paragraphs, etc.



Markov Chain Language Models

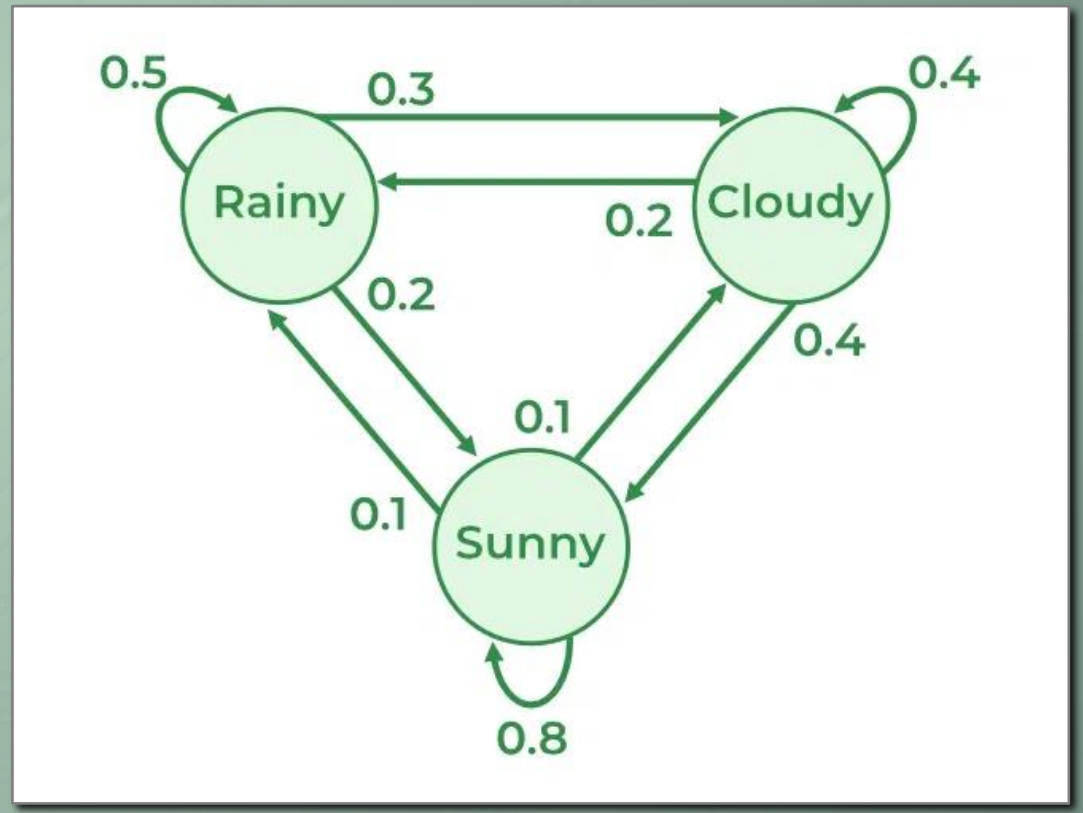
… are statistical models that predict next word in sentence based on current state (previous word or n words), assuming future states depend only on current state (Markov Property).

Key concepts

- States: Words or sequences of words.
- Transitions: Probabilities of moving from one state to another.
- Order: Defines how many previous states (words) are considered (e.g., first-order, second-order).

Example: First-Order Markov Model

- Sentence: "I love natural language processing."
- Transitions: P("love" | "I"), P("natural" | "love"), P("language" | "natural"), etc.

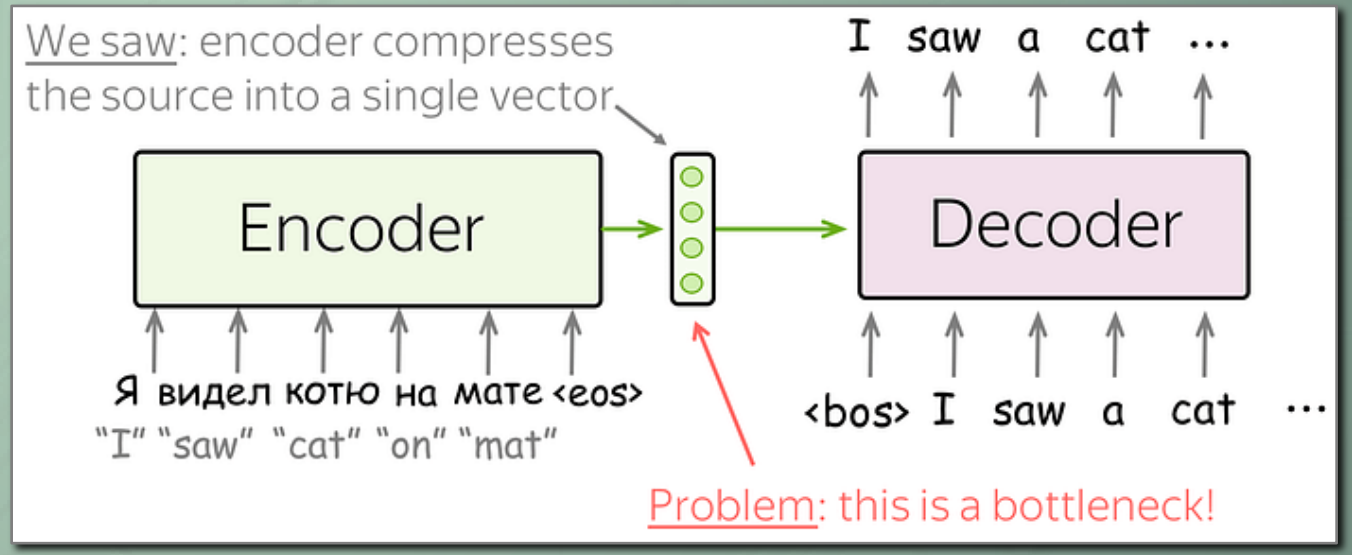


https://www.geeksforgeeks.org/markov-chains-in-nlp/

Recurrent neural nets (RNN)

… are neural networks designed for sequential data, using loops to persist information but struggling with long-term dependencies due to the vanishing gradient problem.

- Designed for sequential data processing.
- **Memory of previous inputs through loops:**
  Each neuron is connected to itself
- Effective when order and context are important
  (e.g., language modeling, time-series prediction).
- Relatively easy to train and low computational intensity
- **Suffers from vanishing gradient problem:**
  difficult to learn long-term dependencies.
- **Applications:**
  Simple text generation and time-series forecasting.

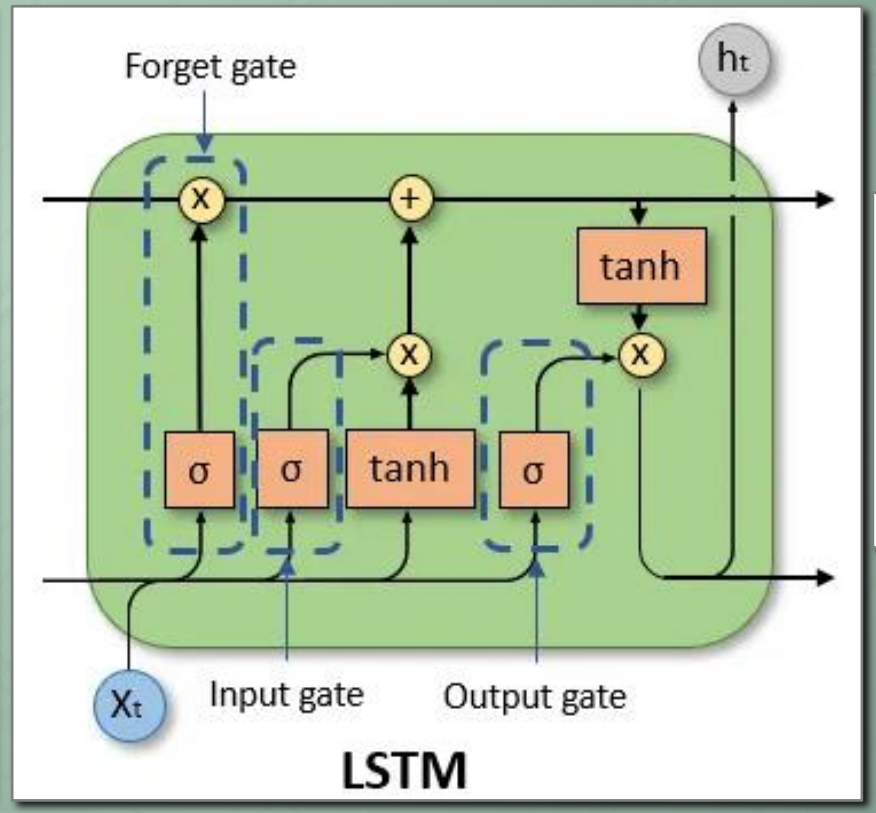


https://medium.com/@mahdi.hammi/understanding-recurrent-neural-networks-rnns-and-long-short-term-memory-lstm-1915c130e141

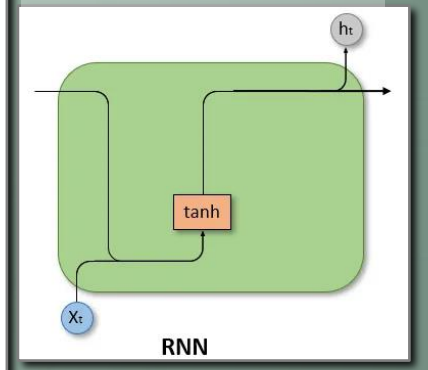
Long Short-Term Memory (LSTM)

… are advanced type of RNN that include mechanisms called gates to effectively manage and retain long-term dependencies, overcoming limitations of standard RNNs.

- Designed to overcome vanishing gradient problem
- Utilizes unique architecture with cell states and three types of gates: forget gate, input gate, and output gate.
- Capable of learning and retaining long-term dependencies.
- More computationally intensive to train than standard RNNs.
- Were widely used before Transformers in applications such as
  - text generation
  - speech recognition
  - machine translation
  - sentiment analysis,
  - time-series forecasting



Comparison



https://medium.com/@mahdi.hammi/understanding-recurrent-neural-networks-rnns-and-long-short-term-memory-lstm-1915c130e141

RNNs and LSTMs – Comparison and Limitation

LSTM handles long sequences more effectively. Nevertheless context length remains limited

Feature	RNN	LSTM
Definition	Neural network for sequential data	Advanced RNN variant for long-term dependencies
Architecture	Simple structure with loops	Complex structure with cell states and gates
Memory Capability	Short-term memory, struggles with long sequences	Long-term memory, handles long sequences more effectively
Vanishing Gradient	Prone to vanishing gradient problem	Designed to mitigate vanishing gradient problem
Components	Neurons with recurrent connections	Cell state, forget gate, input gate, output gate
Training Complexity	Easier to train, less computationally intensive	More complex to train, computationally intensive
Applications	Basic sequence modeling, time-series prediction	Text generation, speech recognition, time-series forecasting

https://medium.com/@mahdi.hammi/understanding-recurrent-neural-networks-rnns-and-long-short-term-memory-lstm-1915c130e141

Comparison

… of NLP approaches

Model	N-grams	Vector embedding	Markov Chain	LSTM
Examples	▪ Predicting next word ▪ Spam classification ▪ Autocomplete suggestions	▪ Finding similar words ▪ Semantic document classification ▪ Improving search relevance	▪ Generating style-specific text ▪ Predictive text input ▪ Simple chatbots	▪ Machine translation ▪ Sentiment analysis ▪ Complex text generation
When to Use	▪ Simple, fast text prediction ▪ Limited computational resources ▪ Tasks not requiring deep semantics	▪ Capturing word semantics ▪ Tasks needing dense word vectors ▪ Context-crucial applications	▪ Modeling sequential data ▪ Generating word sequences ▪ Capturing text patterns	▪ Long-term dependencies ▪ Complex sequence tasks ▪ Large-scale language modeling
Key Strength	Simplicity and efficiency	Semantic understanding	Simple sequence generation	Long-term memory and context
Limitation	Very limited context of few words	No sequential information	Limited to local dependencies	Context window of few sentences
Model Type	Statistical	Neural network (shallow)	Probabilistic	Neural network (deep)
Context Handling	Fixed, short context	Bag-of-words context	Fixed-order context	Variable, potentially long context

Attention mechanism before “Transformers (2017)”

Attention mechanisms were a critical step in advancing natural language processing and understanding before the advent of Transformers

Bahdanau Attention
Neural Machine Translation by Jointly Learning to Align and Translate, Bahdanau, et al., 2014

- Introduced dynamic focus on different parts of the input sequence using additive scoring for neural machine translation.

Luong Attention
Effective Approaches to Attention-based Neural Machine Translation, Luong et al., 2015

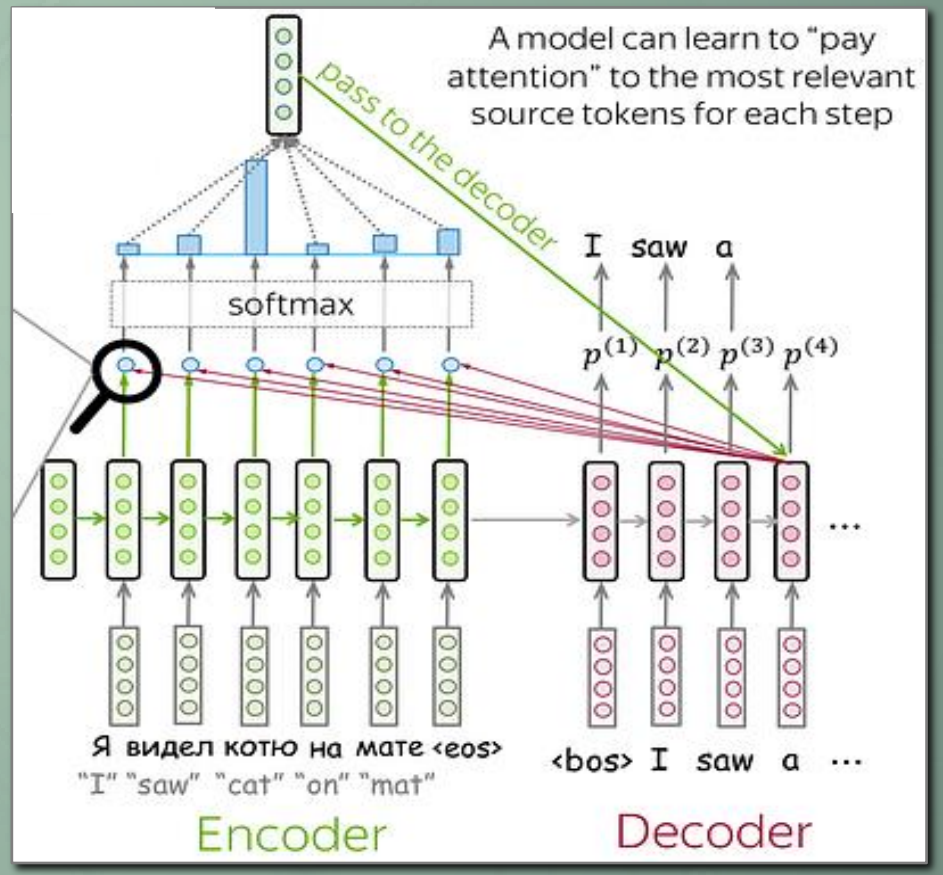
- Improved computational efficiency with dot-product and general scoring functions for alignment in sequence-to-sequence models.

Self-Attention
Long Short-Term Memory-Networks for Machine Reading, Cheng et al., 2016

- Enabled parallel computation by allowing each word to attend to all other words, forming the basis of the Transformer architecture.

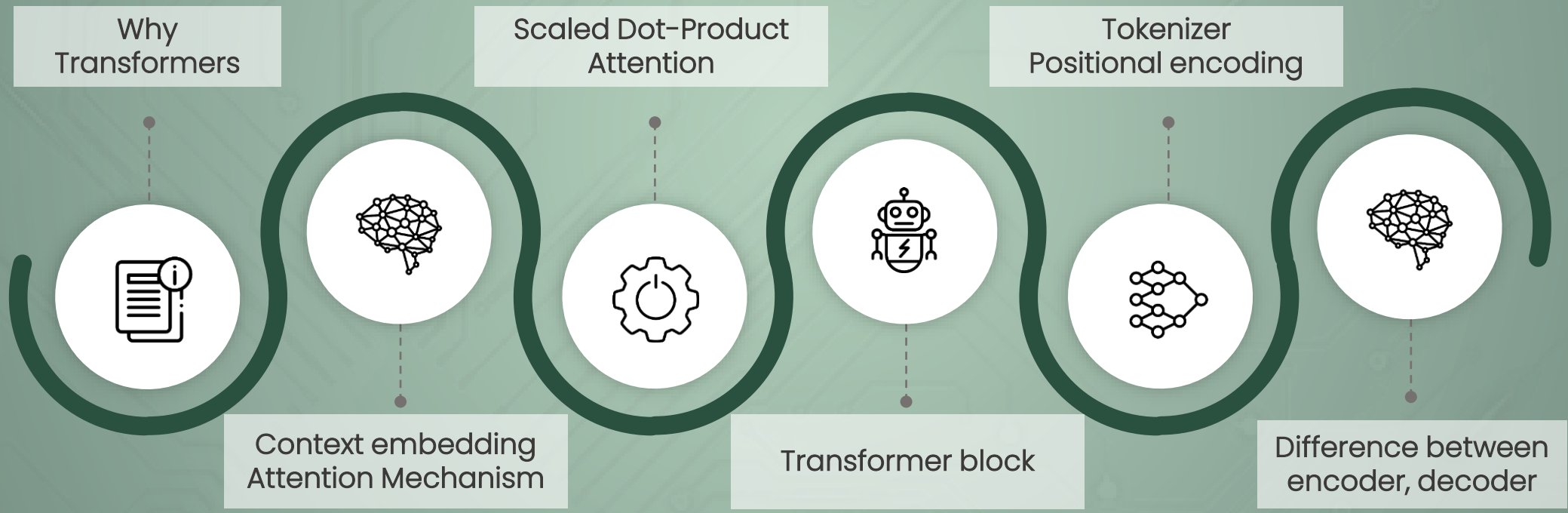
Hierarchical Attention
Hierarchical Attention Networks for Document Classification, Yanh et al., 2016

- Applied attention at multiple levels (word and sentence) to better capture context in hierarchical data structures.



Transformers

… are type of neural network architecture that uses self-attention mechanisms to process entire input sequences in parallel, enabling efficient handling of long-range dependencies



Why Transformers are significant

Transformers excel at modeling sequential data like natural language.
Nowadays encoders, decoders are used separately.

Encoder:
- Sentiment Analysis
- Text classification

Decoder:
- Conversation (ChatGPT)
- Translation

Comparison with RNNs

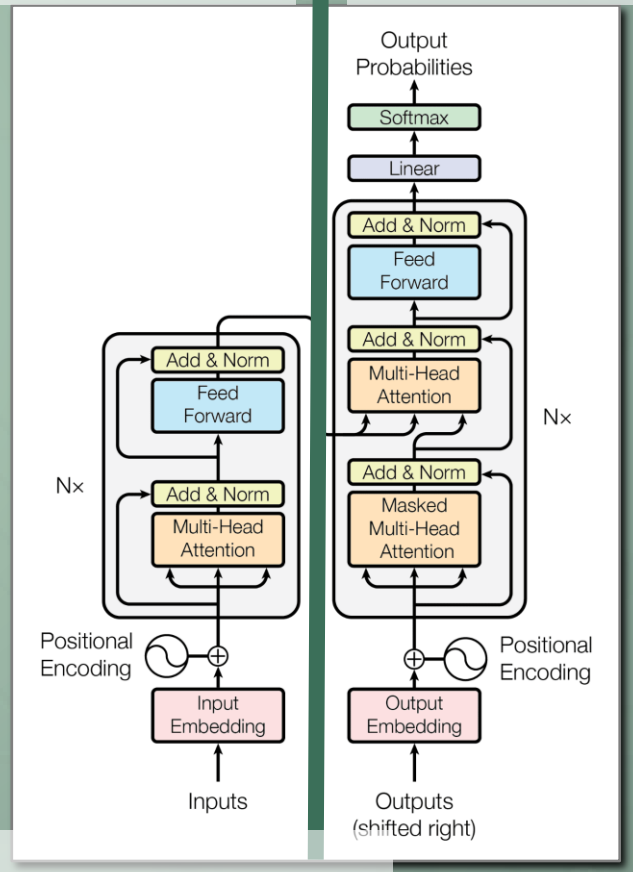
- Parallelizable and efficient on GPUs & TPUs.
- Replaces recurrence with attention for simultaneous computations.
- Outputs computed in parallel, unlike RNNs' series.

Advantages Over RNNs and CNNs

- Captures distant or long-range contexts in data.
- Connects distant positions in sequences for longer connections.
- Uses attention to access entire input at each layer, unlike RNNs, CNNs.

Unique Characteristics

- No assumptions on temporal/spatial relationships.
- Ideal for processing sets of objects (e.g., StarCraft units).

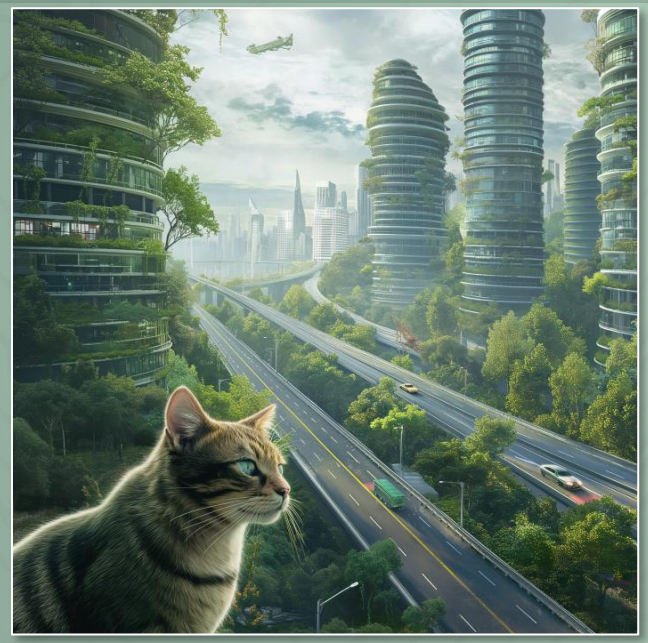


Paper:
Attention is all you need (2017)

Context embedding

The cat was afraid to pass the street because it was too broad.

Result



Does “it” refer to cat or to street?

Based on training humongous amount of text data the transformer learns that in this context “it” refers to “street”

→ “it” has maximal attention to “street”

Attention Mechanism

A technique in deep learning models, especially in sequence-to-sequence tasks, that allows the model to focus on specific parts of the input when producing an output.

Key Features

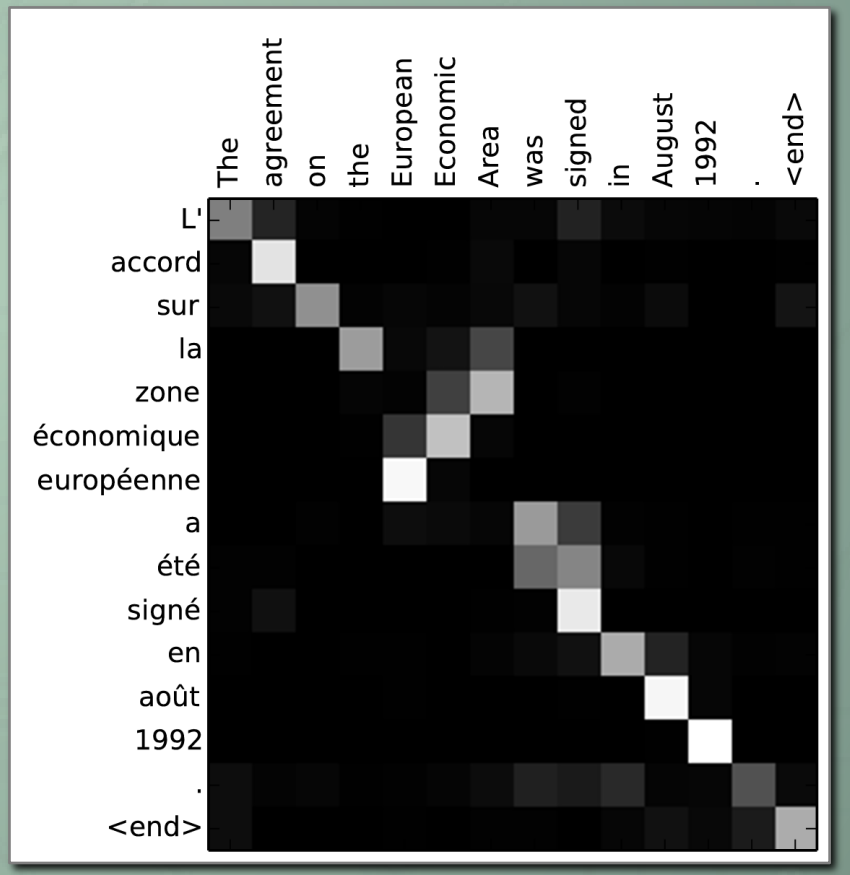
- Dynamically weighs input elements.
- Enhances the capturing of long-range dependencies in sequences.

Usage

- Machine Translation: Helps in aligning words in source and target languages.
- Text Summarization: Prioritizes crucial parts of the content.

Benefits

- Improves model's ability to remember long sequences.
- Enhances accuracy in tasks like translation and summarization.



Attention Mechanism

What is the most probable missing word? It is determined by putting the other words into context to each other.

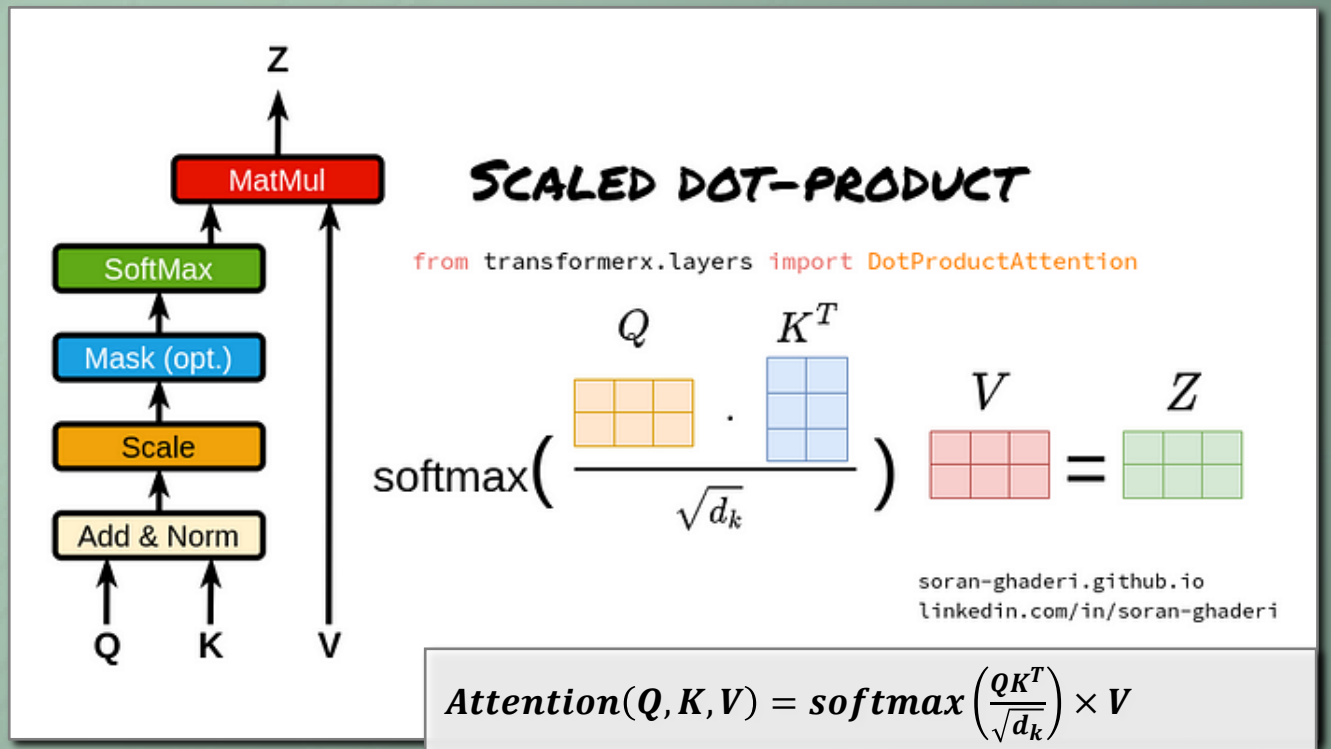
Scaled Dot-Product Attention

…is a mechanism used in attention models that calculates attention scores based on dot product of query and key, scaled down by square root of their dimensionality.

- Queries: derived from input data, represent focus of model
- Keys: also derived from input data, interpretation: "labels" for the input data
- Values: are weighted based on compatibility of query and corresponding key

How it works

- For each query, attention mechanism computes score with each key in input. Score represents how well query aligns with particular key.
- Scores are used to create a weighted combination of the values.
- If key aligns well with query, value associated with that key gets larger weight in final output.

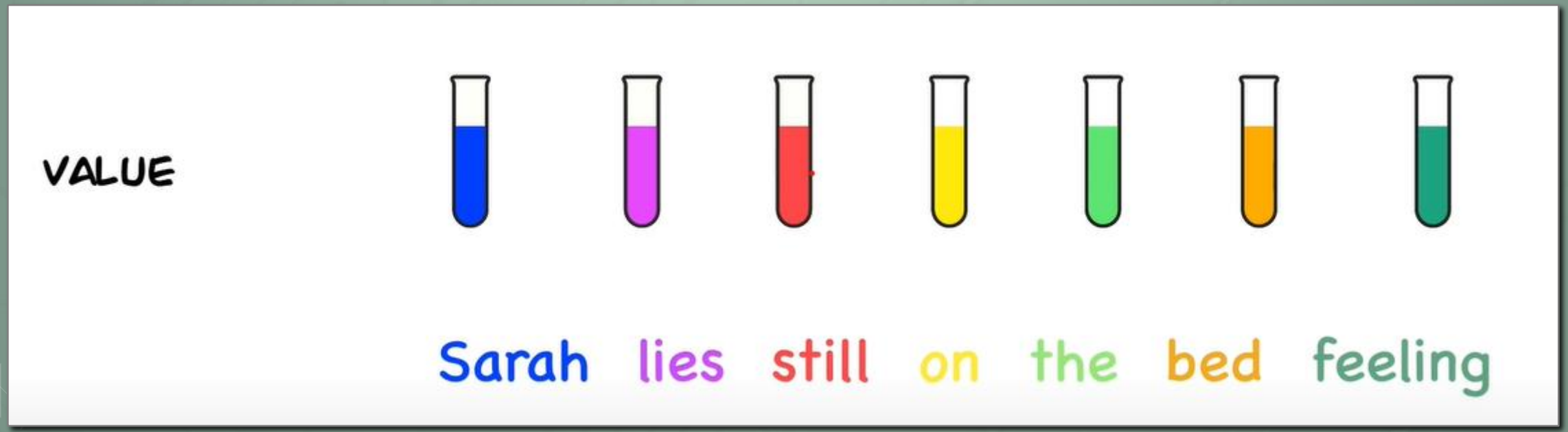


$$\textbf{Attention}(Q, K, V) = \textbf{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \times V$$

- Q, K, V: Query, Key, and Value vectors respectively
- d_k: dimensionality of query/key vectors

Attention Mechanism: Value (V) vectors for all words

These contain actual information of each word that will be aggregated to form output



Position is not important for coding, if word occurs several times it receives the same code

[0.1, 0.2] [0.3] [0.4] [0.5] [0.6] [0.7] [0.8, 0.9]

Attention Mechanism: Key (K) vectors for all words

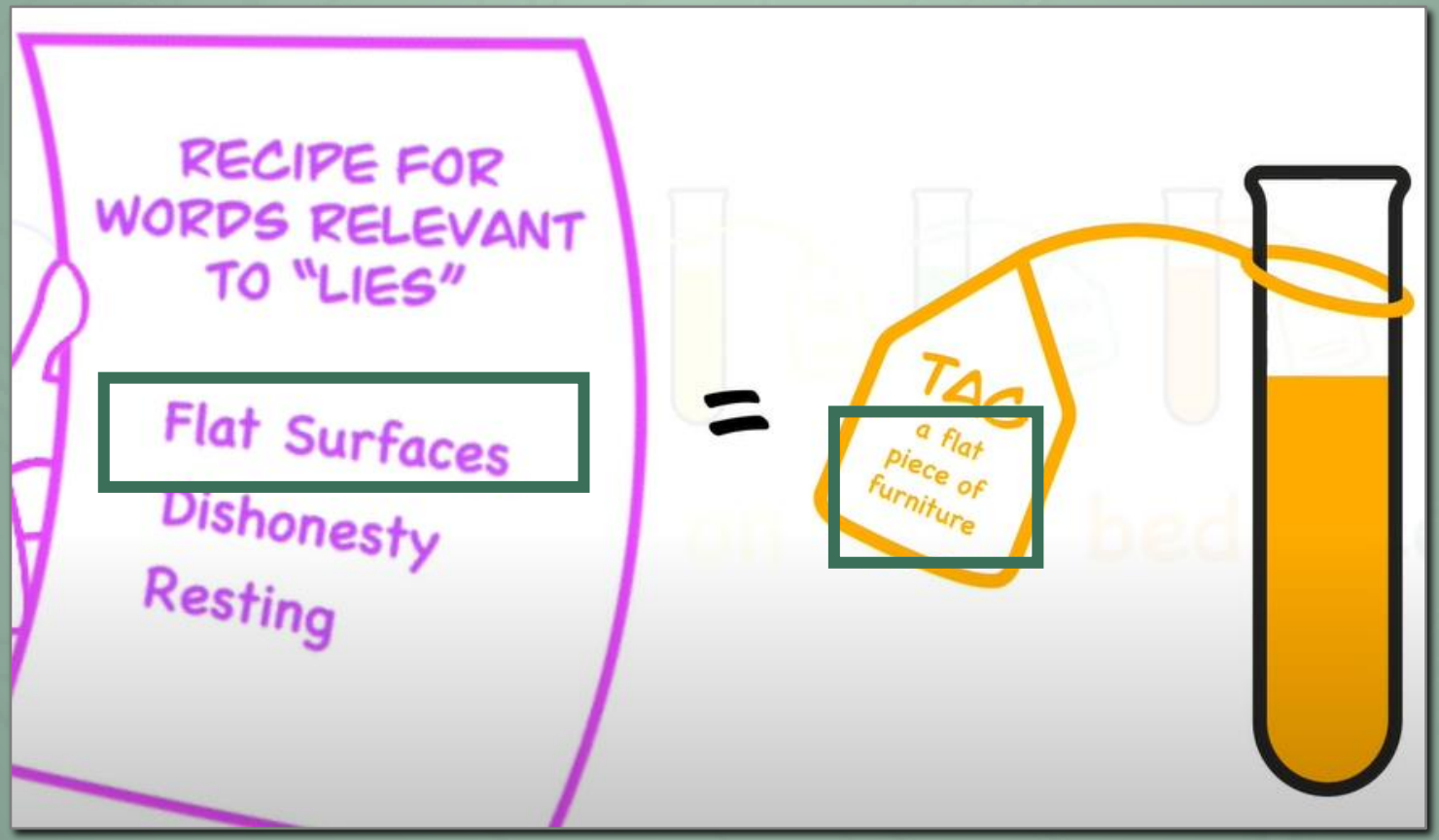
These represent how each word in the sentence can be queried

Attention Mechanism: Query (Q) vectors

… are used to represent the word in question, seeking information from other words.

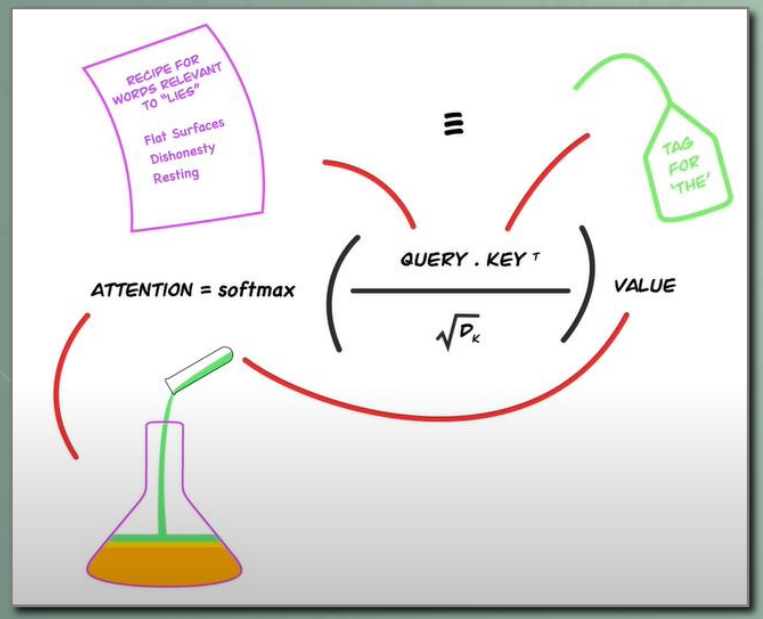
Attention Mechanism: Connecting query and key

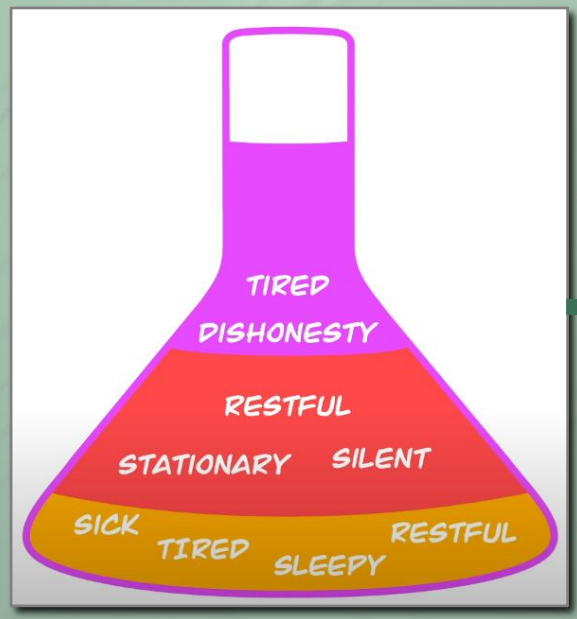
… “Flat Surfaces” from Recepe and “a flat piece of furniture” are similar



Attention

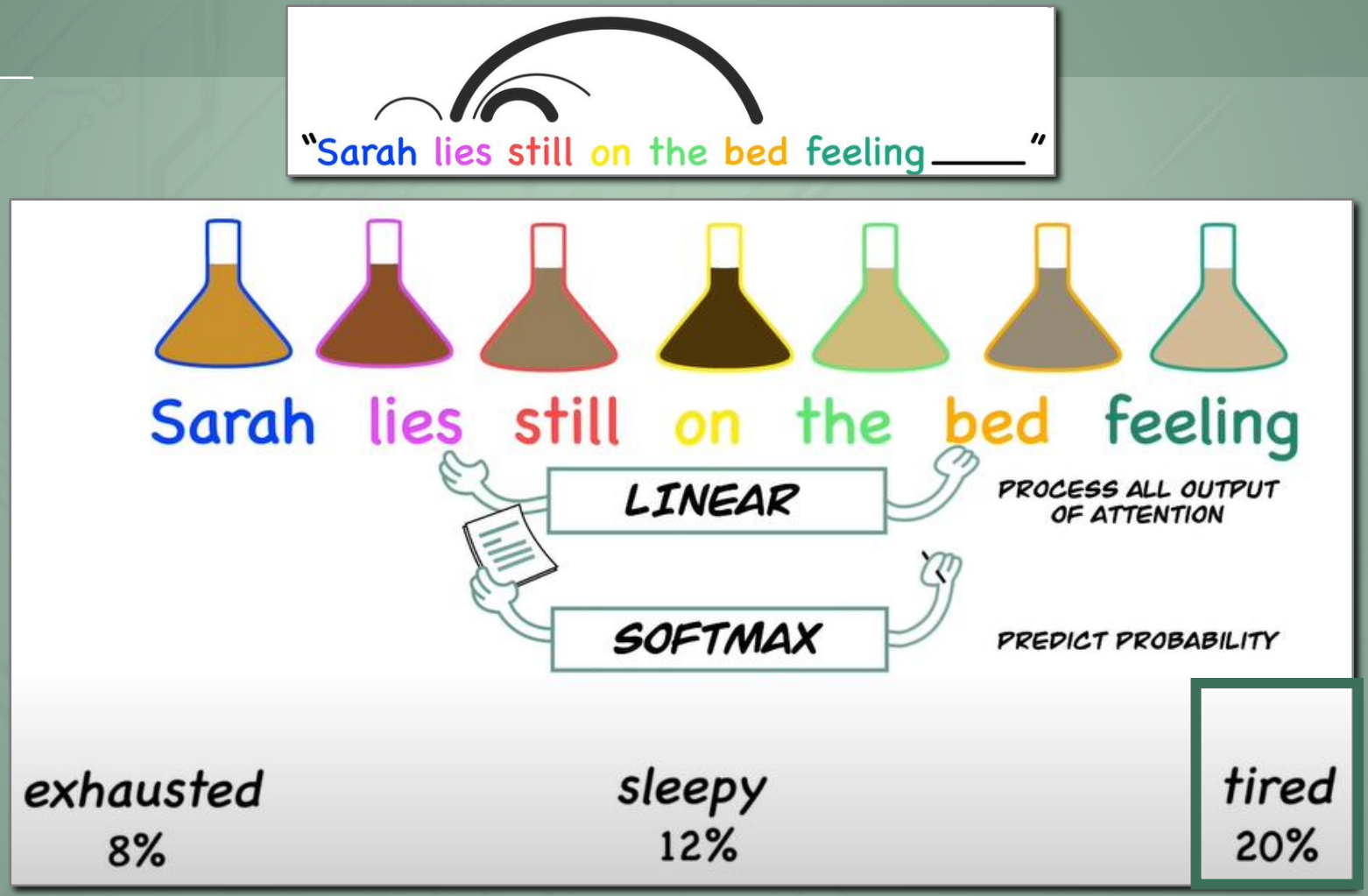
… of the word “lies”





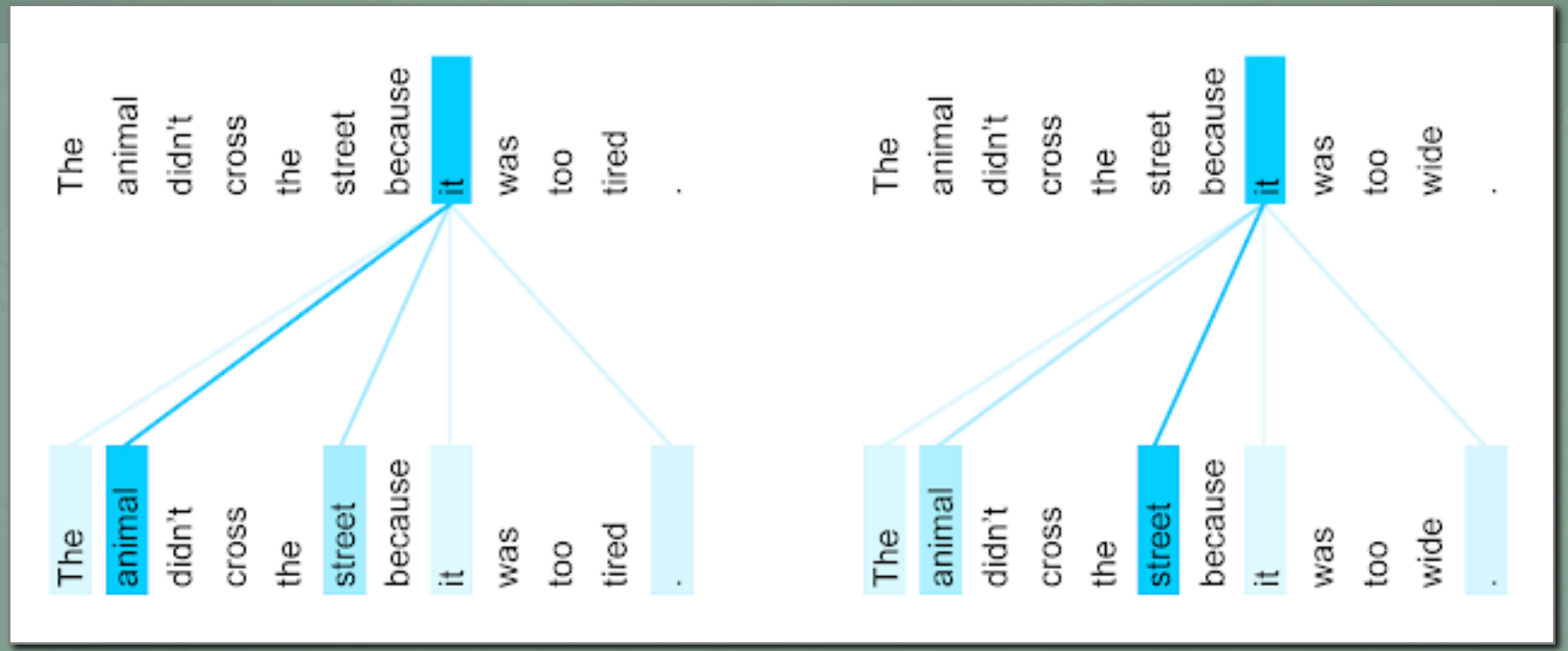
Attention Mechanism

... calculates the word with highest probability through linear transformation & softmax



Self-attention distribution: word “it”

Meaning of the word „it“ depends on context. Any complex task like translation or even Sentiment Analysis of multi-sentence comments needs context awareness.



“it” means animal

“it” means street

Attention Mechanism

What is the most probable missing word?
It is determined by putting the other words into context to each other.

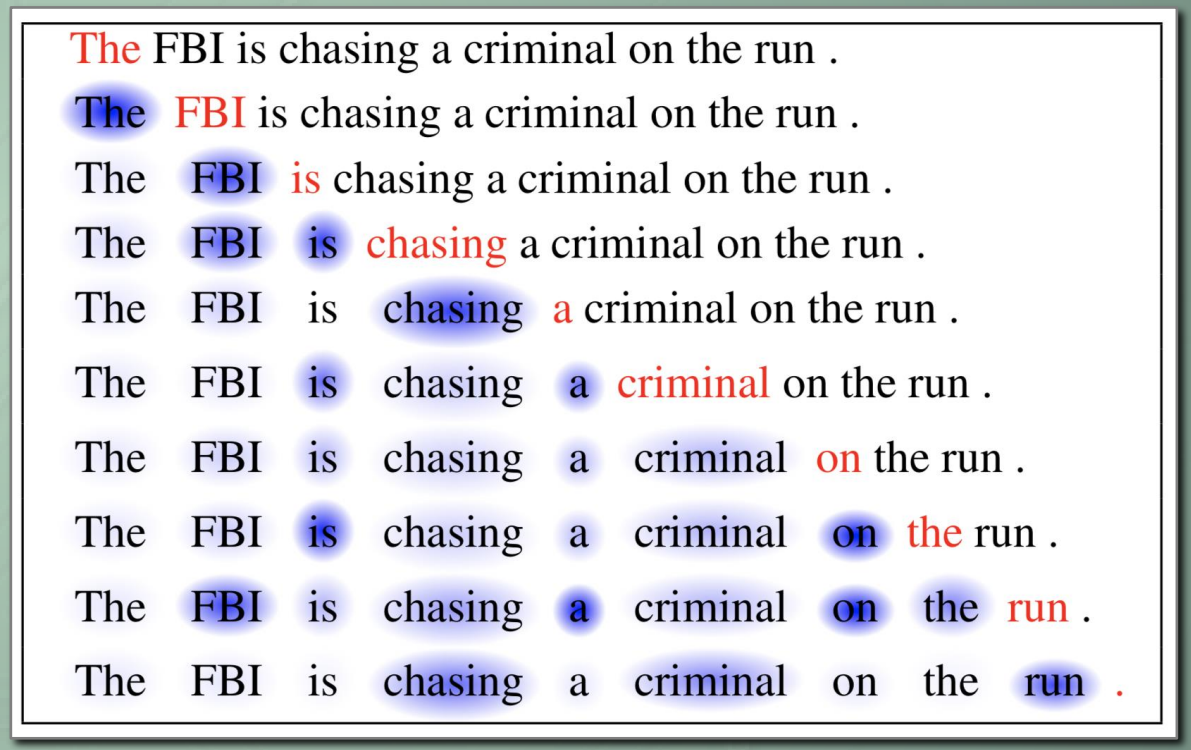
Colors

- Red: represents current word being fixated
- Blue: represents memories.
- Shading: indicates degree of memory activation

Prediction as result of Training

- Sentence was split as triangle, i.e. each word was represented by one row with all previous words
  - Red words: Predicted target values
  - All previous words: input values

→ Self-supervised learning



Long Short-Term Memory-Networks for Machine Reading (2016)

https://slds-lmu.github.io/seminar_nlp_ss20/attention-and-self-attention-for-nlp.html

Multi-head Attention

… is key component of transformer models, allowing the model to focus on different parts of input sequence simultaneously, by applying multiple attention operations in parallel.

Multiple Attention Heads

- Uses sets of Q, K, V weight matrices
- Each set forms an 'attention head', attention scores are calculated for each head

Parallel Processing

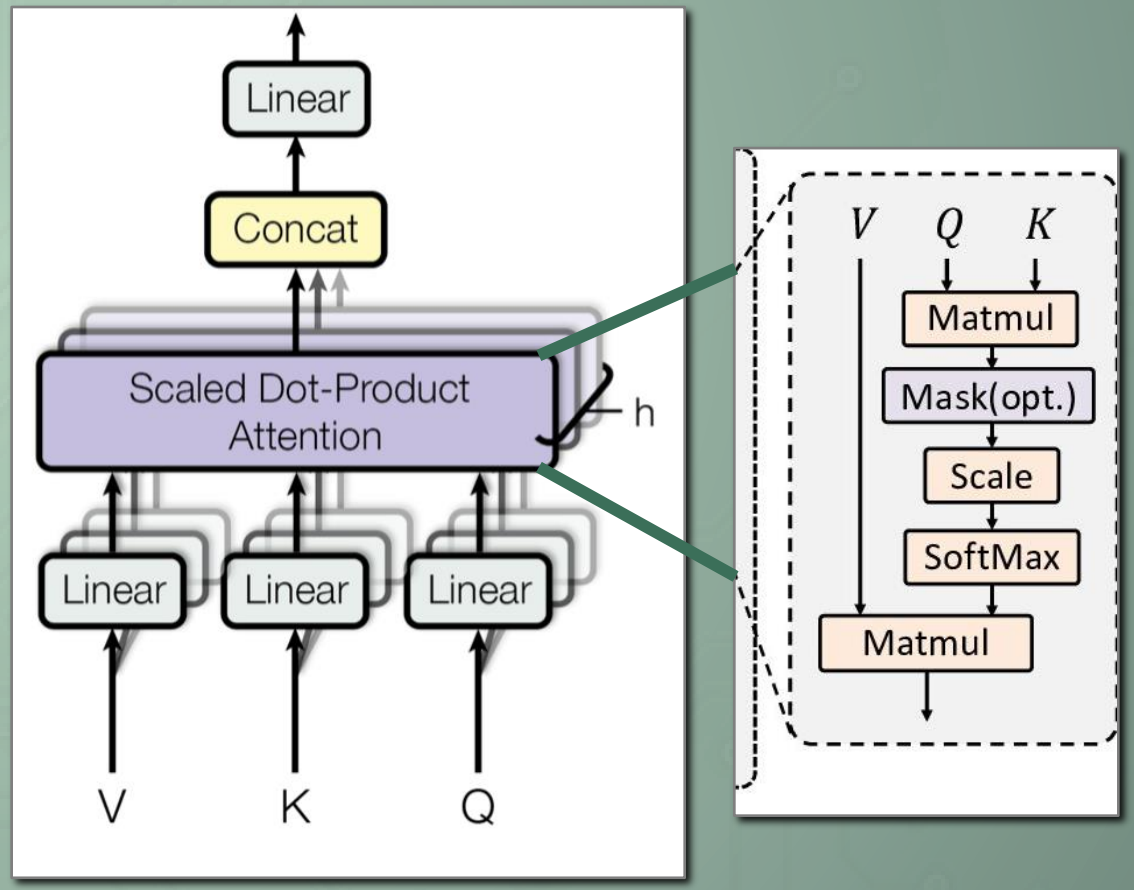
- All heads operate simultaneously
- Captures various data relationships at once

Linear Projections

- Input projected into subspaces for each head
- Allows focus on different input aspects

Concatenation and Linear Transformation

- Outputs from all heads concatenated
- Linearly transformed for final output



Attention is all you need (2017)

Transformer Block including Feed-forward Neural Network

… combines multi-head attention with feed-forward neural network, allowing model to process complex relationships in data

Multi-Head Attention

- Allows model to focus on different parts of the input sequence simultaneously
- Captures various types of relationships within data

Feed-Forward Neural Network

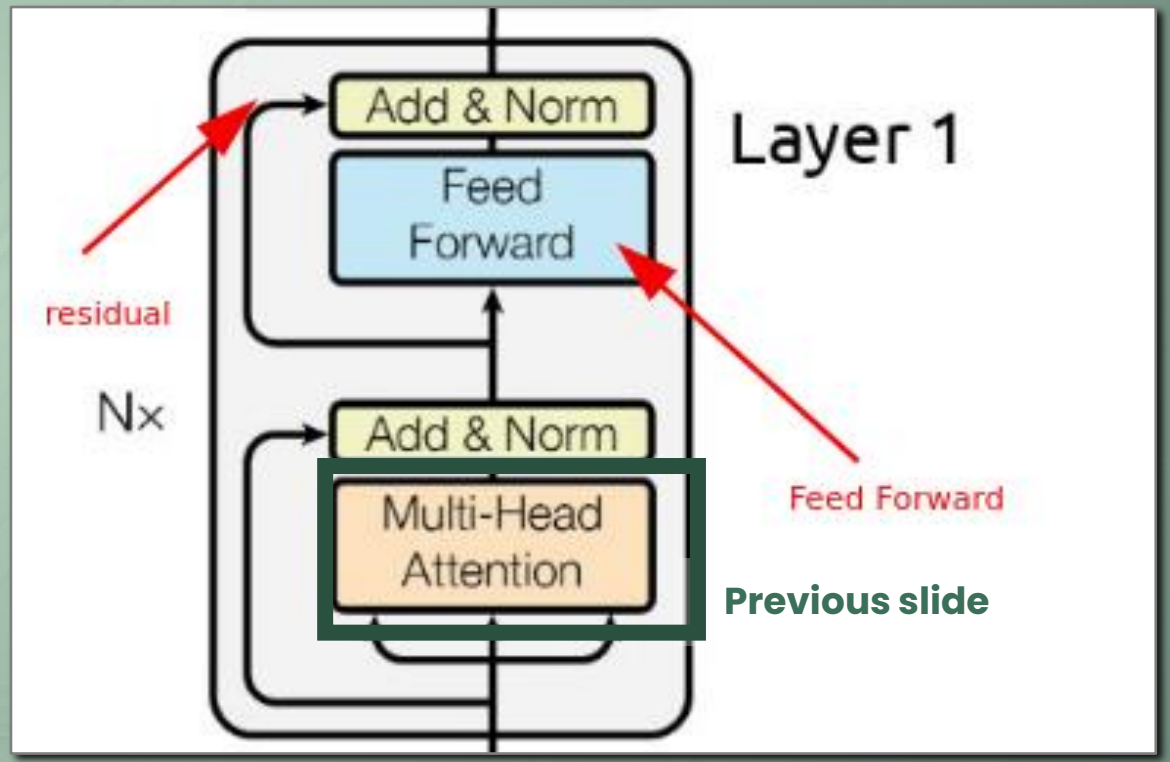
- Fully connected layer that processes output of attention layer
- Adds non-linearity and increases model's capacity

Layer Normalization

- Applied after each sub-layer to normalize activations
- Stabilizes learning process and allows for deeper networks

Residual Connections

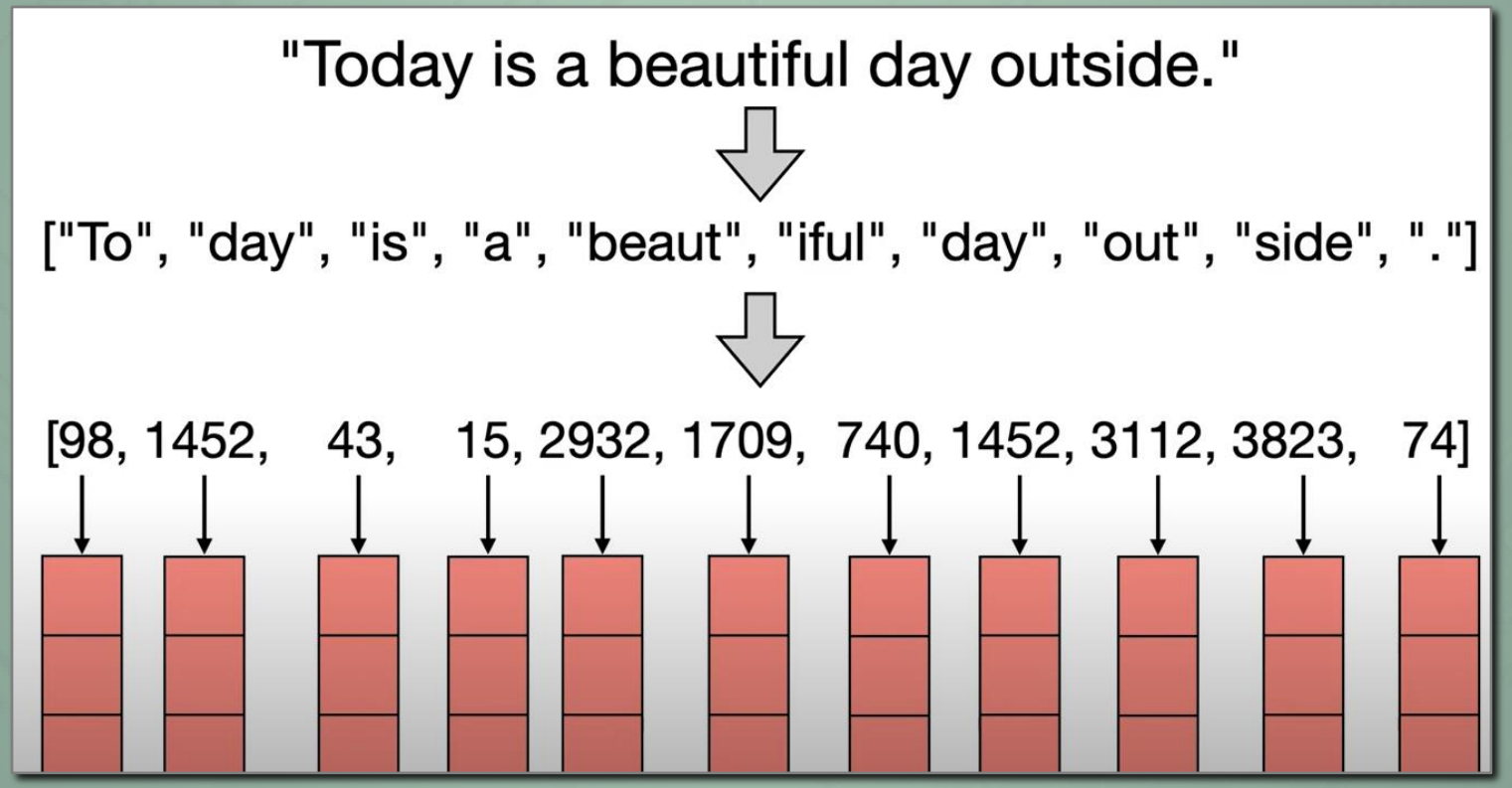
- Skip connections that allow gradients to flow more easily through the network
- Mitigates vanishing gradient problem in deep neural networks



Attention is all you need (2017)

Tokenizer

… converts raw text into sequence of tokens that can be processed by machine learning model by splitting the text into words or subwords, assigning each a unique numerical identifier



https://www.youtube.com/watch?v=hL4ZnAWSyuU

Tokenizer: Byte Pair encoding

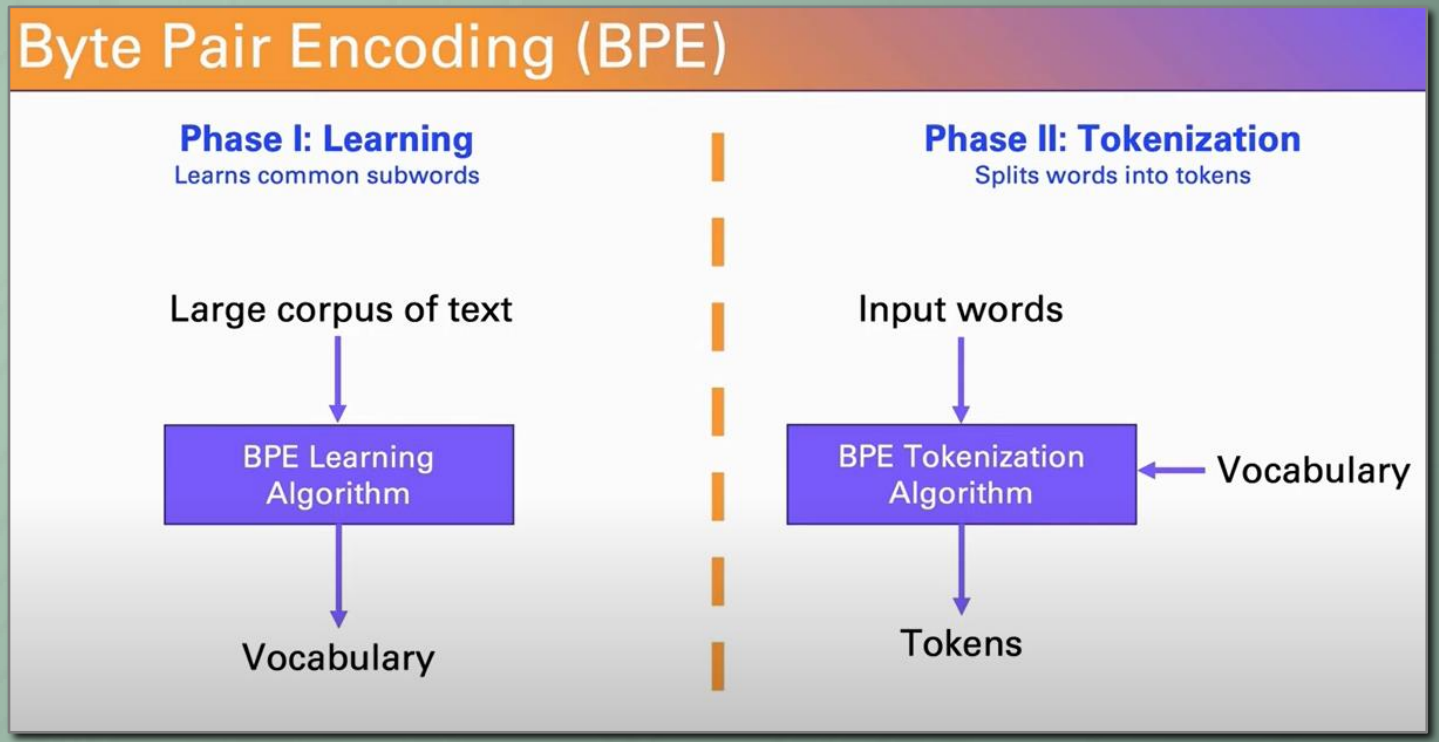
… is subword tokenization algorithm for text compression and processing

Phase I: Learning

1. Initialize with character vocabulary
2. Count token pair frequencies
3. Merge most frequent pair
4. Add new merged token to vocabulary
5. Replace occurrences in corpus
6. Repeat 2-5 until reaching target vocabulary size or iteration limit
7. Finalize vocabulary and encoding rules

Phase II: Tokenization

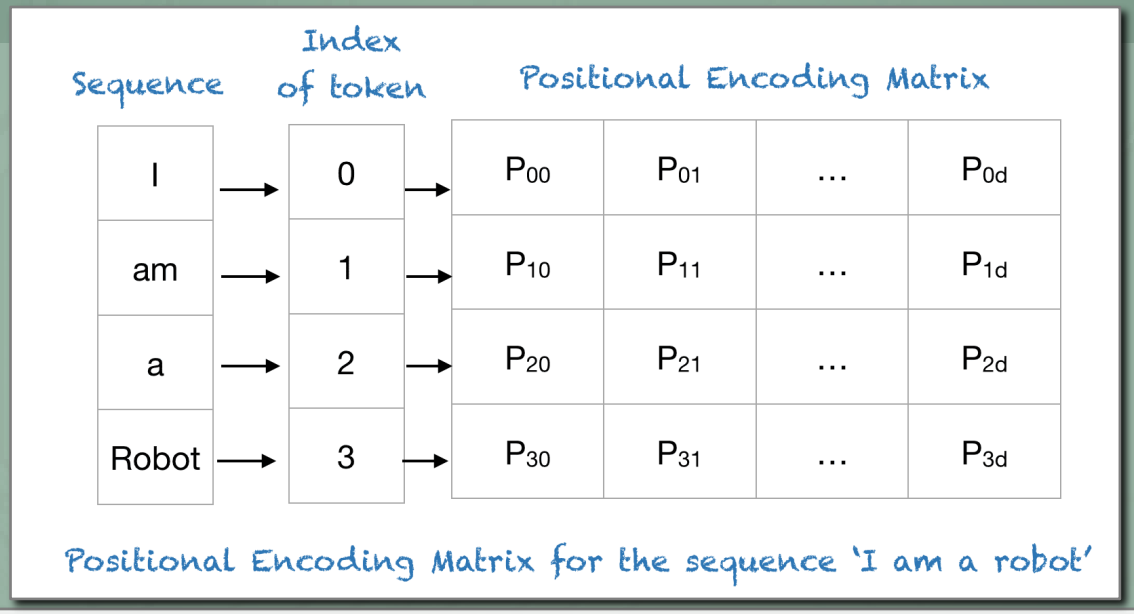
1. Start with input text
2. Apply learned merge rules in order
3. Split text into subword units
4. Replace subwords with corresponding token IDs
5. Output sequence of token IDs

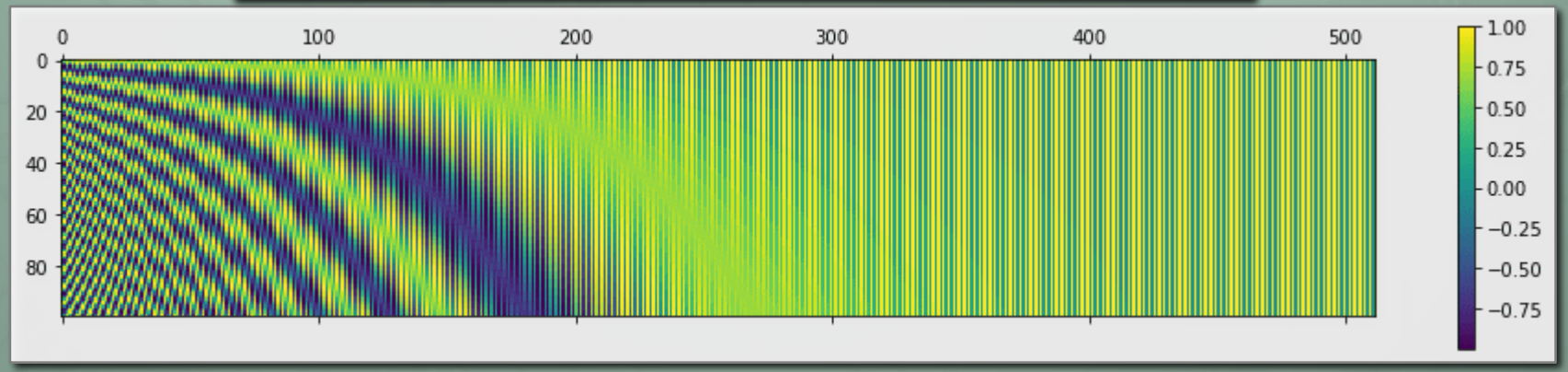


https://www.youtube.com/watch?v=PvZN3-WqAOI
https://huggingface.co/learn/nlp-course/en/chapter6/5

Positional Encoding

… provides sense of position for input tokens. Essential for models like Transformers that do not inherently process sequences in order. Overlapping trigonometric functions are used

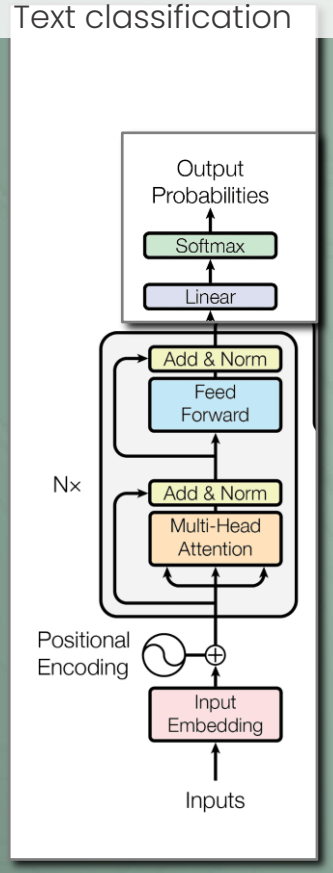


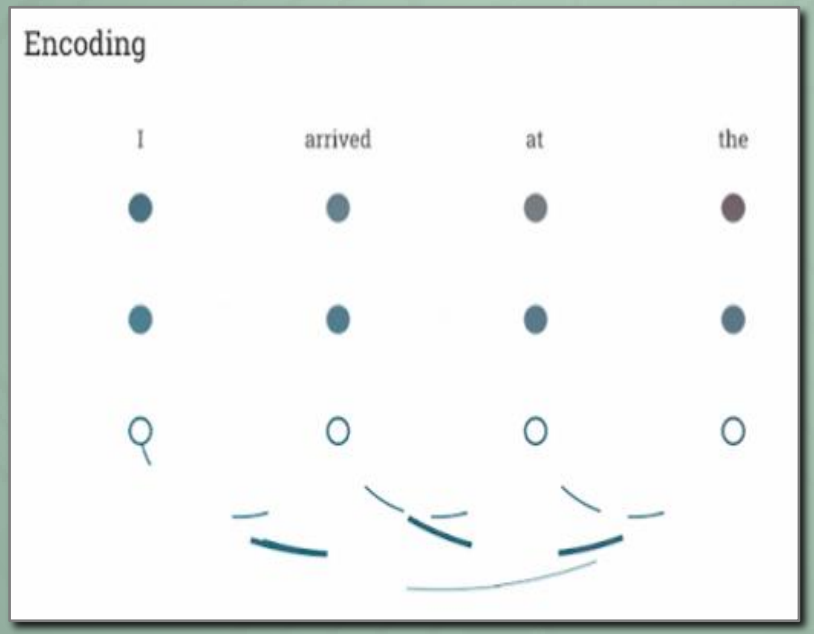


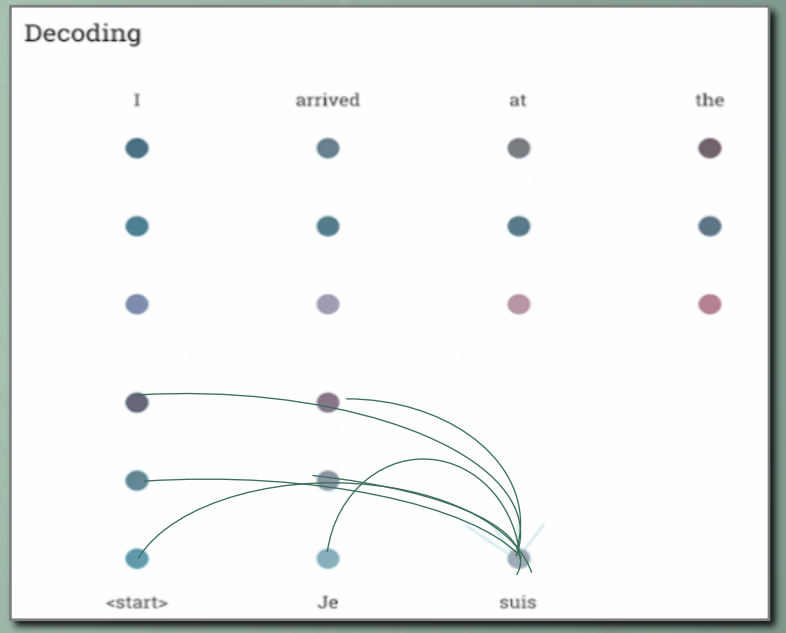
Difference between encoder, decoder

Initially, i.e. in 2014, transformer based translators have used both encoders and decoders. Encoder connectes words (tokens) in both directions, decoder predicts next word.

Encoder:
- Sentiment Analysis
- Text classification

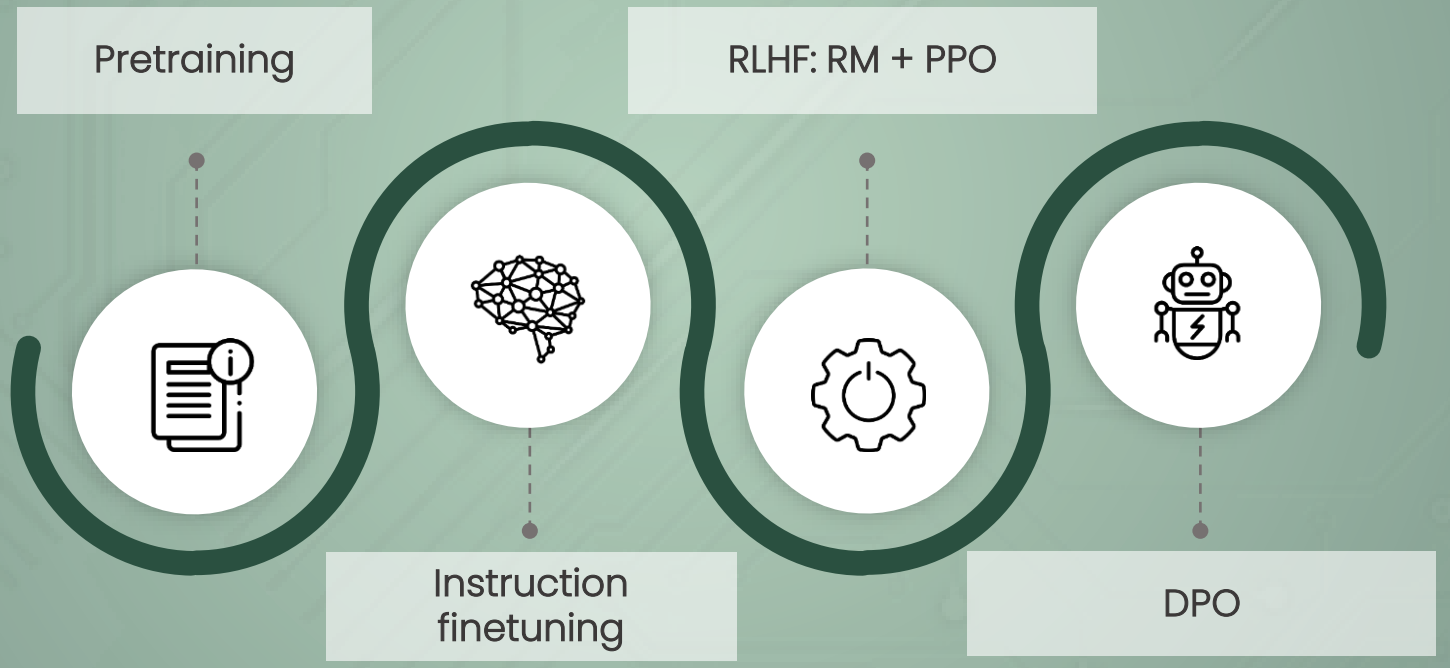






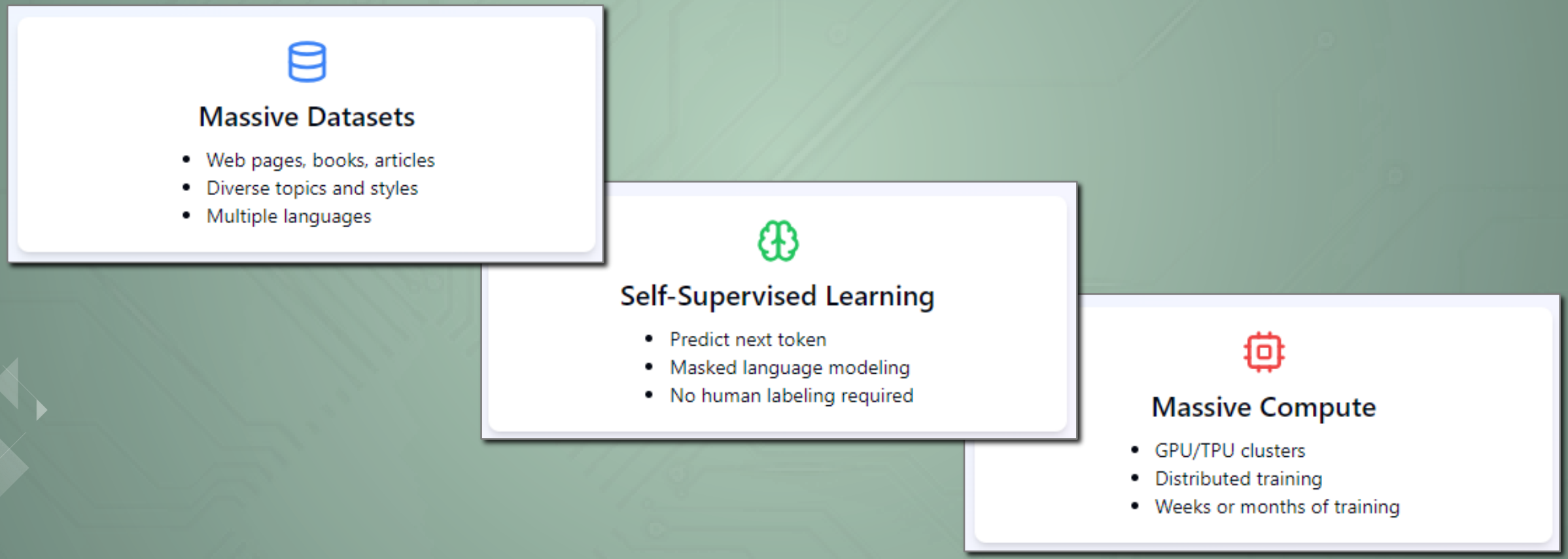
Training Large Language Models

… involves feeding massive amounts of text data into neural networks with billions of parameters to enable model to generate human-like text across wide range of tasks and domains



Pretraining of Large Language Models

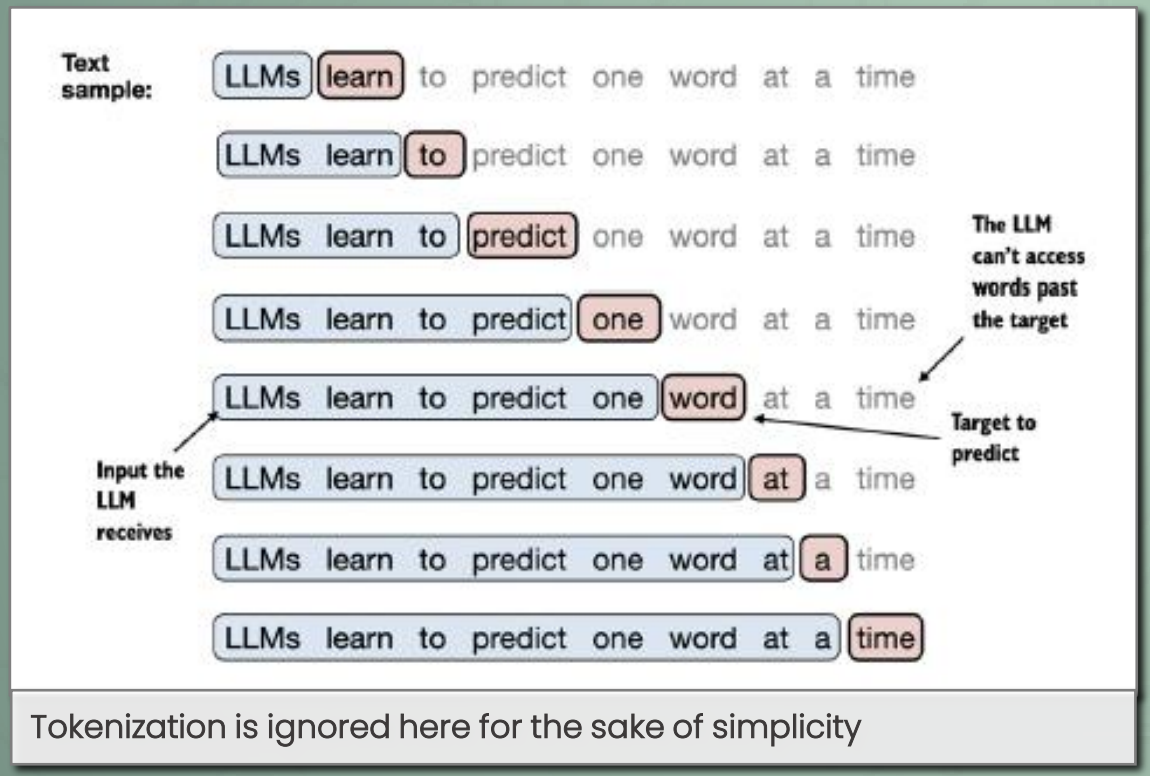
… involves exposing model to vast amounts of diverse text data, allowing it to learn general language patterns and representations through predicting masked words or next sentences



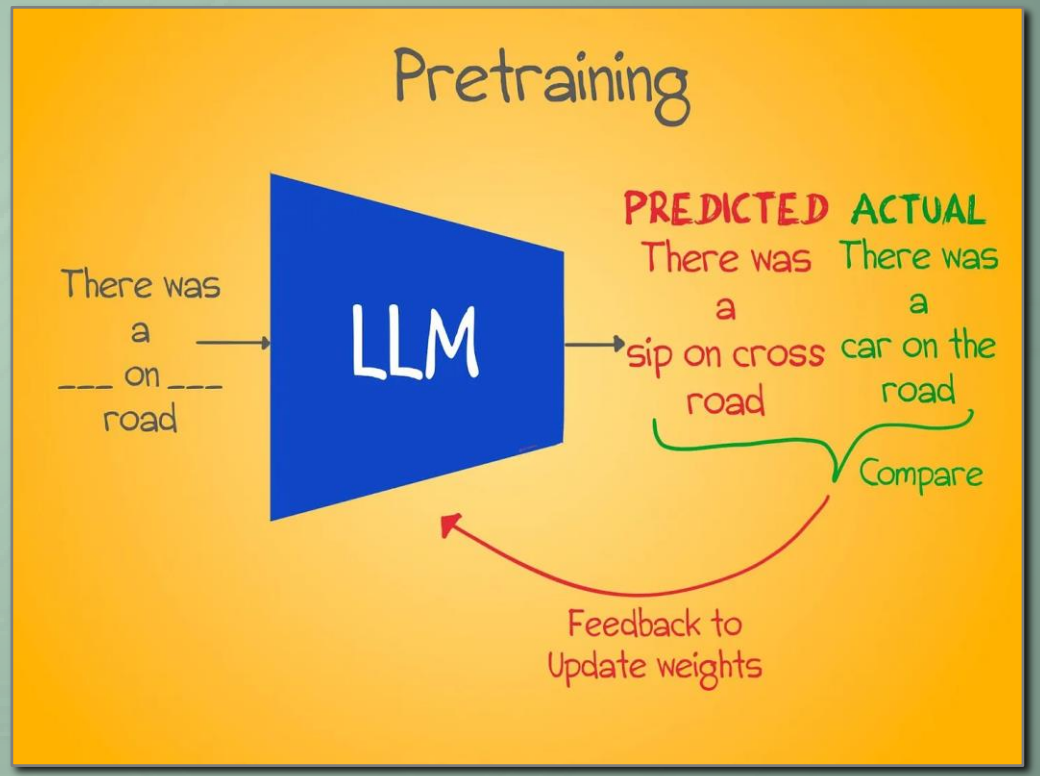
https://newsletter.artofsaience.com/p/a-simple-llm-primer-and-book-launch

Pretraining of Large Language Models

Each sentence of words is predicted by previous (masked attention) words. Feedback is given whether prediction was correct or not. Based on this weights/parameters are updated.



Tokenization is ignored here for the sake of simplicity

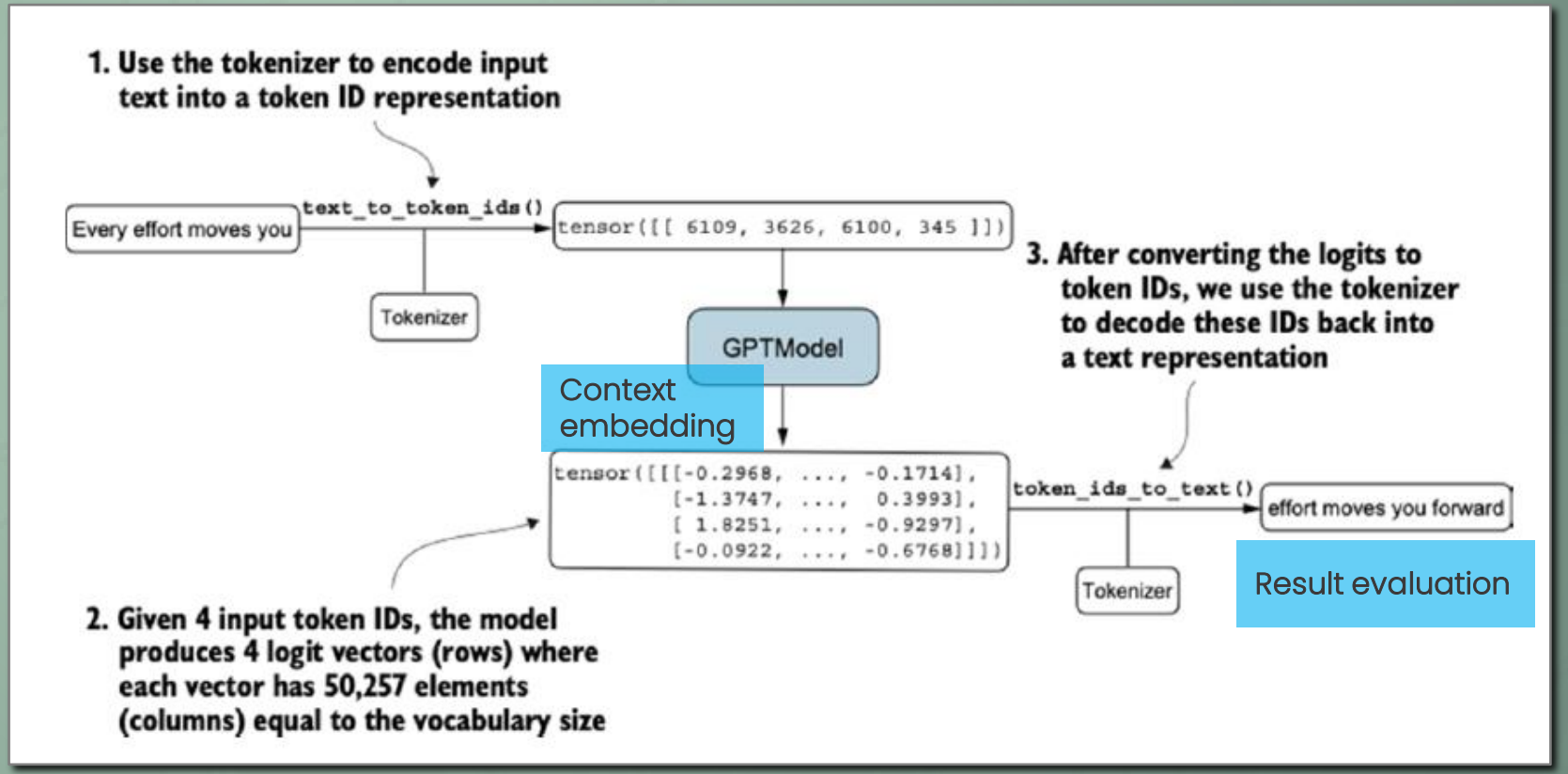


Training process: Pretraining

"Every effort moves you " … "forward"

Pretraining

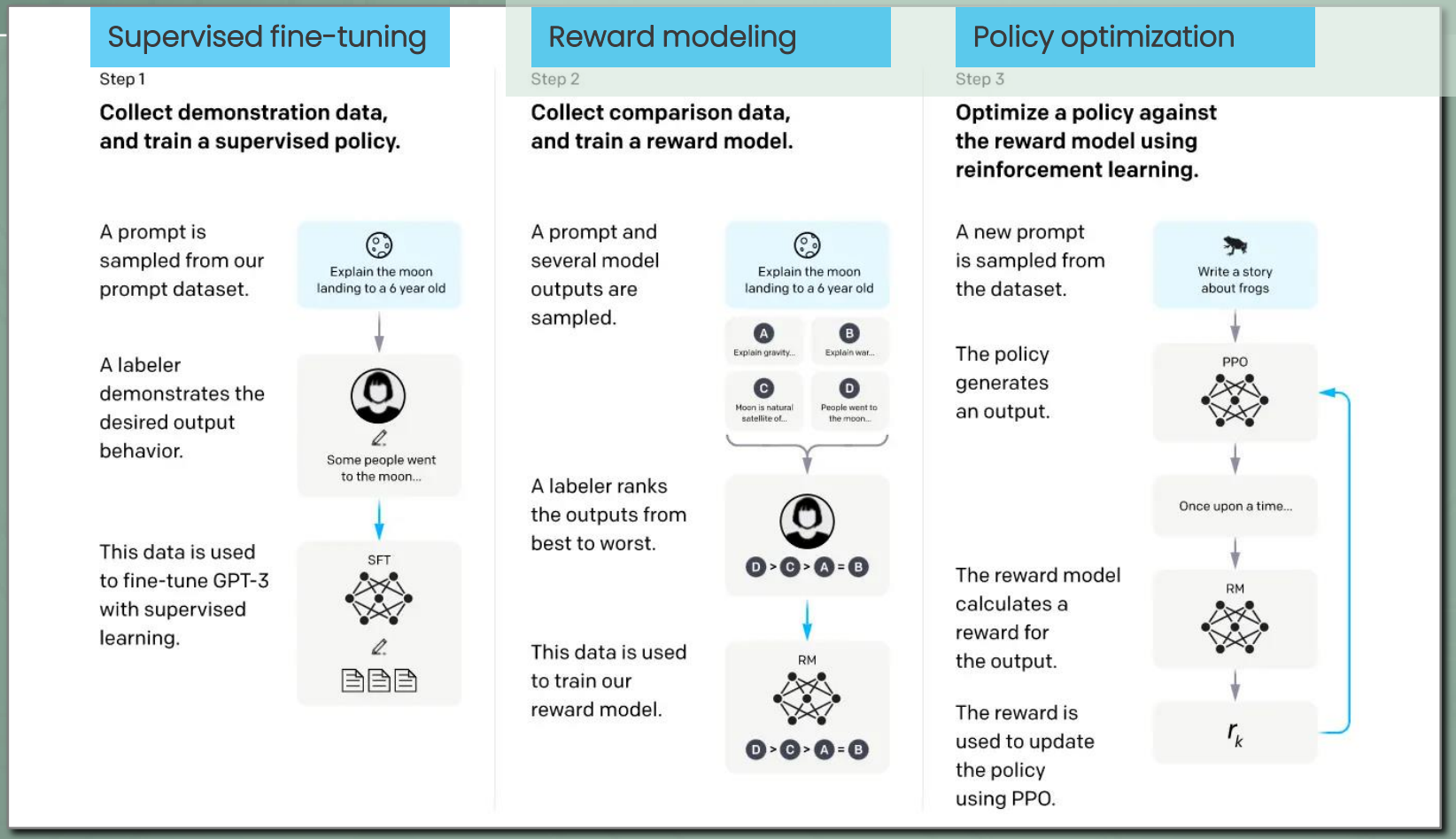
1. Prompt-String is tokenized
2. position-embedded
3. context-embedded through GPTModel
4. most probable continuation token is added
5. tokens are detokenized
6. result is evaluated, LLM weights eventually updated



Training process: Alignment (SFT + RLHF)

Pretrained language model undergoes three-step process using supervised fine-tuning, reward modeling and policy optimization

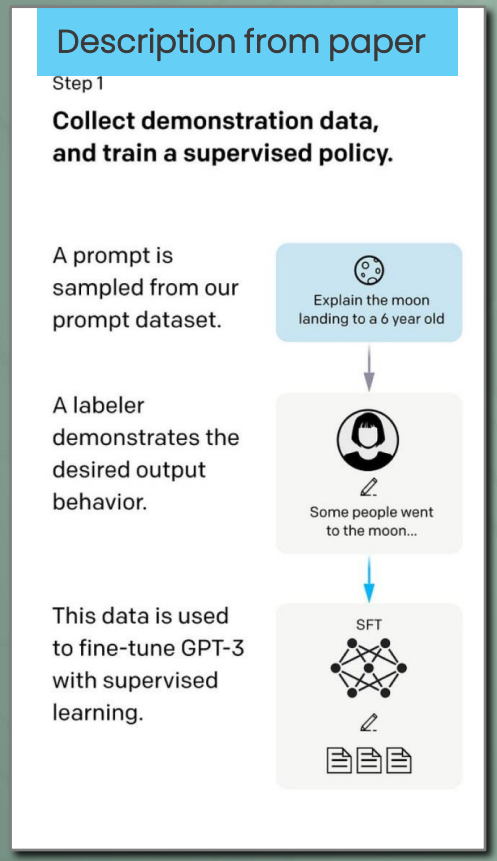
RLHF:
Make model
honest, helpful, harmless

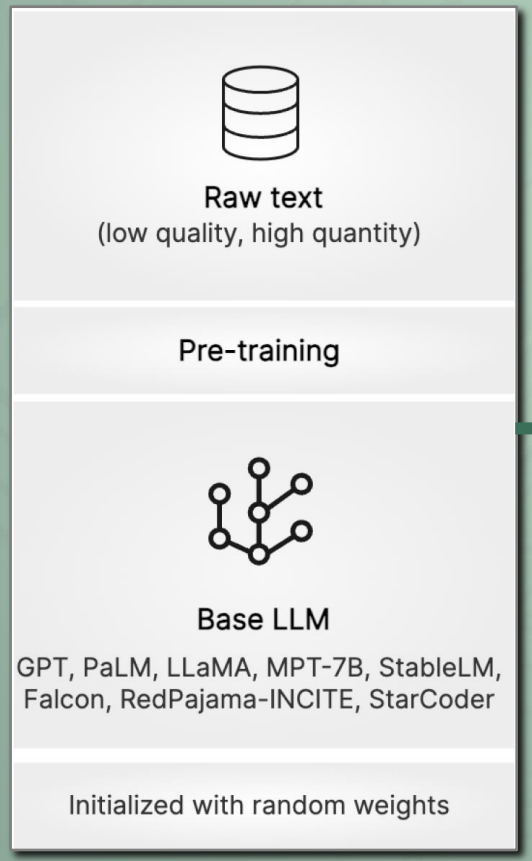


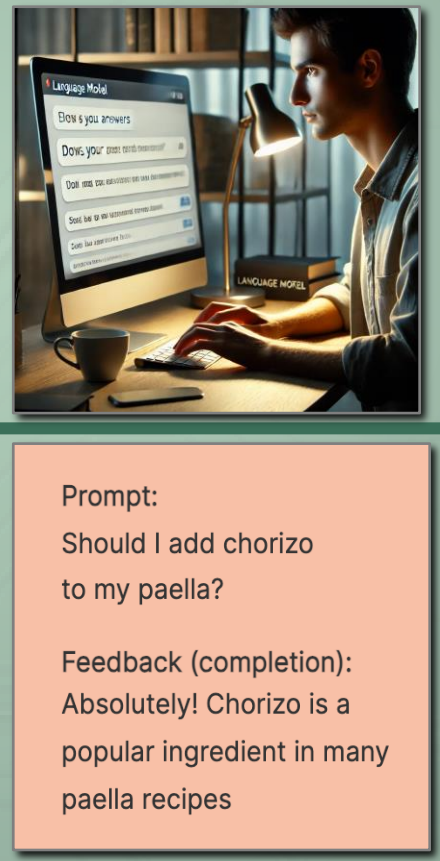
Training language models to follow instructions with human feedback, Ouyang, 2022

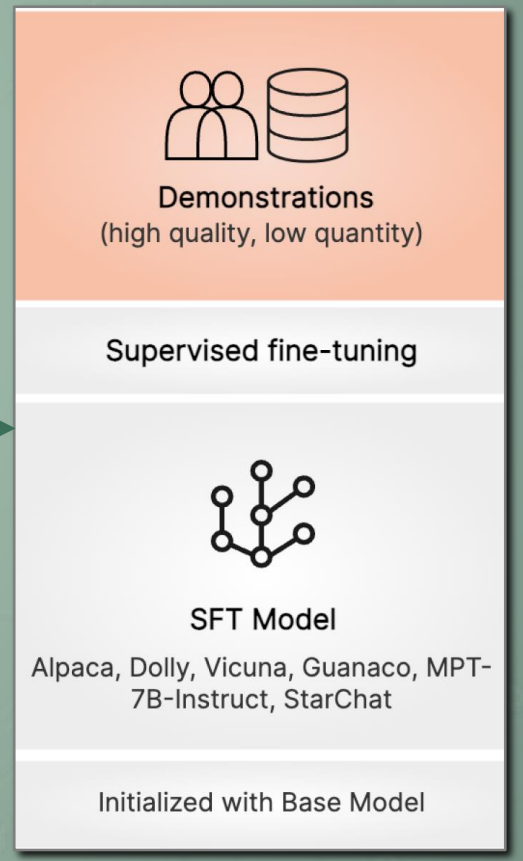
Step 1: Instruction Finetuning

… transforms Base LLMs into question-answering machines by training them on instruction-response pairs, enabling them to respond to wide range of prompts/tasks accurately







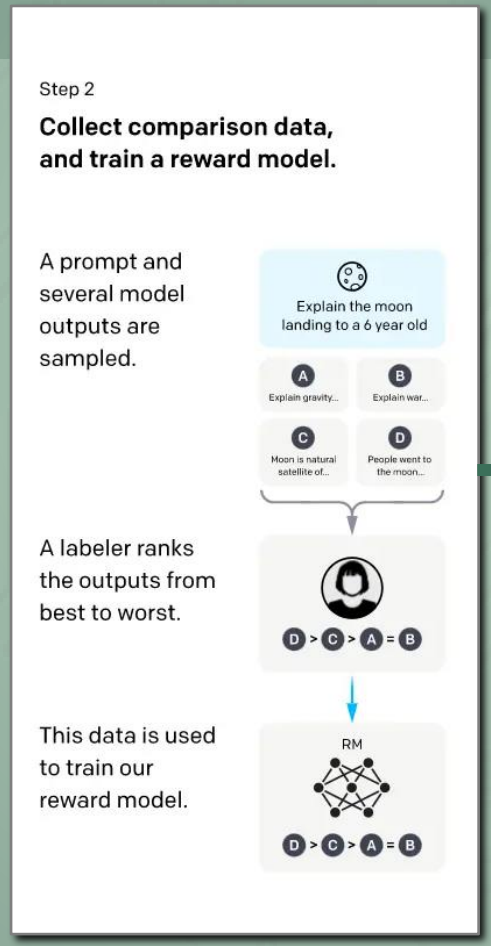


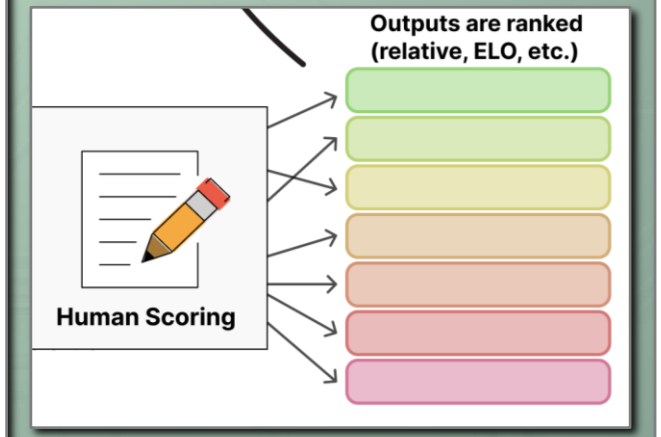
https://klu.ai/glossary/supervised-fine-tuning

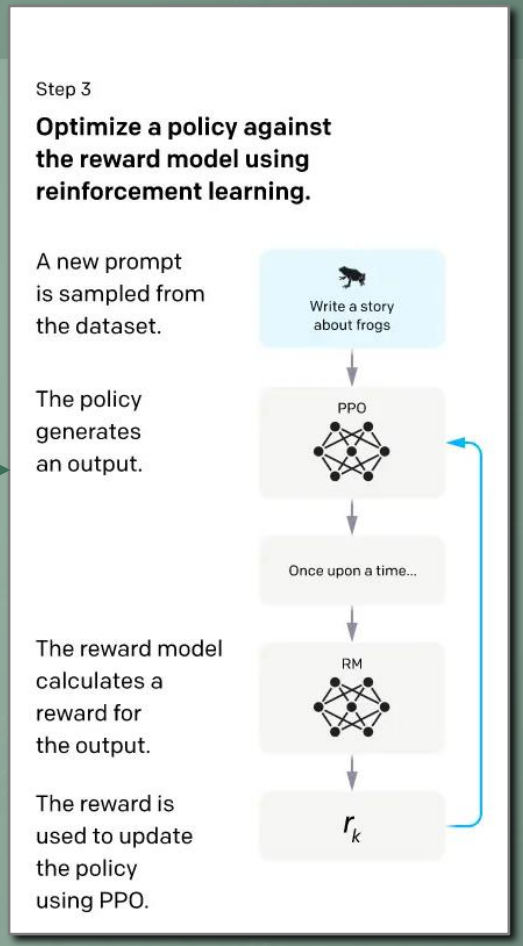
Reinforcement Learning from Human Feedback (RLHF)

… is about making language models honest, helpful, harmless (3H)

- used to align language models with human preferences
- improving their ability to generate more helpful, accurate, and contextually appropriate responses
- across a wide range of tasks and conversations
- Steps:
  - Reward modeling
  - Policy optimization





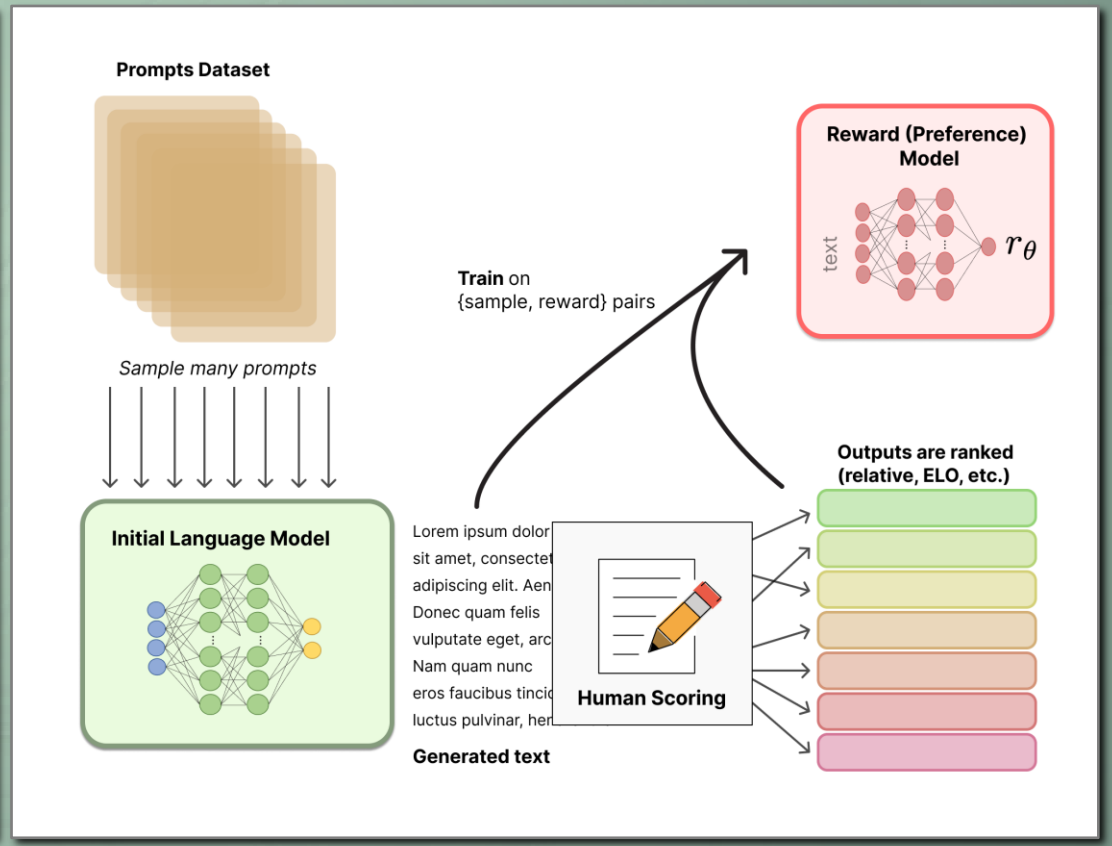


https://huggingface.co/blog/rlhf

Step 2 (RLHF): Reward model

… is typically transformer-based neural network, similar in architecture to the language models it's designed to evaluate, e.g. smaller, obsolete LLM like GPT-2

- **Size:**
  Generally smaller than main language model to be more computationally efficient.
- **Input:**
  Takes in a prompt-response pair (or sometimes just the response, with the prompt as context).
- **Output:**
  Produces a single scalar value representing the predicted quality or preference score for the given input.
- **Training:**
  Fine-tuned on a dataset of human-labeled preferences, often using comparative rankings (e.g., response A is preferred over response B for given prompt).
- **Adaptation:**
  Can be iteratively updated as more human feedback data becomes available during the RLHF process.



https://huggingface.co/blog/rlhf

Step 3 (RLHF): Proximal Policy Optimization

… optimizing Language Model Through Iterative Feedback and Policy Updates

Proximal Policy Optimization (PPO)

- Simple Reinforcement learning algorithm for optimizing AI agent policies
- makes controlled updates to improve performance while avoiding drastic changes
- uses “clipped surrogate objective function”, balancing efficiency and stability

Process

1. Initial language model generates base responses
2. Tuned model (RL policy) produces refined outputs
3. Reward model evaluates based on human preferences
4. PPO algorithm updates policy to maximize rewards
5. Preventing “catastrophic forgetting”:
   KL divergence penalty maintains consistency with base model, i.e. base model must not drift too must

→ Human feedback for reward model is ongoing!

https://cobusgreyling.medium.com/catastrophic-forgetting-in-llms-bf345760e6e2

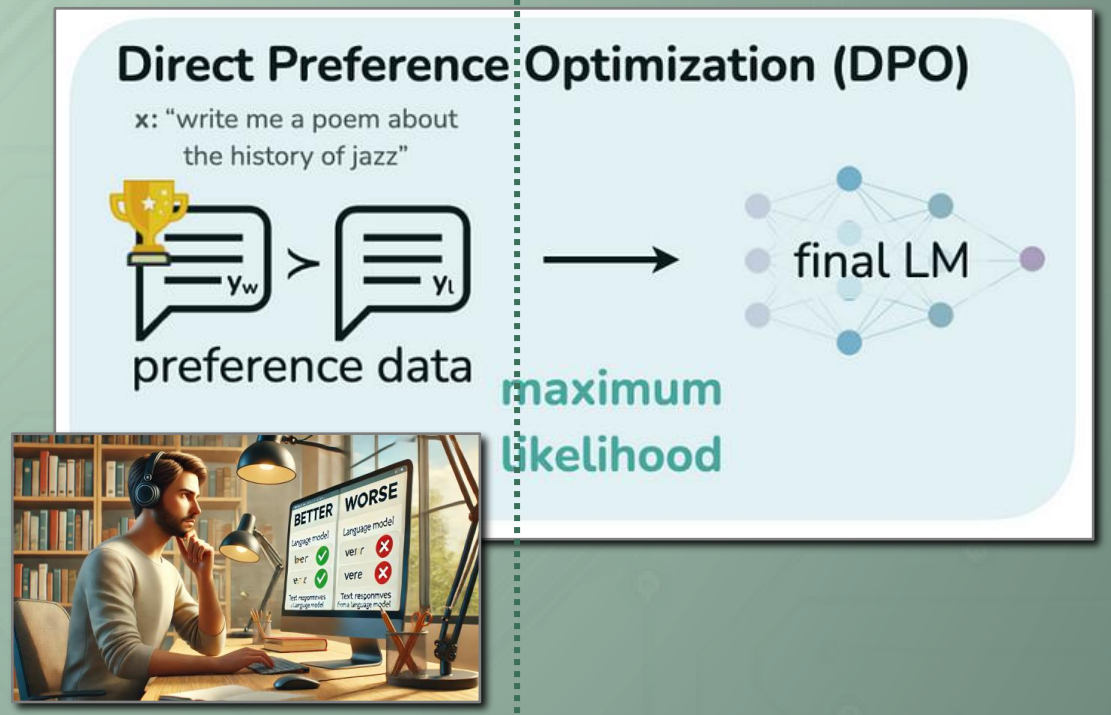
Training process: DPO

… offers streamlined alternative to complex RLHF pipelines, potentially revolutionizing preference-based language model training

Process

1. Starts with pre-trained, instruction finetuned language model.
2. Collects pairs of responses, where humans have indicated preference for one over the other.
3. Instead of training a separate reward model, DPO directly optimizes the language model to produce preferred outputs.
4. It uses loss function that encourages the model to assign higher probability to preferred responses.
5. Optimization process maintains a balance between improving on preferences and staying close to the original model's behavior.

→ Human feedback is collected beforehand, not in real-time during training

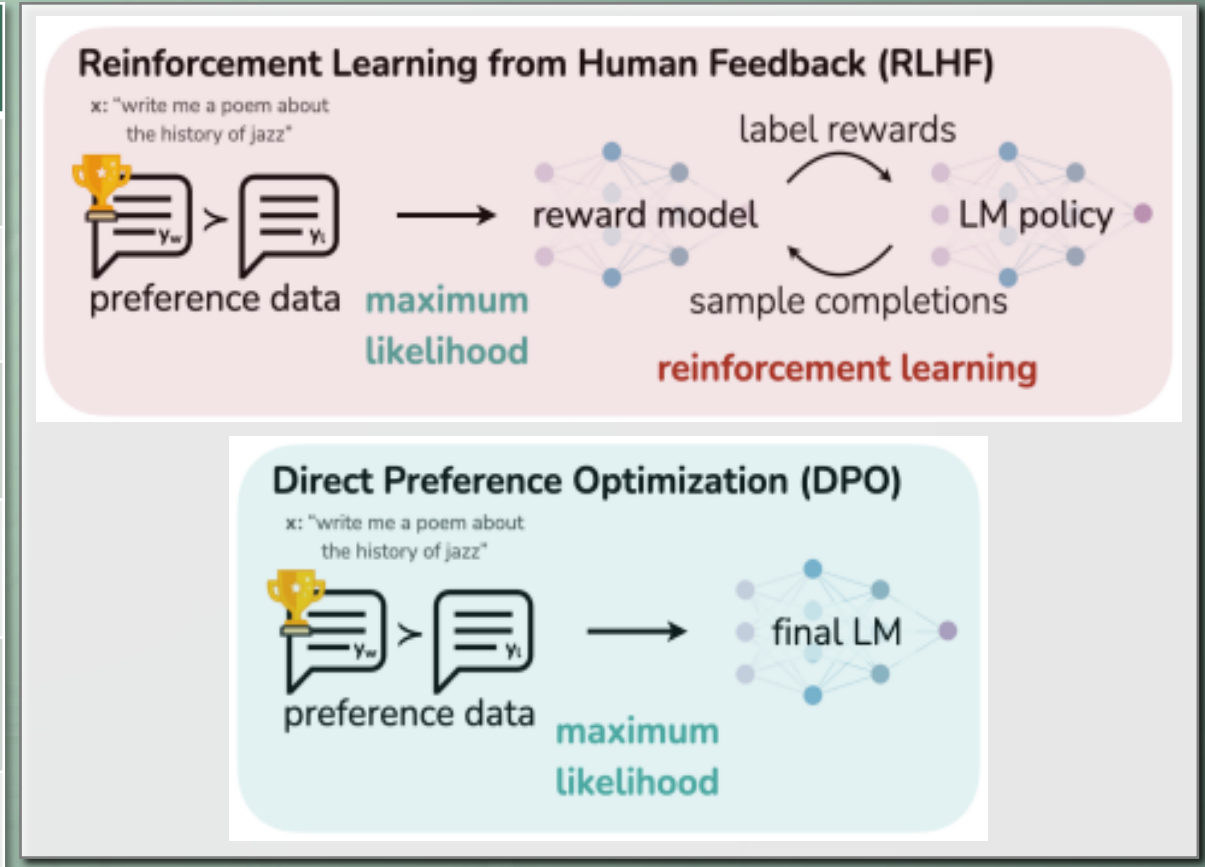


https://huggingface.co/blog/pref-tuning
Direct Preference Optimization, 2023, https://arxiv.org/abs/2305.18290

Comparison: RLHF/PPO vs. DPO

RLHF/PPO offers greater flexibility, adaptability at cost of computational complexity
DPO provides simpler approach that may sacrifice adaptability, fine-grained control.

Aspect	RLHF with PPO	DPO
Approach	Simple Reinforcement Learning	Supervised Learning
Complexity	high	▪ Lower ▪ however needs more data
Reward Model needed	yes	no
Training Process	Needs ongoing human feed-back loop	Uses precollected preference data
Advantages	▪ More flexibility ▪ fine-grained control ▪ Needs much less data	▪ Simpler, more stable implementation ▪ easier to maintain
Examples	▪ OpenAI ChatGPT ▪ Anthropic Claude	▪ Mistral ▪ Meta Llama 3

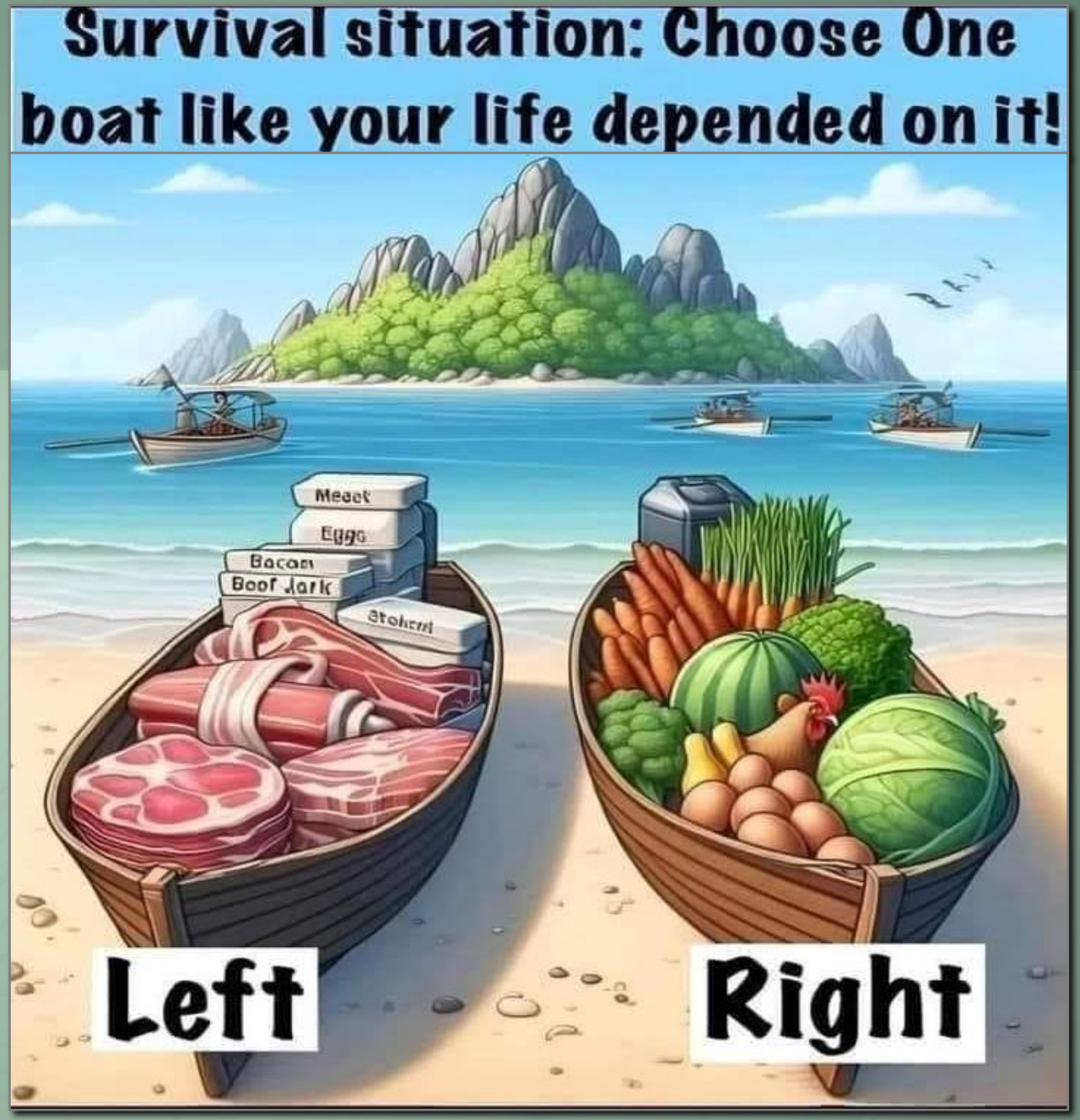


https://www.linkedin.com/pulse/rlhf-dpo-simplifying-enhancing-fine-tuning-language-models-kirouane/

RLHF/PPO, DPO

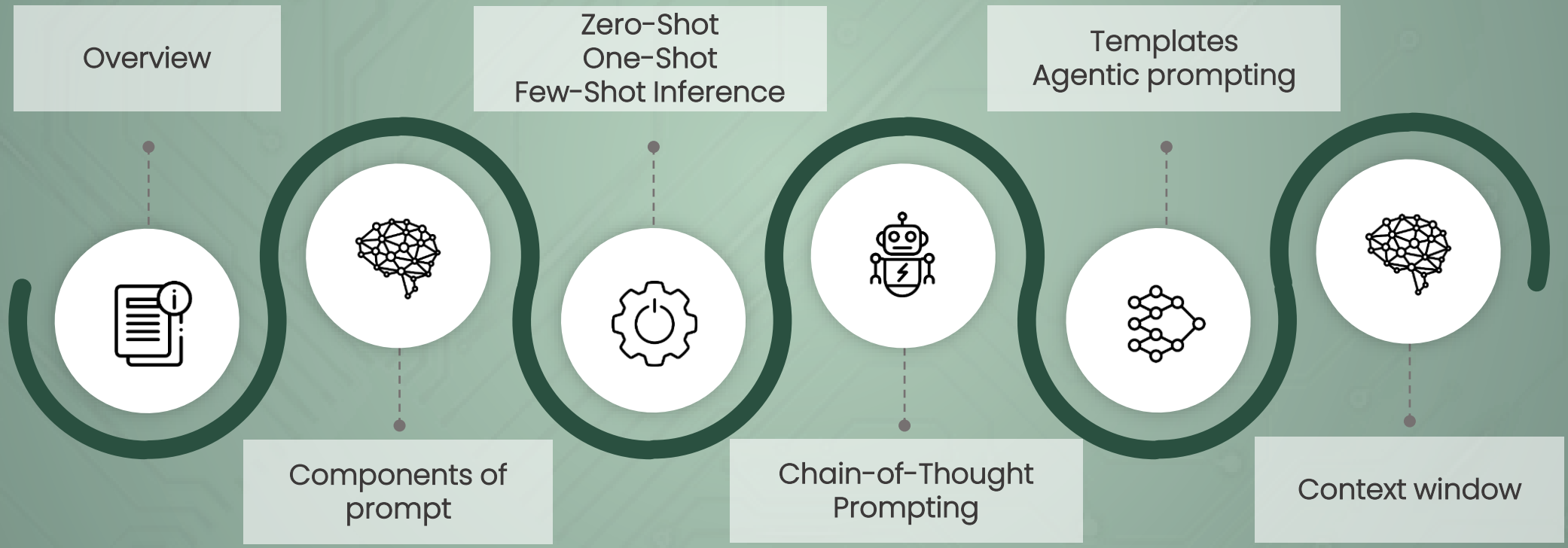
… face same challenge:
preferences are individual

- selection of “better” answer is often based on individual preferences
- No perfect solution, however as outcome LLM must be
  - Honest
  - Helpful
  - harmless



Data Augmentation: Prompt Engineering

… is the art and science of crafting precise and effective input prompts to elicit desired outputs from large language models, optimizing their performance for specific tasks or applications



Prompt Engineering Overview

What is it good for?

Improved Relevance, Specificity

- Guides model to generate content more closely aligned with the intended topic or purpose
- Encourages more detailed and precise outputs by providing clearer expectations
- Allows augmentation of new data beyond training set
- Allows to use roles, examples:
  - explain like you are pirate
  - Explain like I am 5 (ELI-5)

Consistency Control

- Helps maintain consistent tone, style, format throughout generated text.

Creativity Direction

- Can steer model's creative outputs in desired directions

Components of a Prompt

… include user input (query, task), instruction set (response guidelines), context (background information, examples), system prompt (overarching instructions shaping model's behavior).

Aspect	User Input	Instruction Set	Context	System Prompt
Description	▪ specific query or task ▪ provided by end-user	Guidelines, context to direct model's response	Relevant background information, examples, data to inform model's output	Overarching instructions that shape model's behavior across all interactions
Example	"Where do migratory birds spend the winter?"	"Answer the following question and explain it as if the user was five years old."	Examples from a VectorDB search for closest document	You are an expert ornithologist specializing in bird migration patterns and winter habitats. Your role is to provide accurate, informative answers about where migratory birds spend their winters.
Purpose	Defines ▪ primary question ▪ task model needs to address	Shapes ▪ Format ▪ tone ▪ complexity of model's output	Provides additional information to make the response more accurate or relevant	Sets overall tone, role, and boundaries for model's interactions
Potential Variations	Can range from ▪ simple questions to ▪ complex multi-part requests	May include formatting instructions, language preferences, or specific output requirements	Can include historical data, related examples, or domain-specific knowledge	Can define model's persona, ethical guidelines, specific domain expertise

Prompting Language Considerations

… use clear, simple language, be specific about expectations, maintain logical structure, include illustrative examples, and tailor prompt to AI system's capabilities

Use plain language

- Avoid jargon, complicated terms.
- Make instructions easy to understand so the AI quickly grasps intent.

Be specific

- Clearly state expectations, desired outcomes.
- Provide detailed information

Maintain logical structure

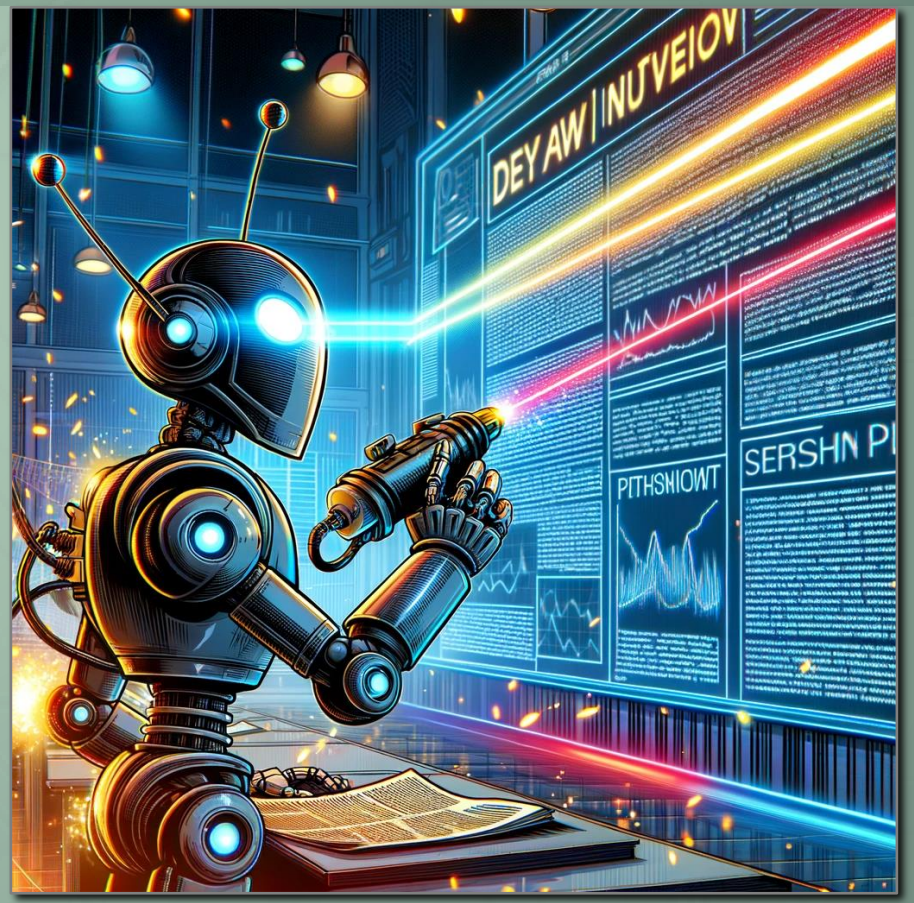
- Organize prompt in coherent manner

Include examples

- If possible, provide examples to illustrate epected type of output
- This helps AI generate more accurate results.

Consider context

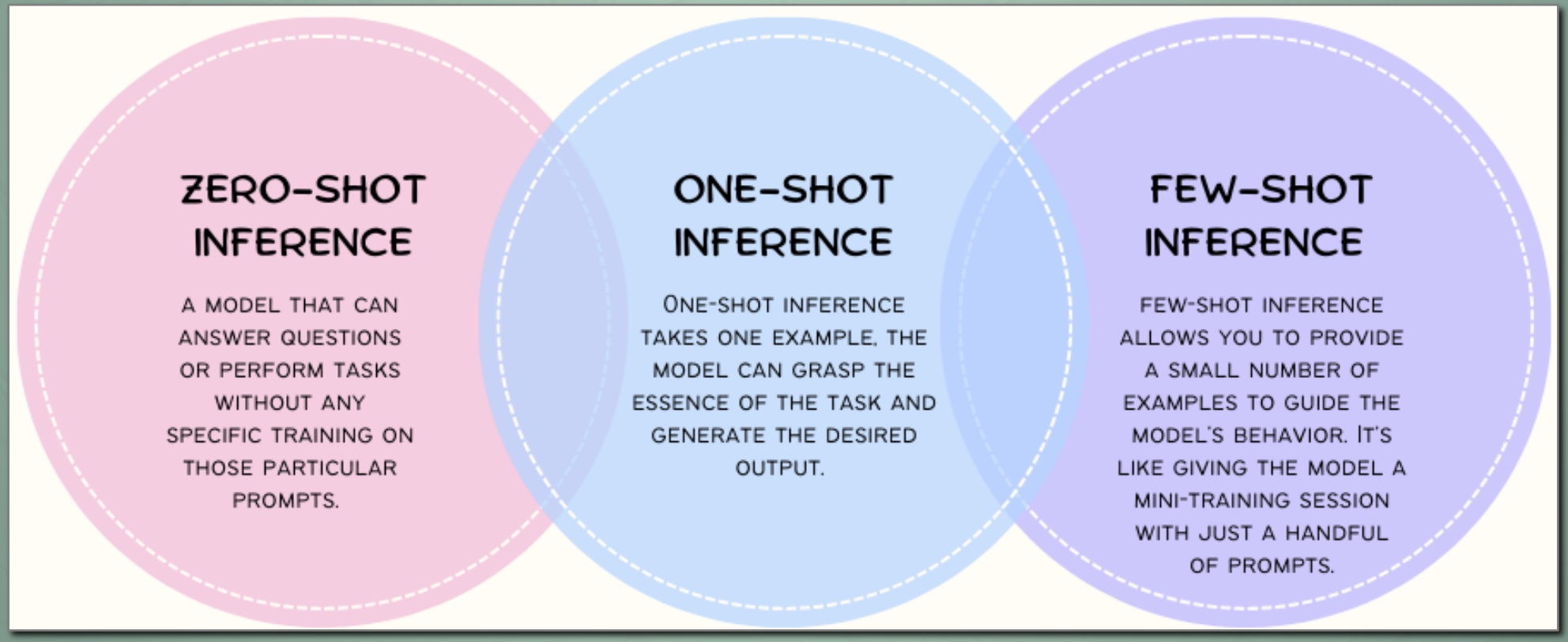
- Tailor your prompt to the AI system and its capabilities.



https://promptengineering.org/master-prompt-engineering-demystifying-prompting-through-a-structured-approach/

From Zero-Shot to few shot Prompting

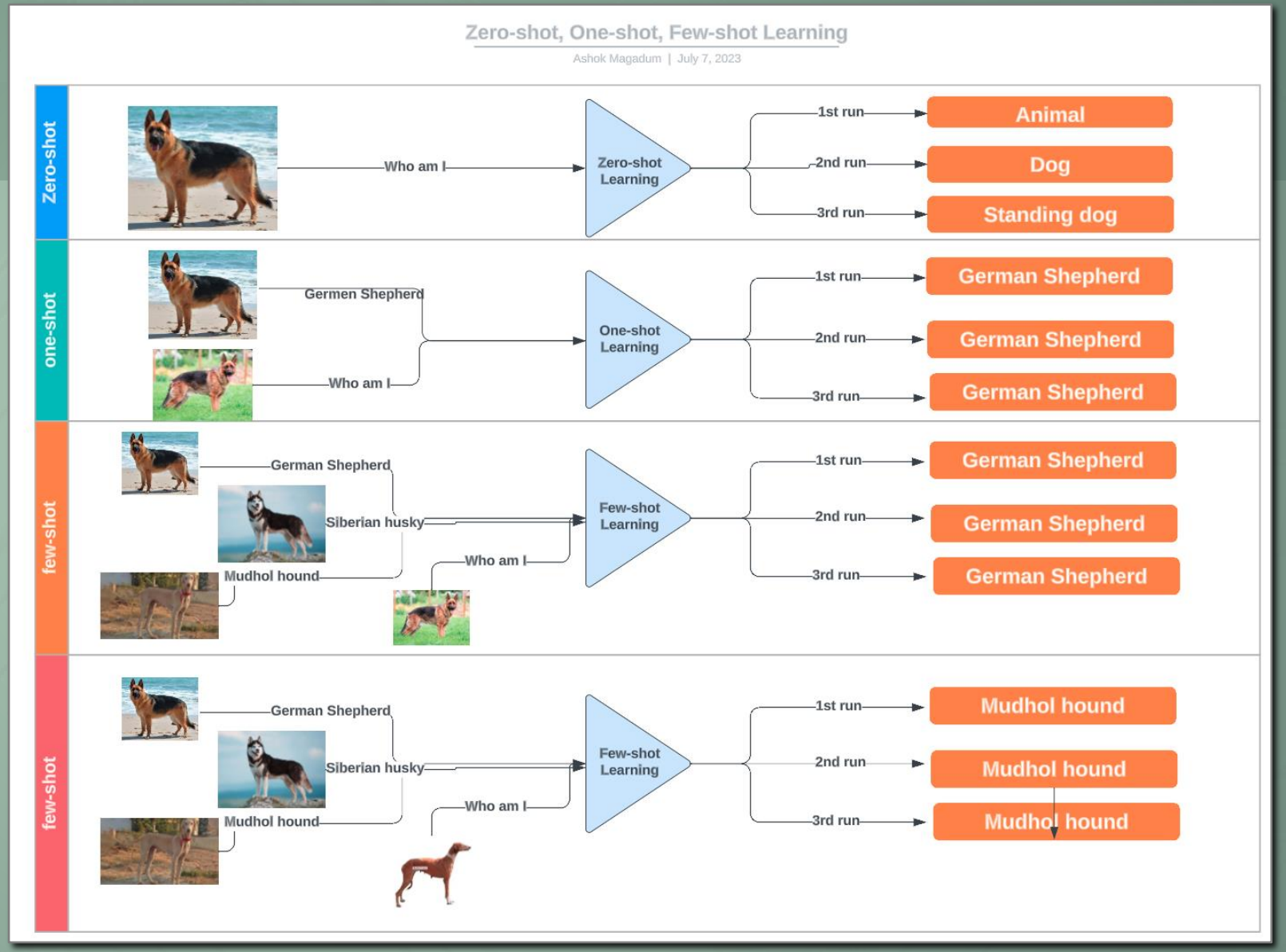
Basically LLM is zero-shot learner, i.e. it transfers knowledge to new contexts.
However by adding one or few examples performance significantly increases.



https://medium.com/@S.Shakir/unleashing-the-power-of-prompt-engineering-zero-shot-one-shot-and-few-shot-inference-29b88ba16ee0
https://machinelearningmastery.com/what-are-zero-shot-prompting-and-few-shot-prompting/
https://alphamoon.ai/blog/zero-shot-learning/

From Zero-Shot to few shot Prompting

… Example of increasing precision through examples



https://www.datacamp.com/blog/what-is-few-shot-learning?dc_referrer=https%3A%2F%2Fwww.google.de%2F
https://medium.com/ubiai-nlp/step-by-step-guide-to-mastering-few-shot-learning-a673054167a0

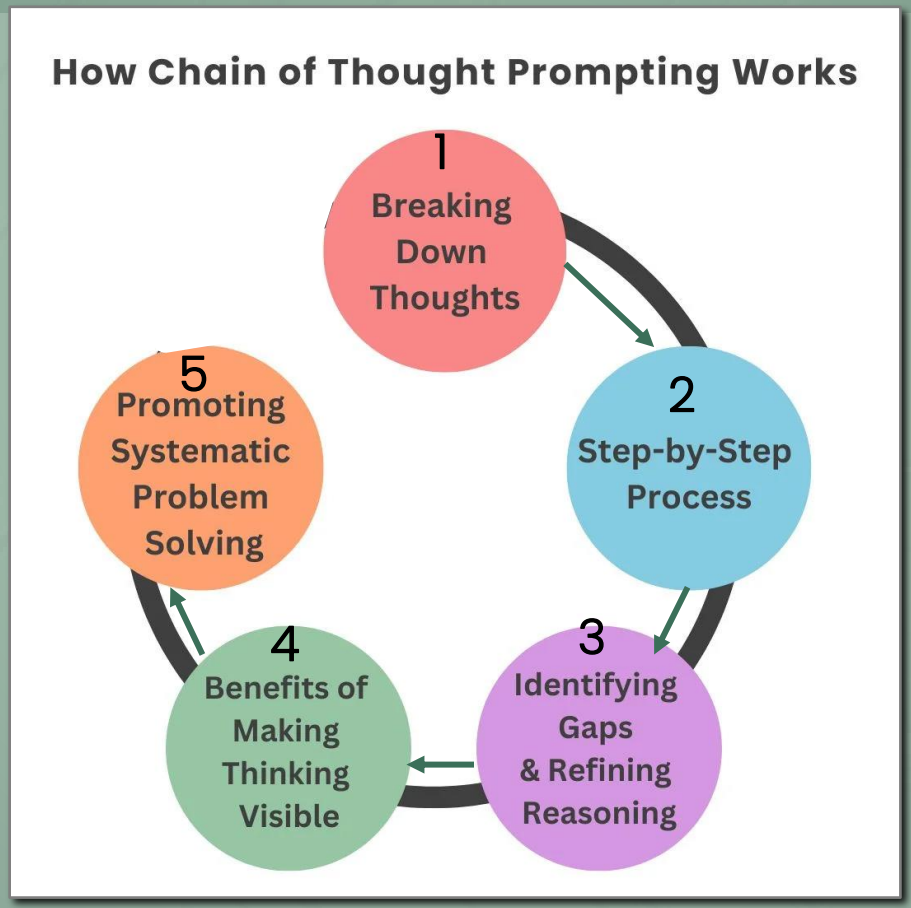
Chain-of-Thought Prompting

… is technique in natural language processing where model generates intermediate reasoning steps to improve accuracy and coherence of its final output

1. Bearking down thoughts
   - Chain of Thought (CoT) Prompting encourages people to dissect their thought processes into smaller, connected steps.
   - This method helps reveal fundamental logic, reasoning behind their ideas.

2. Step-by-step Process
   - Approach begins by presenting initial question or issue.
   - Participants then divide thoughts into manageable segments.
   - Each step is clearly expressed using linking phrases such as “because,” “therefore,” or “as a result.”

3. Identifying Gaps & Refining Reasoning
   - As individuals progress through process, they may identify gaps in their reasoning.
   - They can gather additional information to refine their thoughts and strengthen their arguments.



4. Benefits of Making Thinking Visible
   - CoT Prompting makes thought process visible and explicit.
   - This helps individuals better understand their own reasoning.
   - It also enables them to communicate their ideas more effectively to others.

5. Promoting Systematic Problem-Solving
   - Technique fosters systematic and thorough approach to problem-solving.
   - It encourages individuals to consider multiple viewpoints. It also aids them in exploring alternative solutions.

https://aiperceiver.com/everything-you-need-to-know-about-chain-of-thought-prompting/

Chain-of-Thought Prompting Example

Language models cannot contemplate logically. Chain-of-thought helps simulate logical thinking

Prompt:
Solve the following word problem step by step:
"A bakery sold 136 loaves of bread on Monday. On Tuesday, they sold 42 more loaves than on Monday. On Wednesday, they sold half as many loaves as they did on Tuesday. How many loaves of bread did the bakery sell in total over these three days?"

Please show your reasoning for each step.

Expected Response:
Step 1: Calculate the number of loaves sold on Tuesday
- Monday sales: 136 loaves
- Tuesday sold 42 more than Monday
- Tuesday sales = 136 + 42 = 178 loaves

Step 2: Calculate the number of loaves sold on Wednesday

- Wednesday sold half as many as Tuesday
- Wednesday sales = 178 ÷ 2 = 89 loaves

Step 3: Sum up the total loaves sold over three days

- Total = Monday + Tuesday + Wednesday
- Total = 136 + 178 + 89 = 403 loaves

Therefore, the bakery sold a total of 403 loaves of bread over these three days.

Template-Based Prompting

… is structured approach to crafting AI prompts using pre-defined formats or patterns to ensure consistency and effectiveness in AI interactions

Benefits of Template-Based Prompting

- Consistency:
  Ensures a uniform structure across multiple prompts
- Efficiency:
  Streamlines the prompt creation process
- Clarity: Provides clear instructions to the language model
- Customization:
  Allows for easy adaptation to various tasks and contexts
- Reproducibility:
  Facilitates consistent results across different runs

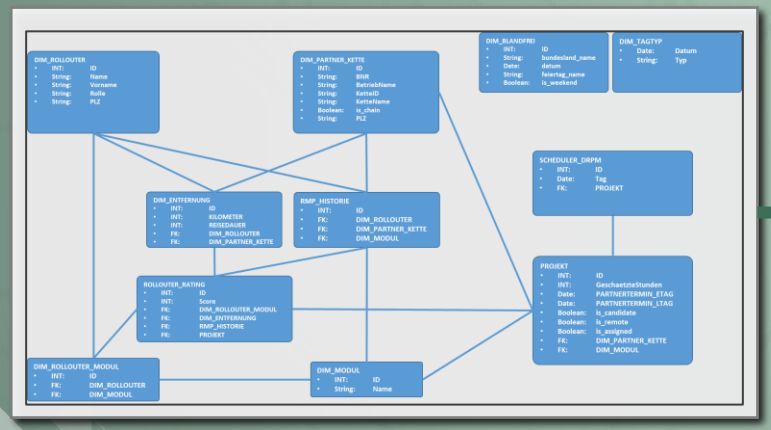
```
# Python example of prompt template

sentiment_template = PromptTemplate(
    "Analyze the sentiment of the following text:\n\n" +
    "Text: {text}\n\n" +
    "Provide your analysis in the following format:\n" +
    "Sentiment: [Positive/Negative/Neutral]\n" +
    "Confidence: [0-100]%\n" +
    "Key phrases: [List key sentiment indicators]"
)
```

```
# Using the template

text = "I absolutely loved the new restaurant! The food
was amazing."

prompt = sentiment_template.format(text=text)

print(prompt)
```

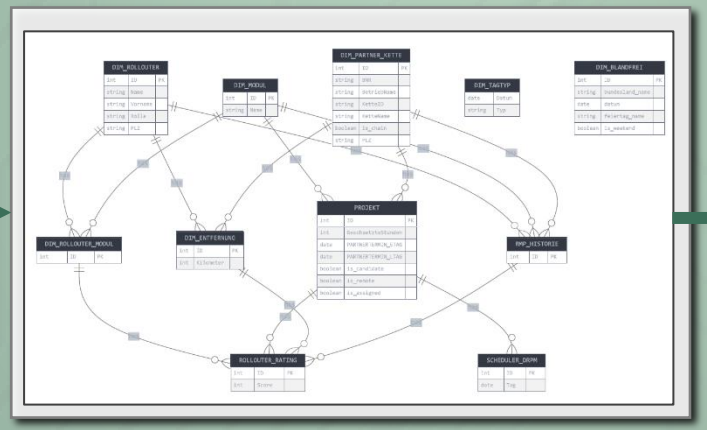
Agentic prompting

Example: Business process automatization with Java Spring Boot can be generated by agentic pipeline in Python, i.e. consecutive prompts that use results of previous steps

Entity-Relation diagram (PowerPoint)



Entity-Relation diagram (technical)



Backend (Java Spring Boot) With different layers

CONTROLLER:	REST API
SERVICE:	Business logics
REPOSITORY:	Technical
DTO:	Technical
MODEL:	Database

Agentic prompting

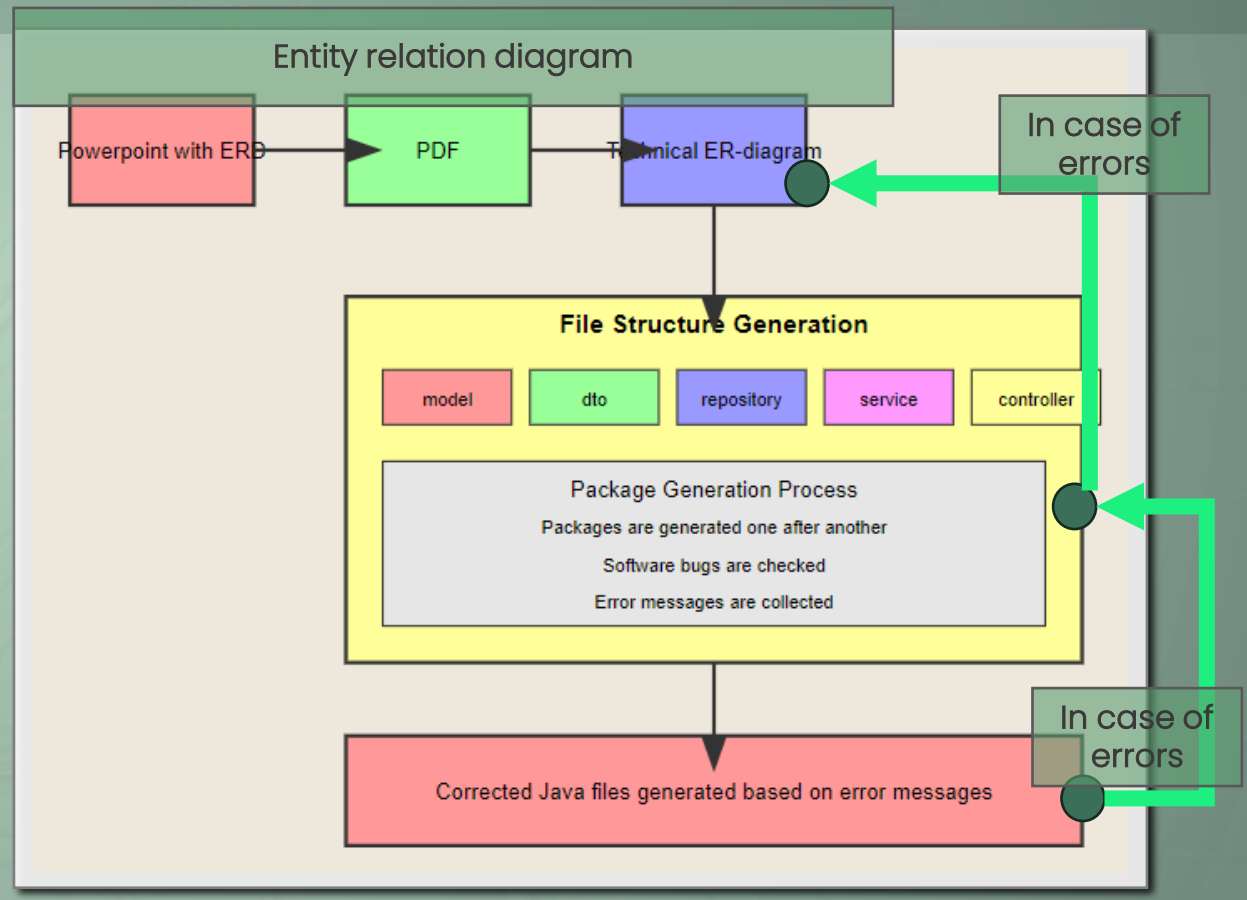
Example: Business process automatization with Java Spring Boot can be generated by agentic pipeline in Python, i.e. consecutive prompts that use results of previous steps

Python controls following

- Entity Relation diagram is made by hand in Powerpoint
- Powerpoint is saved in PDF and transformed by LLM to technical ER diagram
- Java Spring Boot file structure is generated
- Based on it all Java packages are generated
- All files are saved at correct position
- If bugs occur they are put into LLM, which then generates corrected files
- Finally program runs correctly

Problem with agentic pipelines

- results are not 100% correct.
- Therefore at each step results must be checked, step must be repeated eventually.

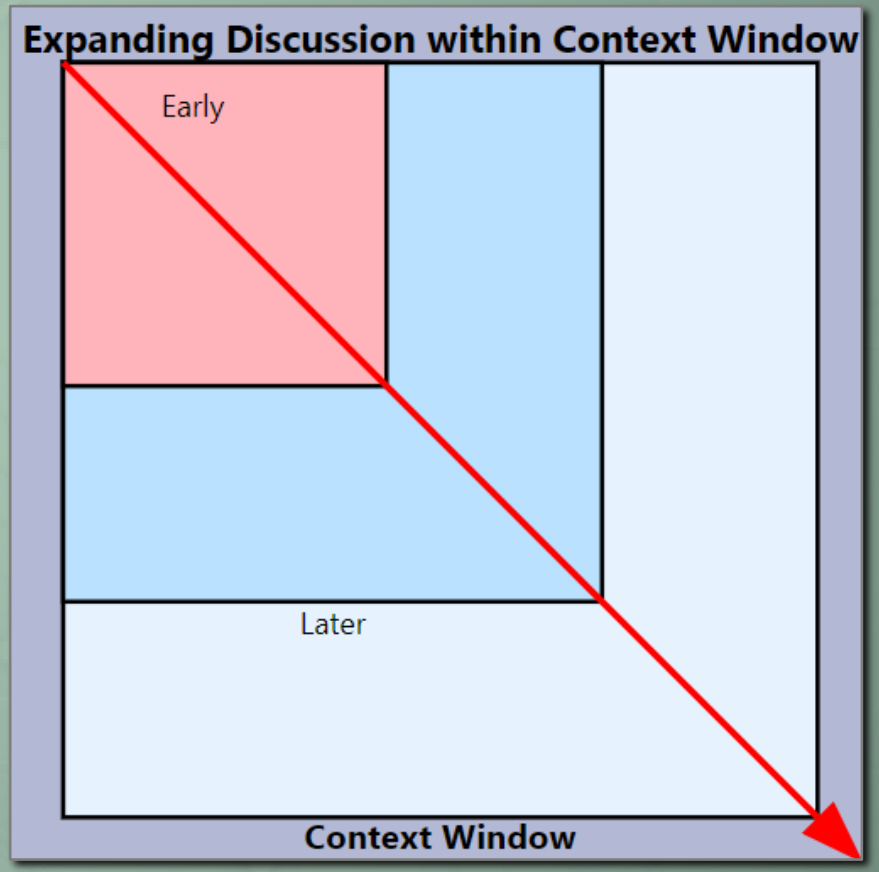


Context window

… or input window, token limit, refers to maximum amount of text (i.e. tokens) that a language model can process and consider at one time when generating responses

- Conversation thread of LLM is basis for referring next questions, i.e. answer to questions also depends on
  - conversation history
  - Additional prompt information, like instruction set, context, system prompt
- All conversation history and question are
  - Tokenized
  - Squared
  - All tokens tuples are evaluated, i.e. attention is calculated
- Based on it, next most probable token is generated.

→ Context window is maximal size of thread



https://promptengineering.org/master-prompt-engineering-demystifying-prompting-through-a-structured-approach/

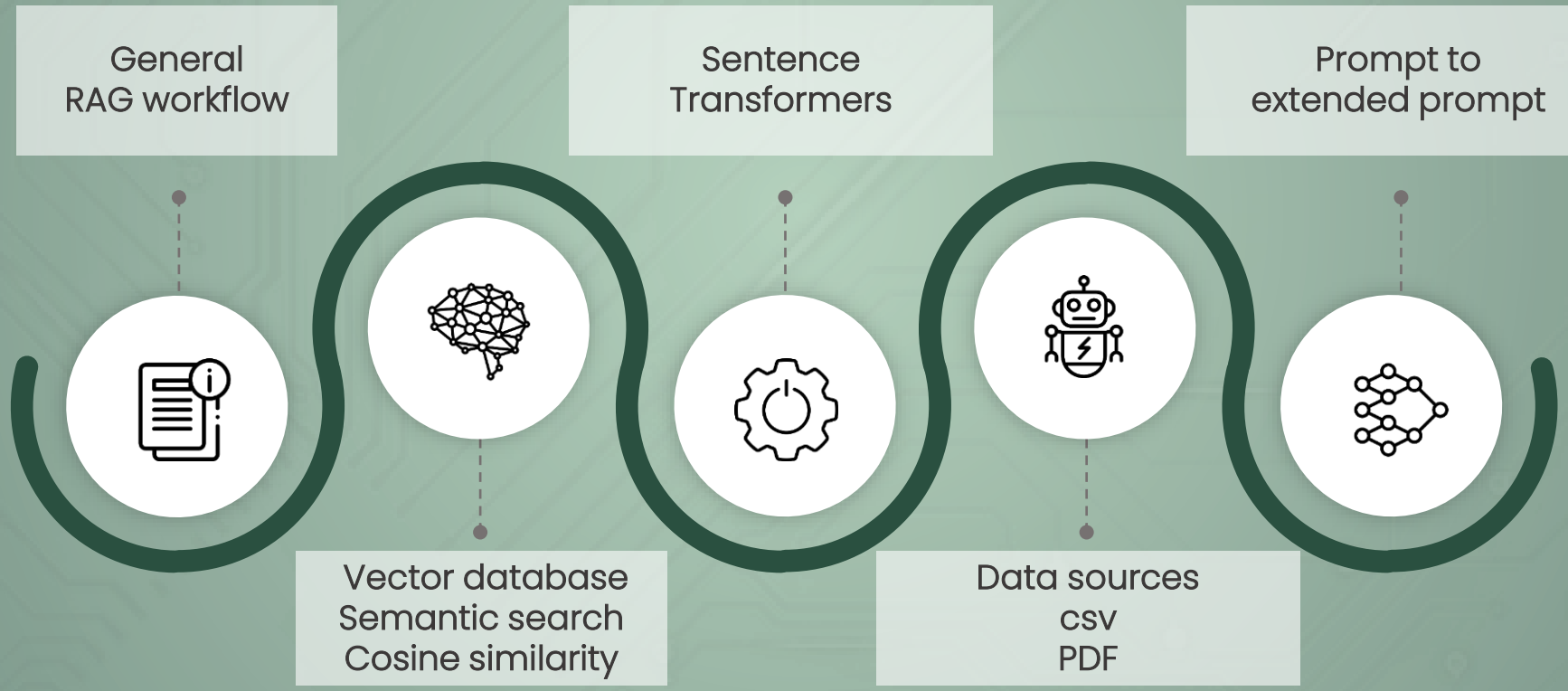
Context window

… or input window, token limit, refers to maximum amount of text (i.e. tokens) that a language model can process and consider at one time when generating responses

LLM	OpenAI GPT-4	Anthropic Claude Sonnet 3.5	Llama 3.1 70B	Mistral 7B	Gemini 1.5
Input Context window (number tokens)	128k	200k	128k	32k	1M (up to 2M)
Maximum output tokens (number tokens)	4096	4096	2048	4096	8192
Words	24.000	48.000	24.000	6.000	192k (up to 384k)
Pages	48	96	48	8	384 (up to 768)

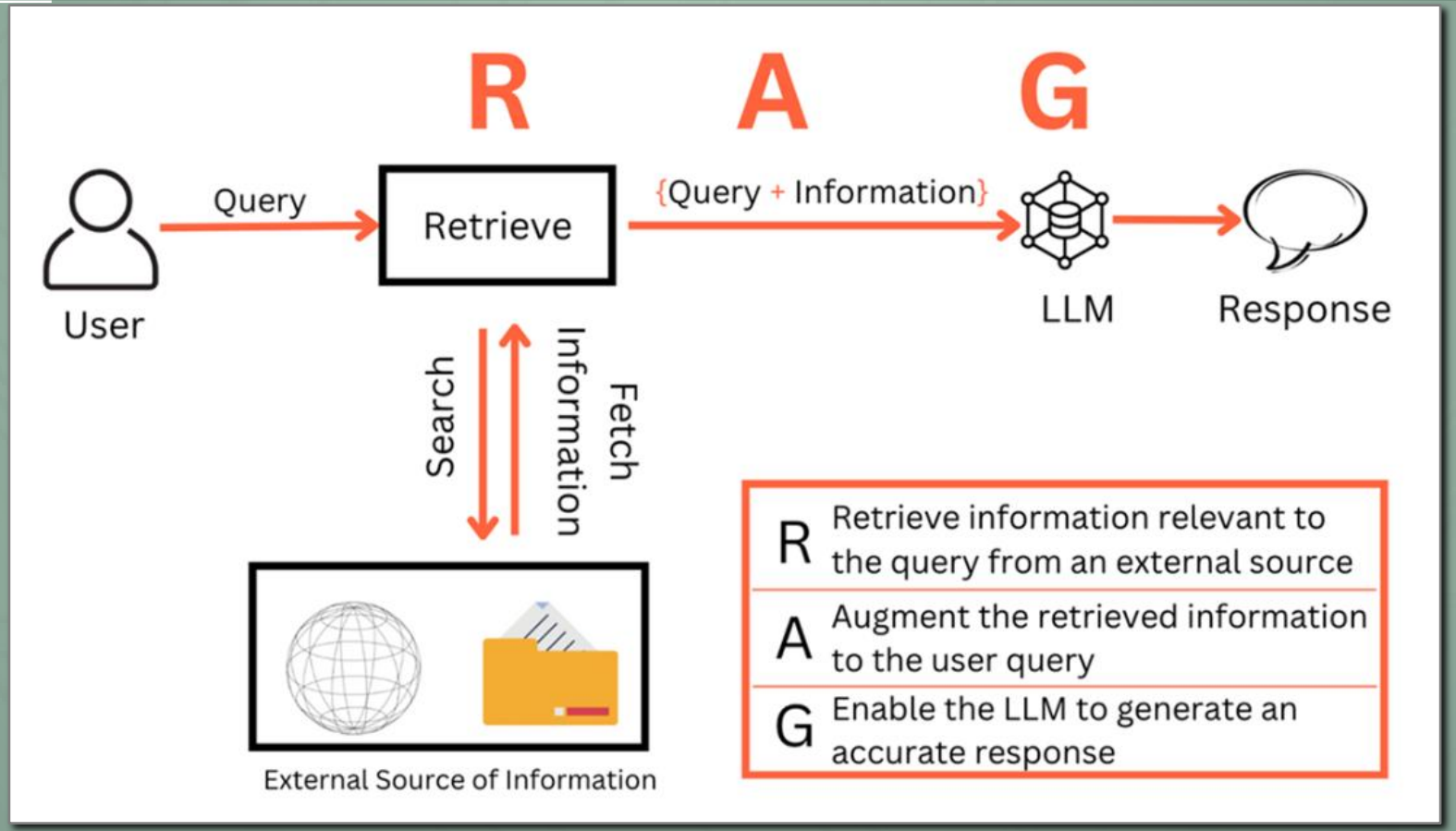
Data Augmentation: Retrieval Augmented Generation

… enhances LLMs by retrieving relevant information from external sources before generating response, allowing model to access up-to-date, specific information beyond training data



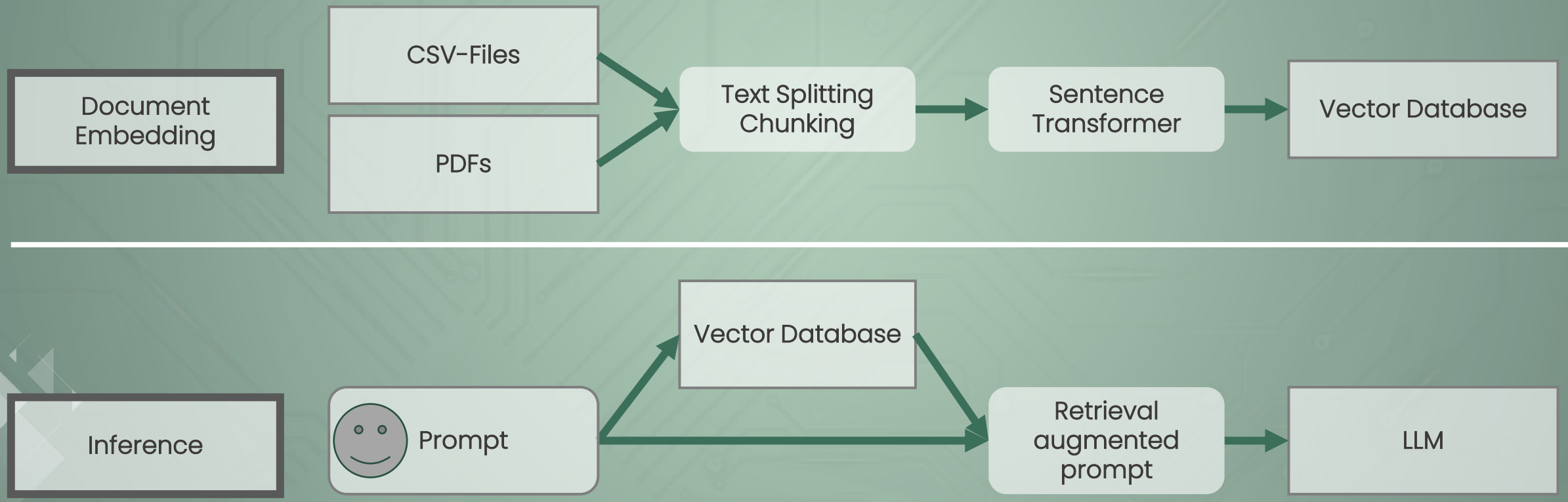
Retrieval Augmented Generation (RAG)

… enhances language model responses by integrating relevant external information with user queries before processing by a language model



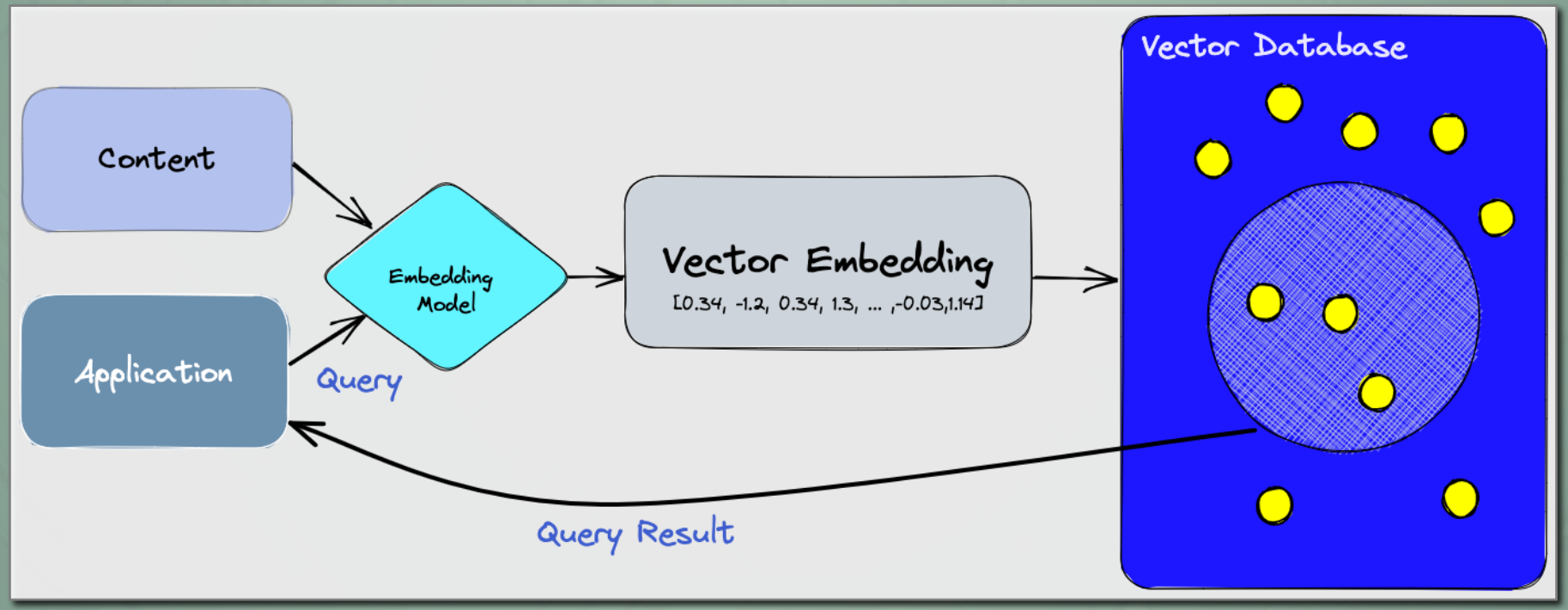
Document embedding vs. inference using vector database

Document embedding transforms data into vectors; inference combines query with relevant vectors for LLM processing.



Vector Databases

… are designed to store, manage, query high-dimensional vector data efficiently, being optimized for semantic similarity search operations on large-scale datasets of embeddings.

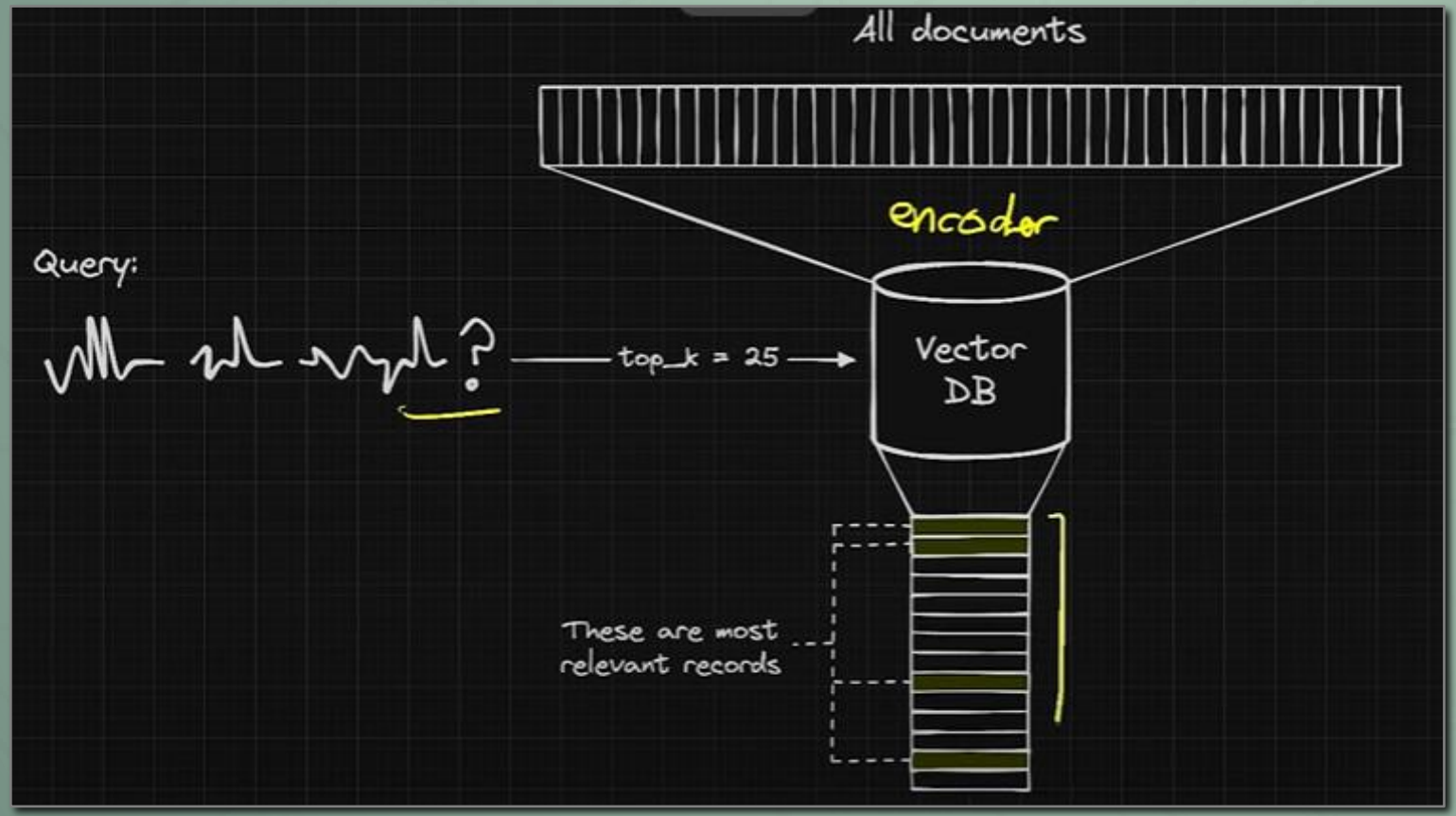


Why vector embedding?
Semantic similarity of documents must be represented by vector similarity

https://www.pinecone.io/learn/vector-database/

Vector Databases

Based on query / prompt the vector database provides k (here =25) most relevant records based on semantic similarity search, e.g. with cosine similarity



Example of Semantic Search in vector space

https://medium.com/nlplanet/two-minutes-nlp-sentence-transformers-cheat-sheet-2e9865083e7a
https://www.youtube.com/watch?v=emKkoeFjH_M

Cosine similarity

… is metric used to measure how similar two documents represented by vectors are irrespective of their size. It is commonly used for semantic search in vector databases.

It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction.

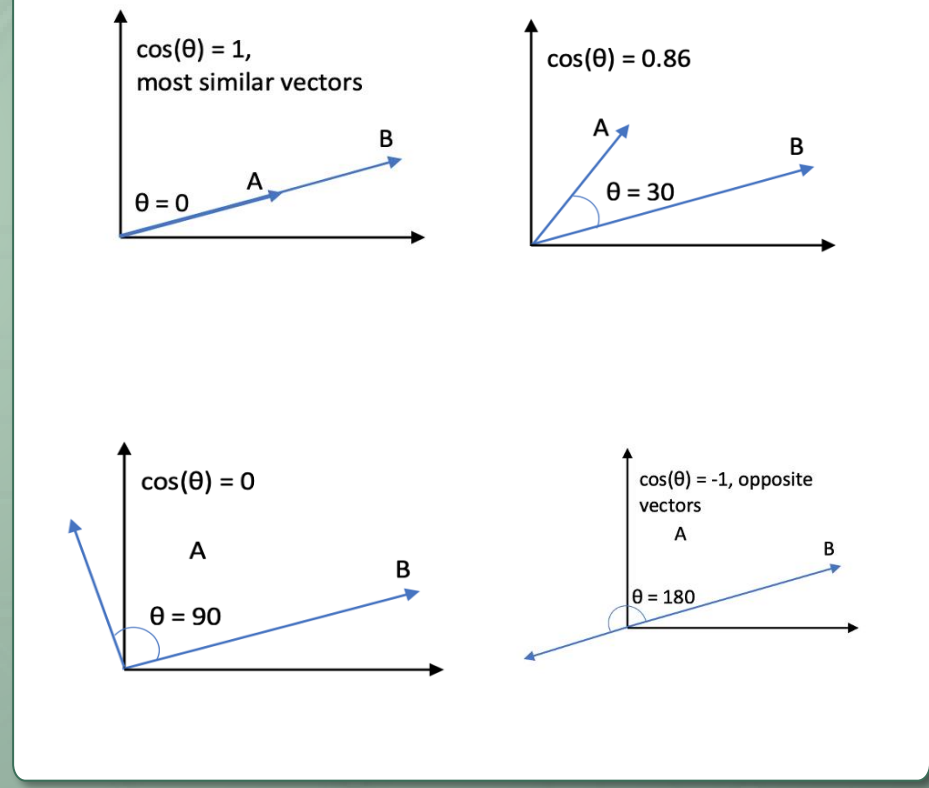
Formula

$\cos\boldsymbol{\theta} = \frac{\vec{a}\times\vec{b}}{\|\vec{a}\|\times\|\vec{b}\|}$ with $\|\vec{a}\| = \sqrt{a_1^2 + a_2^2 + \cdots a_n^2}$ and $\|\vec{b}\| = \sqrt{b_1^2 + b_2^2 + \cdots b_n^2}$

- $\vec{a}$ and $\vec{b}$ are vectors, $\vec{a} \times \vec{b}$ is their dot product
- $\|\vec{a}\|$ and $\|\vec{b}\|$: magnitudes i.e. lengths of vectors $\vec{a}$ and $\vec{b}$

Explanation

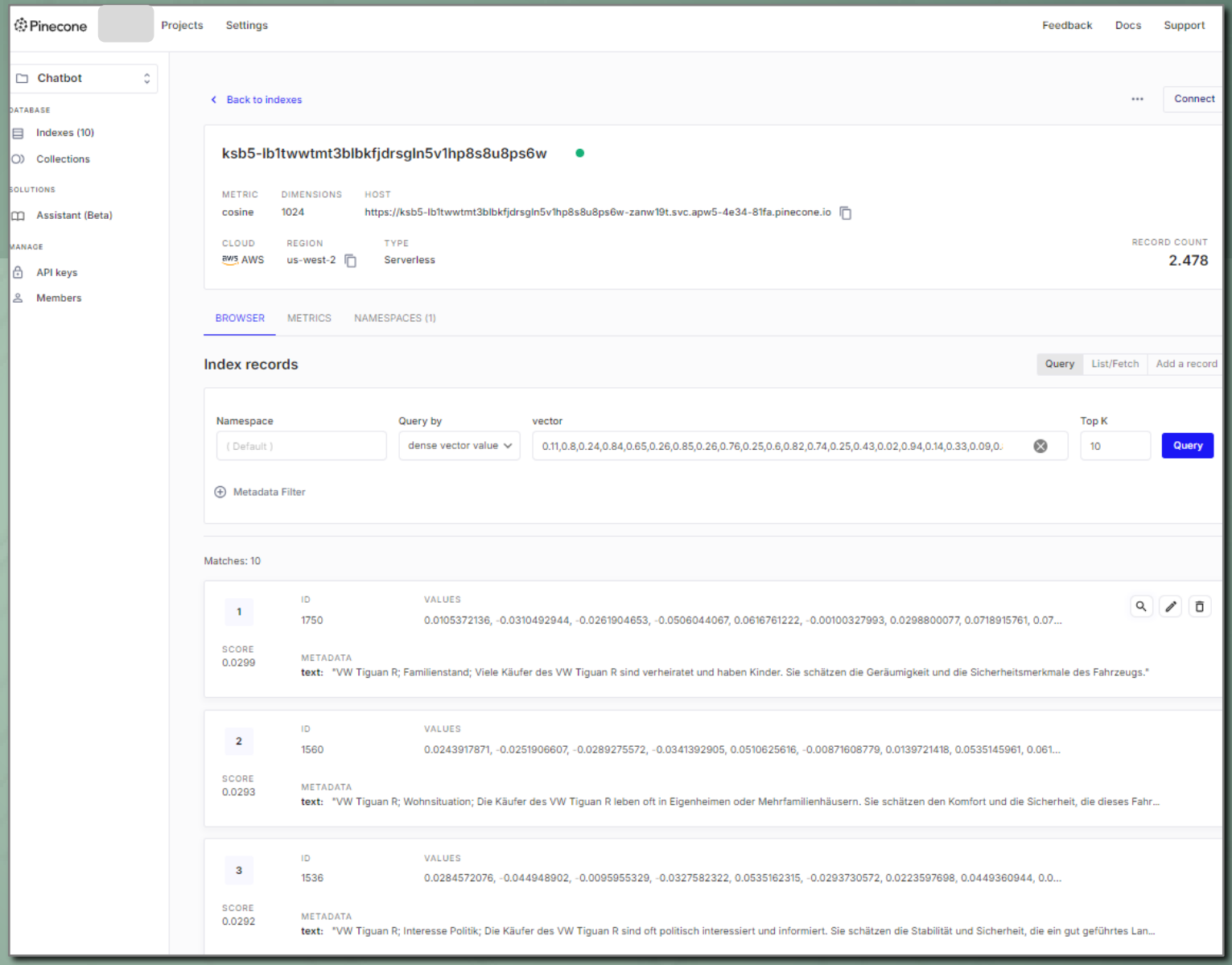
- Ranges from -1 meaning exactly opposite, to 1 meaning exactly the same
- 0 usually indicating independence
- in-between values indicating intermediate similarity or dissimilarity



Documents in vector database

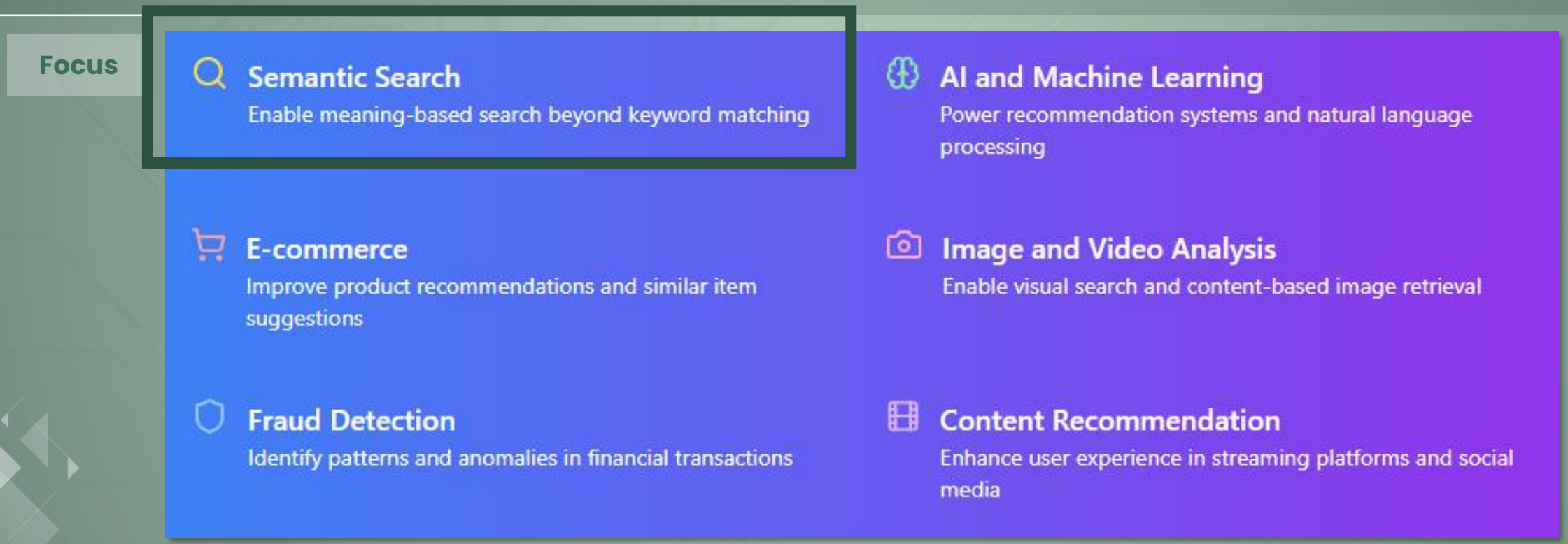
… can be listed by similarity

- Image shows Pinecone vector database management interface.
- Displayed are details of index named "ksb5-lb1twwt …"
- index uses cosine similarity with 1024 dimensions, contains 2,478 records
- Hosted on AWS, us-west-2 region, serverless
- Query results show matches with
  - IDs
  - vector values
  - similarity scores
  - metadata.
- metadata includes German text descriptions related to VW Tiguan R and its buyers.



Typical applications of Vector Databases

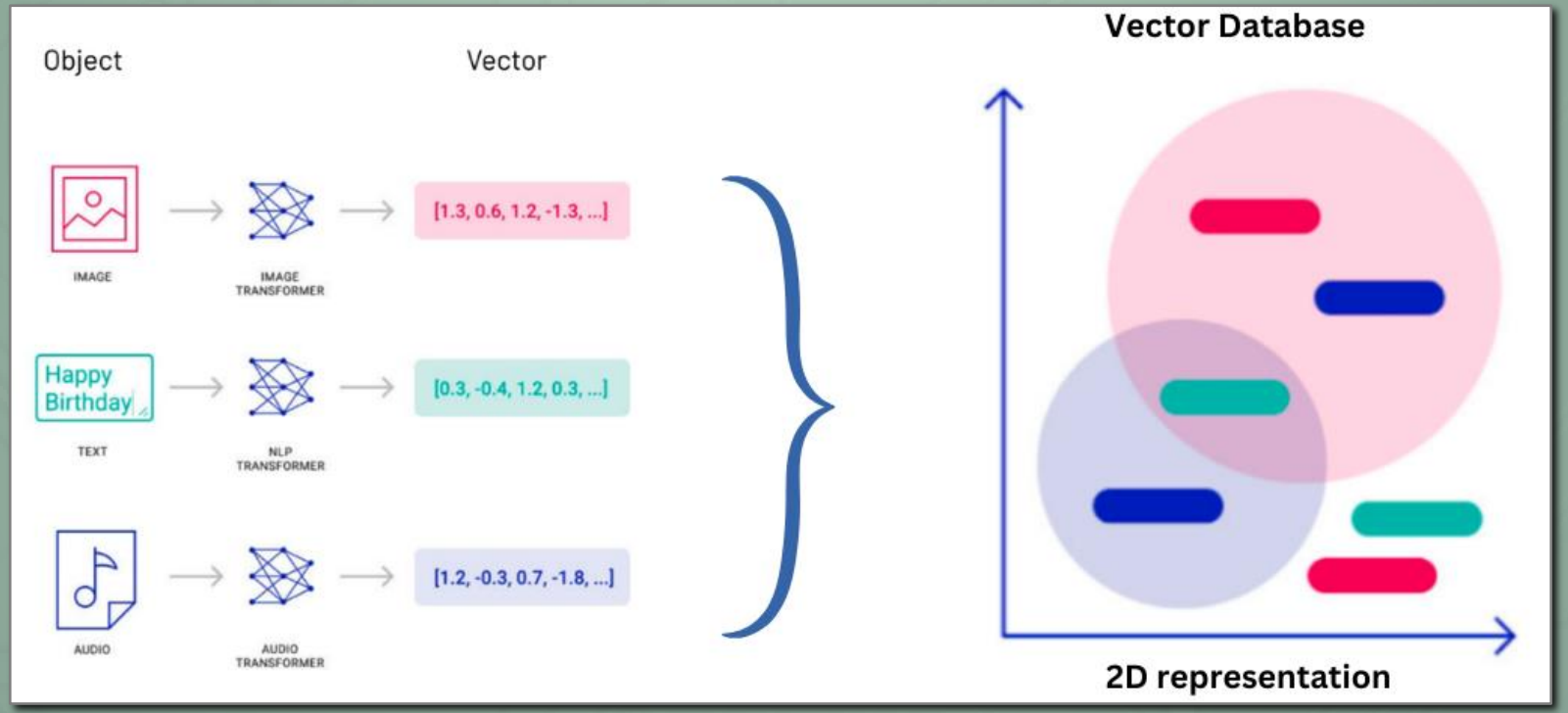
… go beyond semantic search which is used for RAG



https://spotintelligence.com/2023/10/17/semantic-search/
https://www.elastic.co/guide/en/elasticsearch/reference/current/semantic-search.html
https://redis.io/blog/vector-databases-101/

Sentence transformers

… are BERT based models designed to convert sentences, paragraphs, longer texts into fixed-size dense vector representations (embeddings) that capture semantic meaning.



Example from HuggingFace:
https://huggingface.co/intfloat/multilingual-e5-large
intfloat/multilingual-e5-large

https://sbert.net/examples/training/multilingual/README.html
https://huggingface.co/sentence-transformers

Connecting data sources like CSV-, PDF-files

… via Python pipeline to vector database can be tedious, as tables, columns etc. have to be vector embedded in a way that they can be semantically found, ranked etc. afterwards

Ship_name	Cruise_line	Age	Tonnage	passengers	length	cabins	passenger_density	crew
Journey	Azamara	6	30.277	6.94	5.94	3.55	42.64	3.55
Quest	Azamara	6	30.277	6.94	5.94	3.55	42.64	3.55
Celebration	Carnival	26	47.262	14.86	7.22	7.43	31.8	6.7
Conquest	Carnival	11	110	29.74	9.53	14.88	36.99	19.1
Destiny	Carnival	17	101.353	26.42	8.92	13.21	38.36	10
Ecstasy	Carnival	22	70.367	20.52	8.55	10.2	34.29	9.2
Elation	Carnival	15	70.367	20.52	8.55	10.2	34.29	9.2
Fantasy	Carnival	23	70.367	20.56	8.55	10.22	34.23	9.2

Python pipeline

[1.3, 0.6, 1.2, -1.3, ...]

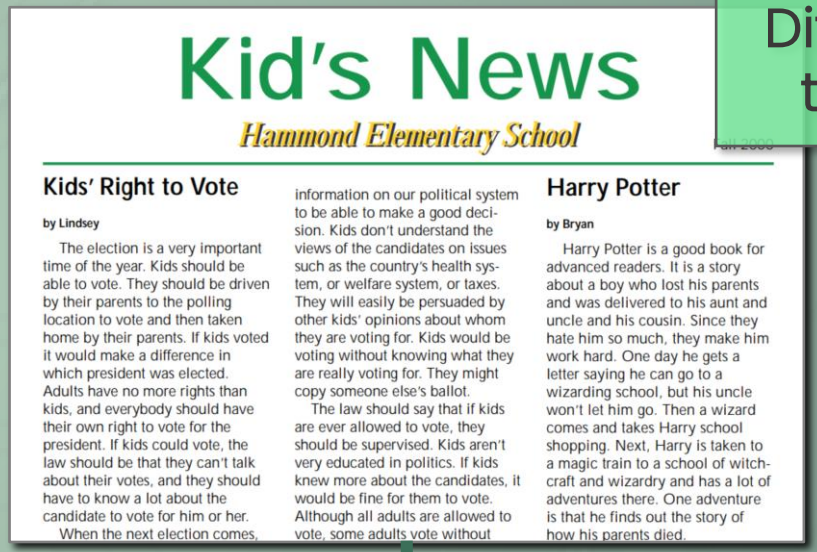
[0.3, -0.4, 1.2, 0.3, ...]

[1.2, -0.3, 0.7, -1.8, ...]

Meta-Tags:

- Rows (e.g. ship name)
- Columns (e.g. feature)

Difficult task!



Kid's News

Hammond Elementary School

Kids' Right to Vote

by Lindsey

The election is a very important time of the year. Kids should be able to vote. They should be driven by their parents to the polling location to vote and then taken home by their parents. If kids voted it would make a difference in which president was elected. Adults have no more rights than kids, and everybody should have their own right to vote for the president. If kids could vote, the law should be that they can't talk about their votes, and they should have to know a lot about the candidate to vote for him or her.

When the next election comes, information on our political system to be able to make a good decision. Kids don't understand the views of the candidates on issues such as the country's health system, or welfare system, or taxes. They will easily be persuaded by other kids' opinions about whom they are voting for. Kids would be voting without knowing what they are really voting for. They might copy someone else's ballot.

The law should say that if kids are ever allowed to vote, they should be supervised. Kids aren't very educated in politics. If kids knew more about the candidates, it would be fine for them to vote. Although all adults are allowed to vote, some adults vote without

Harry Potter

by Bryan

Harry Potter is a good book for advanced readers. It is a story about a boy who lost his parents and was delivered to his aunt and uncle and his cousin. Since they hate him so much, they make him work hard. One day he gets a letter saying he can go to a wizarding school, but his uncle won't let him go. Then a wizard comes and takes Harry school shopping. Next, Harry is taken to a magic train to a school of witchcraft and wizardry and has a lot of adventures there. One adventure is that he finds out the story of how his parents died.

Python pipeline

Content needs to be chunked into small pieces

[1.3, 0.6, 1.2, -1.3, ...]

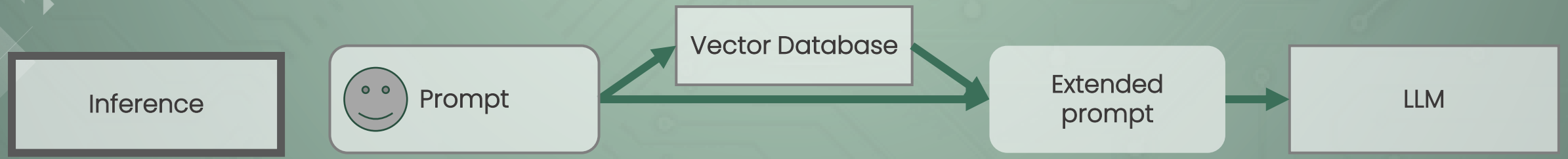
[0.3, -0.4, 1.2, 0.3, ...]

Retrieval augmented prompt

Prompt template is filled with retrieved data from VectorDB and passed to LLM

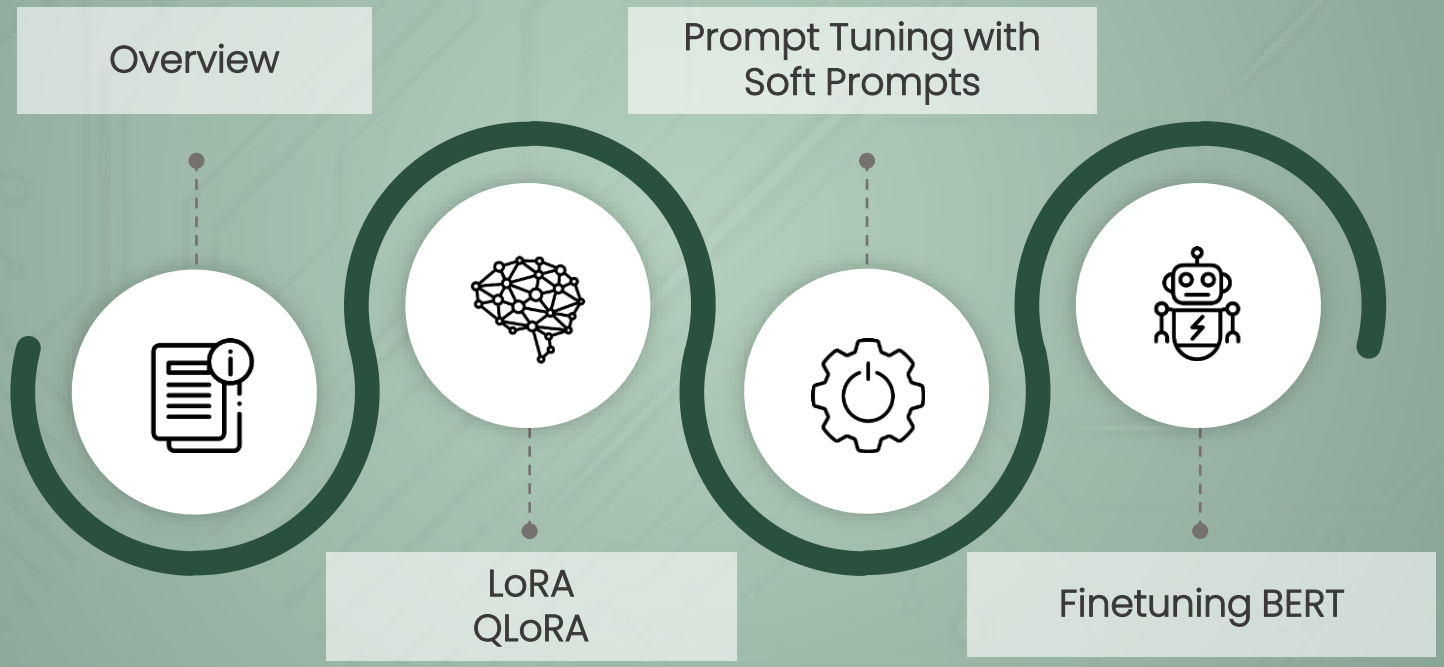
```
# Python example of prompt template

from string import Template

migratory_birds_template = Template("""
Migratory Birds (Zugvögel) and Their Winter Habitats
1. ${species1}: Summers in ${summer1}, winters in ${winter1}
2. ${species2}: Summers in ${summer2}, winters in ${winter2}
3. ${species3}: Summers in ${summer3}, winters in ${winter3}
Interesting fact: ${fact}

Conservation of these habitats is crucial for these birds'
survival.
""")
```

```
# Retrieved information from vector database, sorted by
relevance score

retrieved_info = [
    {
        'species': 'Arctic Tern',
        'summer_habitat': 'Arctic regions',
        'winter_habitat': 'Antarctic coast',
        'fact': 'Makes the longest known migration of any animal,
flying about 70,900 km (44,100 miles) round trip annually.',
        'relevance_score': 0.95
    },
    {
        'species': 'Bar-tailed Godwit',
        'summer_habitat': 'Alaska',
        'winter_habitat': 'New Zealand',
        'fact': 'Can fly over 11,000 km non-stop, the longest known
non-stop flight of any bird.',
        'relevance_score': 0.92
    },
(…)
]
bird_info = {
    'species1': retrieved_info[0]['species'],
    'summer1': retrieved_info[0]['summer_habitat'],
    'winter1': retrieved_info[0]['winter_habitat'],
(…)
}

formatted_prompt =migratory_birds_template.safe_substitute(bird_info)
print(formatted_prompt)
```



Data Augmentation: Parameter efficient Finetuning

… techniques allow for adaptation of large language models to specific tasks, domains while updating only small subset of parameters, significantly reducing computational requirements



Parameter efficient finetuning

… is about adapting LLMs individually, balancing performance, resource efficiency. Pre-trained models can be used in diverse applications without prohibitive costs of full model retraining.

Selective Tuning

Adjusts limited parameters, freezes most

Efficiency, performance

- Reduces resources (fewer GPU, CPU time), enables large model adaptation on consumer hardware
- Comparable to full fine-tuning, especially in low-data scenarios

Retains Knowledge

Prevents catastrophic forgetting of pre-trained information

Techniques

- LoRA, QLoRA
- Prompt Tuning
- Prefix Tuning (omitted)

https://medium.com/aimonks/parameter-efficient-fine-tuning-075954d1db51

LoRA / QLoRA

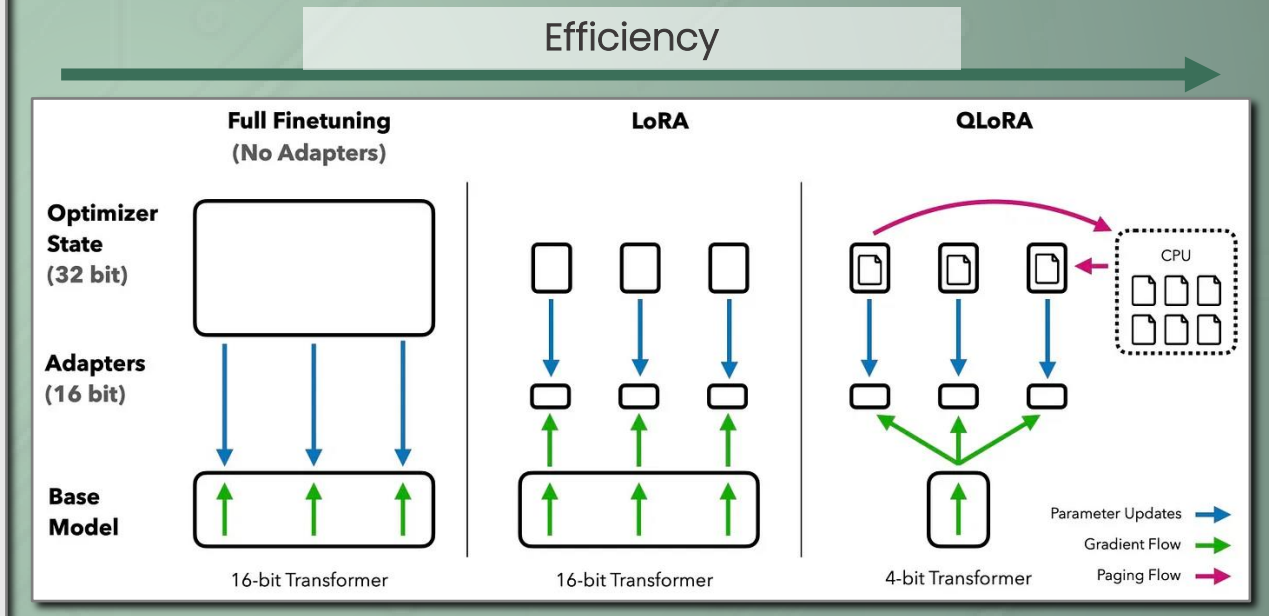
… is parameter-efficient fine-tuning technique that adds trainable low-rank decomposition matrices to the weights of transformer layers in pre-trained language models

LoRA: Low-Rank Adaptation

- Freezes pre-trained model weights
- Injects trainable rank decomposition matrices
- Significantly reduces number of trainable parameters
- Enables fine-tuning on consumer-grade hardware

QLoRA: Quantized Low-Rank Adaptation

- What is quantization:
  - Refers to parameters that are finetuned
  - Converting high-precision numbers (e.g., 32-bit floats) to lower-precision (e.g., 8-bit integers)
  - Reduces the number of bits used to represent each value
- Extends LoRA with 4-bit quantization
- Further reduces memory usage

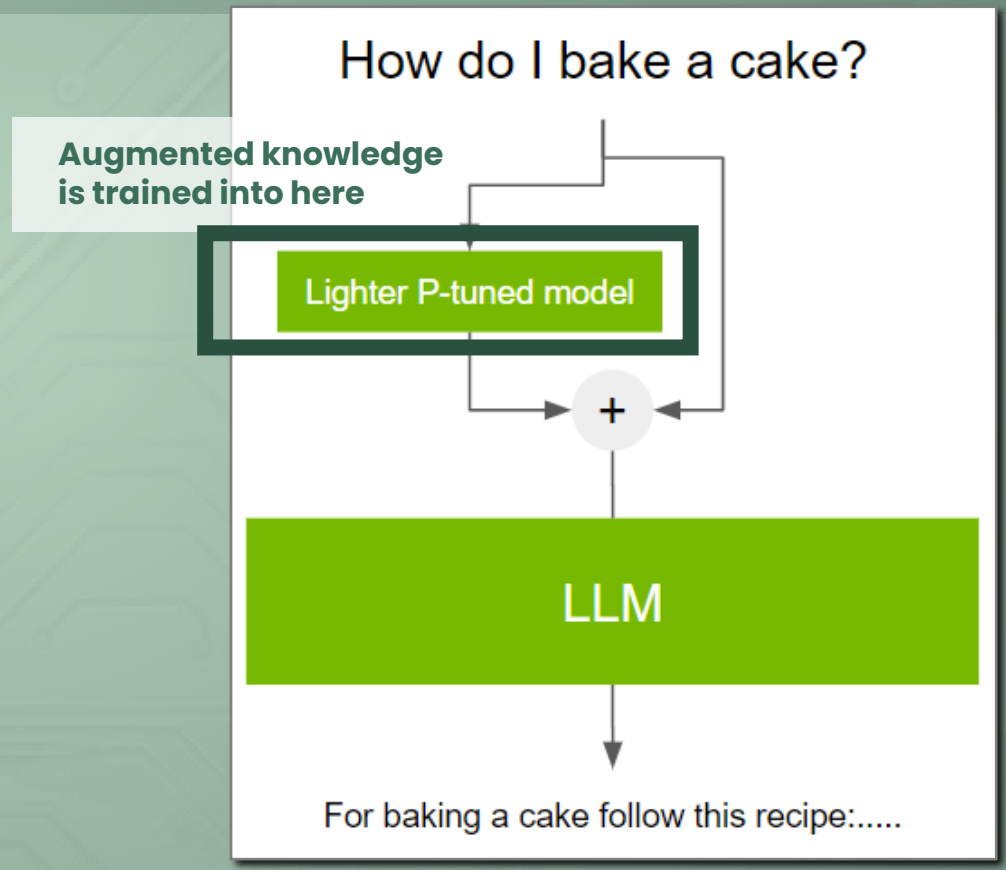


https://towardsdatascience.com/qlora-fine-tune-a-large-language-model-on-your-gpu-27bed5a03e2b
QLORA:Efficient Finetuning of Quantized LLMs, 2023

Prompt Tuning with Soft Prompts

… is fine-tuning technique that prepends learnable continuous vectors (soft prompts) to input of pre-trained LM, allowing task-specific adaptation without modifying original model weights

- Initialization:
  - Start by initializing small set of learnable vectors called soft prompts, which are only parameters to be trained.
  - "soft" prompts designed by AI that outperformed human-engineered "hard" prompts.
- Input Construction:
  For each input text, prepend these soft prompts to input token embeddings before passing them through pre-trained LM.
- Training:
  During training, freeze all model's parameters except for the soft prompts. Optimize these soft prompts to perform desired downstream task (e.g., sentiment analysis).
- Inference:
  After training, use learned soft prompts to guide model in making predictions on new inputs for specific task.

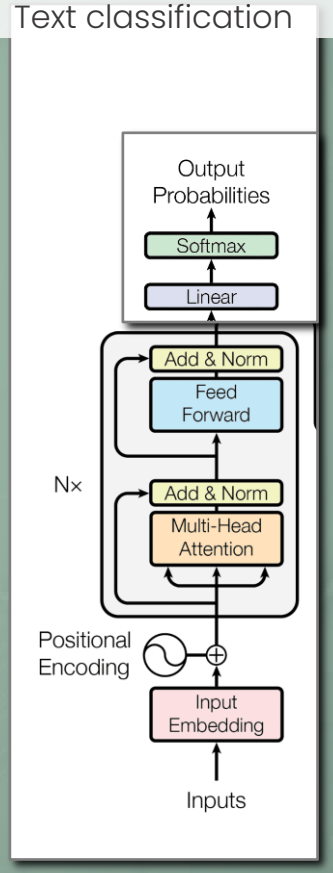


https://cobusgreyling.medium.com/prompt-tuning-hard-prompts-soft-prompts-49740de6c64c
The Power of Scale for Parameter-Efficient Prompt Tuning, 2021

Training & Finetuning BERT for Sentiment Analysis

… Bidirectional Encoder Representations from Transformers (BERT) classifies text data. Due to its realtively small size it is easy to finetune it without sophisticated techniques

Encoder:
- Sentiment Analysis
- Text classification



Training BERT from Scratch (Optional & expensive)

- Corpus Preparation: Gather a large, diverse corpus of text data.
- Pre-training Tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP).
- Computational Requirements:
  High, due to complexity and size of the model
  → better use HuggingFace

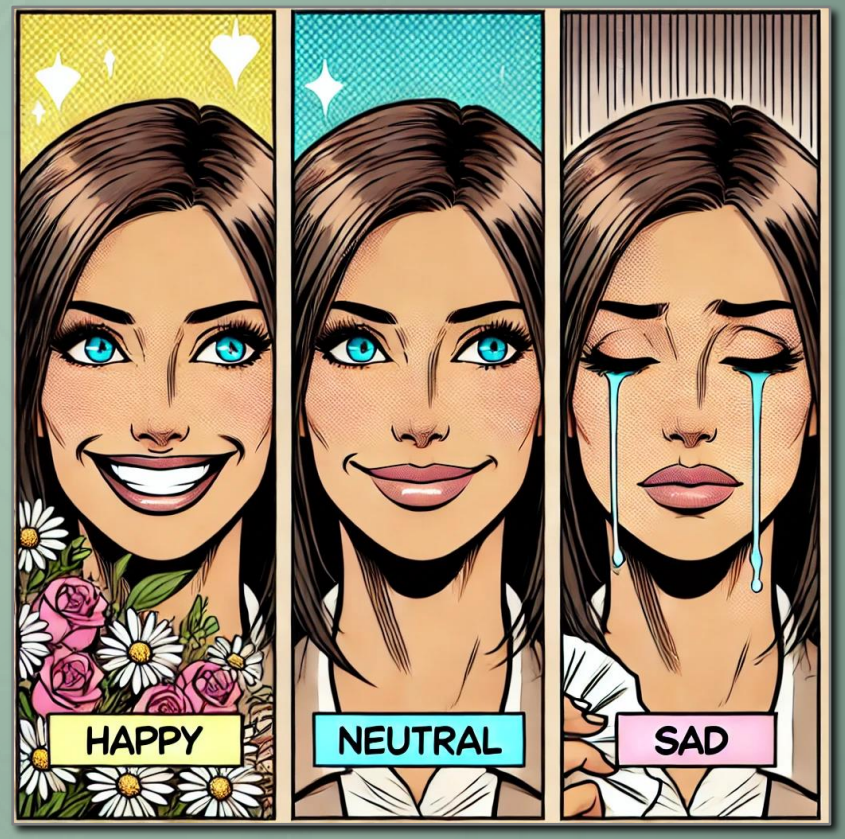
Finetuning Pre-trained BERT

- Objective: Adapt BERT to accurately classify sentiments in text (positive, negative, neutral).
- Dataset:
  Utilize sentiment-labeled dataset specific to the target domain
  (e.g., product reviews, social media posts, movie comments).
- Process:
  - Modify BERT's final layer to output sentiment categories.
  - Train model on sentiment dataset, adjusting parameters for optimal performance.

Training & Finetuning BERT for Sentiment Analysis

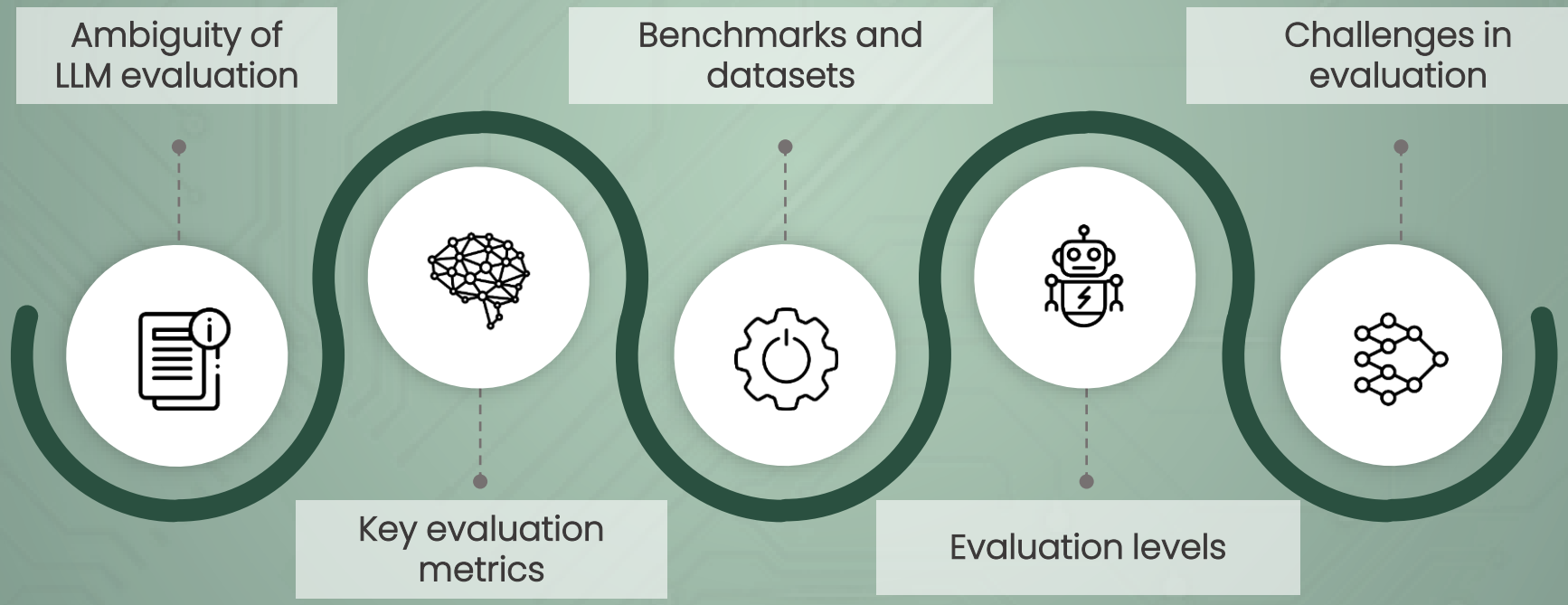
Example of labelled dataset of cinema comments. At training time BERT parameters are adjusted in order to imitating labels. At inference time new comments are classified accordingly

Nr	Comment	Label
1	The cinematography was absolutely stunning! Every shot was like a painting, and the lighting perfectly captured the mood of each scene.	Positive
2	The movie was okay, but the plot felt a bit predictable. The acting was decent, but nothing stood out.	Neutral
3	The pacing of the film was so slow that it became unbearable to watch. I nearly fell asleep halfway through.	Negative
4	The soundtrack was amazing and really enhanced the emotional depth of the film. I found myself getting lost in the music.	Positive
5	The special effects were impressive, but they couldn't make up for the lack of character development and weak story.	Negative
6	The movie had its moments, but it didn't leave a lasting impression. It was just another average film.	Neutral



Evaluating Large Language Models

… involves assessing performance across diverse tasks, like natural language understanding, generation, reasoning, while considering accuracy, fluency, coherence, bias, ethics



What do AI agents and driving have in common?

For both, the question arises: how many per mille is still okay?
(blood alcohol content vs. LLM generation error)

Language Models inherently make mistakes due to

- Lack of necessary domain data
- Hallucinations:
  Even strong models with proper prompts may hallucinate on per mille scale

Therefore quality must be measurable

Problem

- Evaluation is often ambiguous because quality measure depends on given task
- Hence, quality measures can only refer to particular task

Key evaluation metrics

Precision: How many generated words/phrases are correct or relevant?

Recall: How many correct or relevant elements are captured?

Metric	Description	Use Case	Pros	Cons
Perplexity	▪ measures how well probability model predicts sample ▪ indicating model's uncertainty in its predictions ▪ The lower the better	▪ Text generation ▪ Speech recognition	▪ Good for comparing models ▪ Intrinsic measure of model quality, does not require human in the loop	▪ Doesn't capture semantic meaning or coherence of generated text ▪ only the model's confidence in its predictions
BLEU Score (Bilingual Evaluation Understudy)	▪ Algorithm for evaluating quality of machine-translated text ▪ compares n-grams of candidate translation with n-grams of reference translation(s) ▪ range from 0 to 1, where 1 indicates perfect translation	▪ Machine translation	▪ Widely used in translation tasks ▪ Correlates with human judgements	▪ BLEU rewards translations that are accurate with the words they use, but may not capture all information from the source (Precision oriented) ▪ Doesn't account for meaning or fluency
ROUGE Score (Recall-Oriented Understudy for Gisting Evaluation)	▪ Set of metrics for evaluating automatic summarization and machine translation ▪ Measures overlap between generated and reference texts	▪ Text summarization ▪ Text generation	▪ Multiple variants for different aspects ▪ Considers both precision and recall	▪ Relies on exact word matches ▪ May not capture paraphrasing well
F1 Score	Harmonic mean of precision and recall	▪ Classification tasks ▪ information retrieval	▪ Balances precision and recall ▪ Good for imbalanced datasets	▪ May not be suitable for all NLP tasks, focus on classification ▪ Doesn't account for true negatives

https://angelina-yang.medium.com/is-perplexity-bigger-the-better-or-smaller-the-better-94e11da4d4bc

Benchmarks and datasets

… are standardized tools for evaluating and comparing LLMs' performance across diverse language tasks, from basic comprehension to complex reasoning.

Benchmark, Dataset	Description	Tasks	Key Features	Challenges
GLUE	General Language Understanding Evaluation benchmark	▪ Sentence similarity ▪ Natural language inference ▪ Sentiment analysis	▪ Diverse set of tasks ▪ Well-established in NLP community	▪ Some tasks considered too easy for modern LLMs ▪ Limited to English language
SuperGLUE	More challenging successor to GLUE	▪ Question answering ▪ Coreference resolution ▪ Causal reasoning	▪ More difficult tasks than GLUE ▪ Closer to human-level performance benchmarks	▪ Still primarily focused on English ▪ May not fully capture capabilities of most advanced LLMs
SQuAD	Stanford Question Answering Dataset	▪ Reading comprehension ▪ Question answering	▪ Large-scale dataset ▪ Real-world questions and contexts	▪ Answers limited to spans within given text ▪ May not test deeper reasoning capabilities
LAMBADA	Language Model Benchmark on cloze-style task	▪ Word prediction ▪ Language modeling	▪ Tests broad context understanding ▪ Requires common sense reasoning	▪ Focus on specific last-word prediction task ▪ May not cover full range of language understanding

Evaluation levels

… are ranging from zero-shot, few-shot to fine-tuning, balancing generalization and task-specific capabilities.

Approach	Description	Advantages	Limitations	Use Cases
Zero-shot	Model performs task without any task-specific examples	▪ No additional training required ▪ Tests model's general knowledge ▪ Quick to implement	▪ May not perform well on specialized tasks ▪ Highly dependent on prompt engineering	▪ General language understanding tasks ▪ Quick prototyping ▪ Assessing model's base capabilities
Few-shot	Model is given a few examples of the task before performing it	▪ Better performance than zero-shot ▪ No need for model retraining ▪ Flexible for different tasks	▪ Performance can vary based on example selection ▪ Limited by context window size	▪ Task adaptation with limited data ▪ Evaluating model's ability to learn from examples
Fine-tuning	Model is retrained on task-specific data	▪ Best performance on specific tasks ▪ Can learn task-specific knowledge ▪ Adaptable to specialized domains	▪ Requires significant computational resources ▪ Needs substantial task-specific data ▪ Risk of overfitting to specific task	▪ Specialized applications ▪ When high performance on specific tasks is crucial ▪ Domains with ample task-specific data

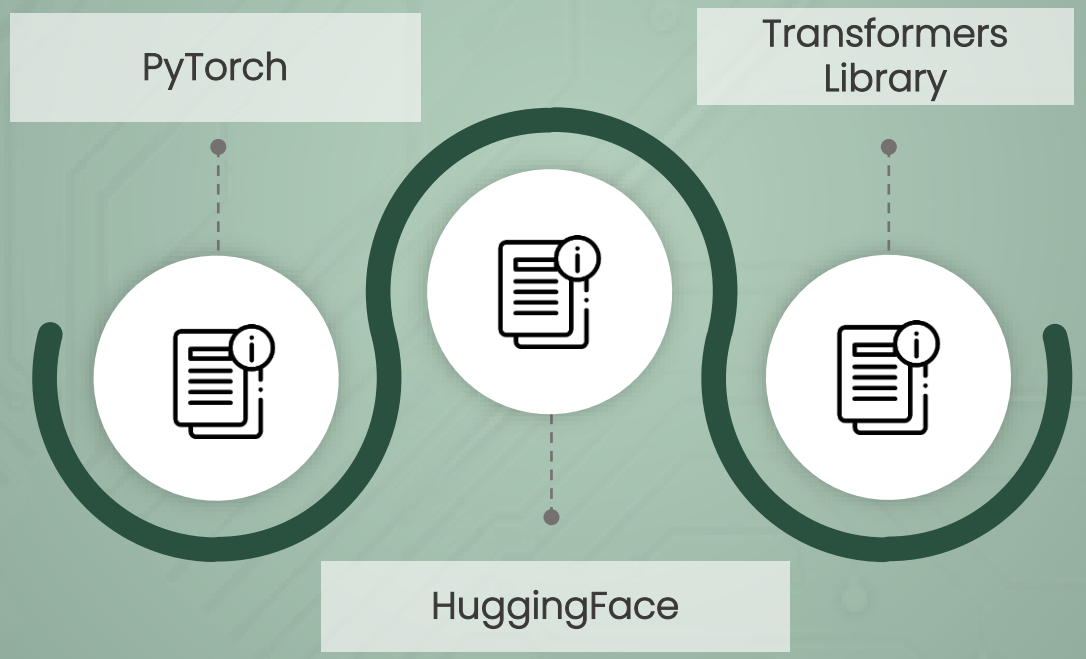
Challenges in evaluating Language Models

… range from addressing bias, contextual understanding to ensuring scalability, ethical compliance, requiring sophisticated and responsible assessment methods.

Challenge Area	Key Issues
Bias and Fairness	▪ Detecting and quantifying inherent biases ▪ Ensuring fairness across diverse groups ▪ Balancing performance with ethical considerations
Contextual Understanding	▪ Evaluating nuanced language comprehension ▪ Assessing performance in domain-specific contexts ▪ Measuring ability to handle ambiguity
Robustness and Generalization	▪ Testing performance on out-of-distribution data ▪ Evaluating consistency across various inputs ▪ Assessing adaptability to new tasks or domains
Scalability of Evaluation	▪ Managing computational resources for large-scale testing ▪ Developing efficient evaluation pipelines ▪ Balancing thoroughness with practicality
Ethical and Safety Considerations	▪ Assessing potential for misuse or harmful outputs ▪ Evaluating alignment with human values ▪ Measuring truthfulness and factual accuracy ▪ Ensuring privacy and data protection in evaluation

Programming Tools

...involves the organization, summarization, and visualization of data. It provides simple summaries about the sample and the measures.



What is PyTorch?

It is an open-source deep learning framework developed by Facebook's AI Research lab.

Key Features:

- Dynamic computational graph, offering flexibility in model design.
- Native support for GPU acceleration, enhancing performance.
- Intuitive tensor library similar to NumPy but with GPU support.

Applications:

- Research prototyping, offering ease and speed.
- Building deep learning models, from neural networks to complex architectures.

Community & Ecosystem:

- Active community contributing to its growth.
- Rich ecosystem with pre-trained models, tools, and libraries.

TensorFlow, PyTorch and Keras

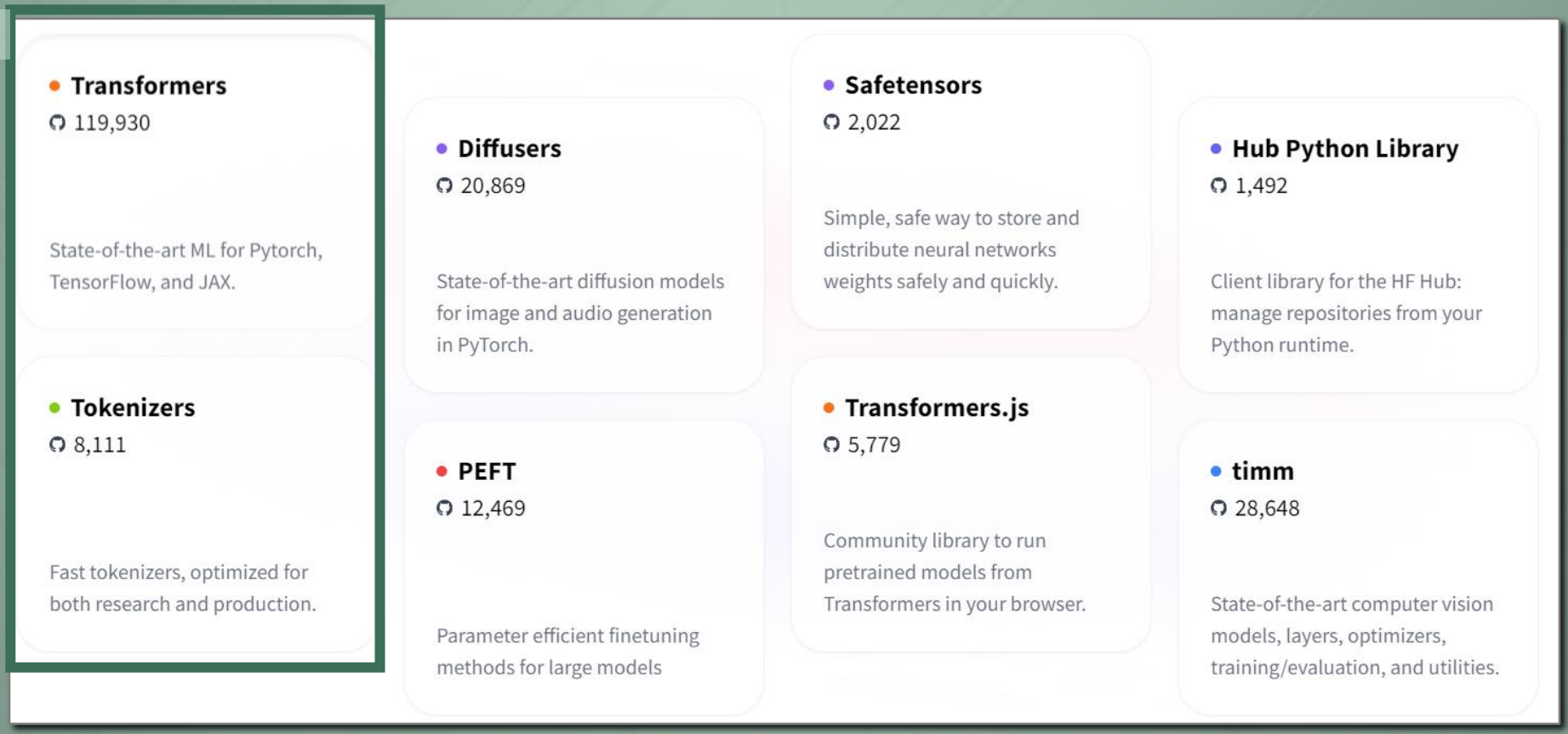
... are popular open-source libraries used for machine learning and deep learning.
Keras is API built on top of TensorFlow, PyTorch and others

	1 TensorFlow	2 PyTorch (Focus)	3 Keras
DEVELOPED BY	Google Brain team	Facebook's AI Research lab	François Chollet
EASE OF USE	Steeper learning curve	Easier to learn	Extremely user-friendly
FLEXIBILITY / CUSTOMIZATION	Very flexible	Very flexible	Limited flexibility
PERFORMANCE / SPEED	Excellent performance	Excellent performance	Performance is dependent on the backend used
USE CASES	Suitable for both production and research, especially in large-scale systems	Preferred for research, rapid prototyping, and academic purposes	Best for quick development of standard neural networks, less suited for research

HuggingFace

… offers 100s of 1000s NPL models, datasets, tokenizers, etc. and has vibrant open source community of AI researchers, engineers, enthusiasts contributing to AI advancement

Focus



HuggingFace: Transformers Library

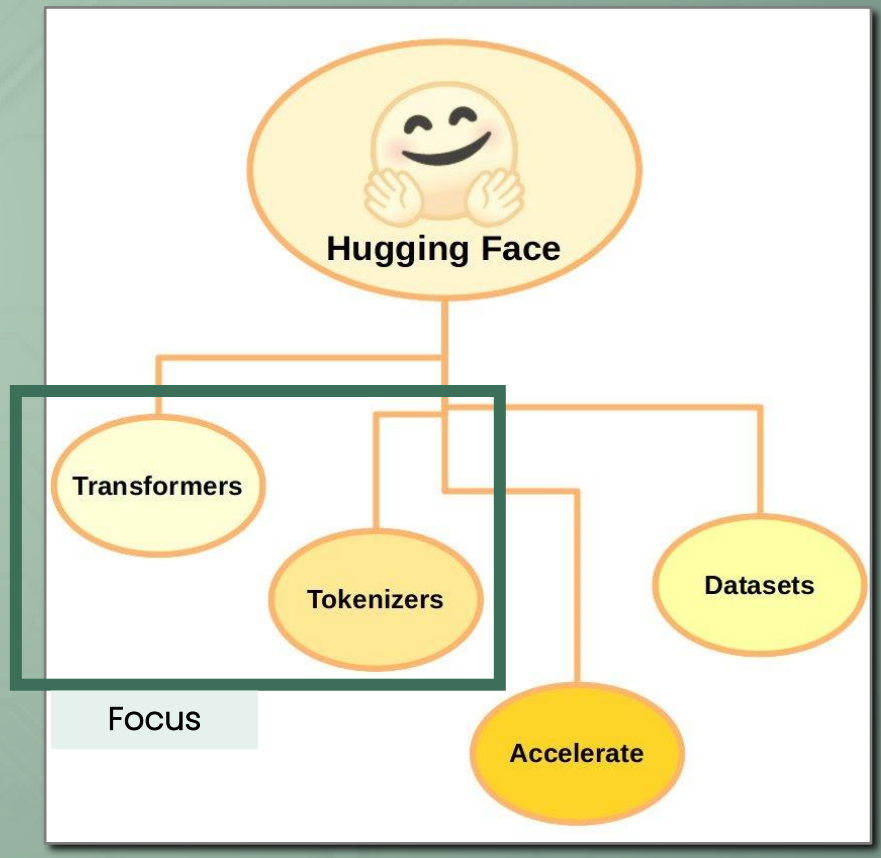
… provides tools to easily download, train state-of-the-art pretrained models. Using pretrained models can reduce time and resources required to train a model from scratch

Key Features:

- Comprehensive and Modular:
  Offers models like BERT, GPT, T5, etc., with simple interface for NLP tasks.
- Ease of Use:
  Enables easy model downloading, training, deployment with minimal code.
- Community-Driven:
  Actively supported by vast community, ensuring continuous updates, enhancements.

Applications:

- Research prototyping, offering ease and speed.
- Building deep learning models, from neural networks to complex architectures.



Examples with Python Code

… in GitHub Repo

	Tools	Training	Inference
Sentiments of movie comments	BERT	x	x
Data Retrieval	LLM		x
RAG-Pipeline CSV	VectorDB, LLM	x	x
RAG-Pipeline PDF	VectorDB, LLM	x	x

Links

i.e. sources for self-learning

	Title	Link
Basics of Natural Language Processing	An Introduction to Bag of Words (BoW) | What is Bag of Words?	An Introduction to Bag of Words (BoW) | What is Bag of Words?
	Understanding TF-IDF: A Traditional Approach to Feature Extraction in NLP	https://towardsdatascience.com/understanding-tf-idf-a-traditional-approach-to-feature-extraction-in-nlp-a5bfbe04723f
	Word Embedding and Word2Vec, Clearly Explained!!!	https://www.youtube.com/watch?v=viZrOnJclY0
	Word2Vec Explained	https://towardsdatascience.com/word2vec-explained-49c52b4ccb71
	Word2vec : NLP & Word Embedding	https://datascientest.com/de/word2vec
	Stanford XCS224U: NLU I Intro & Evolution of Natural Language Understanding	https://www.youtube.com/watch?v=K_Dh0Sxujuc&list=PLoROMvodv4rOwvldxftJTmoR3kRcWkJBp
	Introduction | NLP Tutorial For Beginners In Python	https://www.youtube.com/watch?v=R-AG4-qZs1A&list=PLeo1K3hjS3uuvuAXhYjV2lMEShq2UYSwX

Links

i.e. sources for self-learning

	Title	Link
Transformer based NLP	How GPT Works by Archerman Capital	https://www.youtube.com/watch?v=Mhy7I4E6eXs
	The math behind Attention: Keys, Queries, and Values matrices	https://www.youtube.com/watch?v=UPtG_38Oq8o
	The Attention Mechanism in Large Language Models	https://www.youtube.com/watch?v=OxCpWwDCDFQ
	Let's build GPT: from scratch, in code, spelled out.	https://www.youtube.com/watch?v=kCc8FmEb1nY
	State of GPT | BRK216HFS	https://www.youtube.com/watch?v=bZQun8Y4L2A
	Stanford CS224N: Natural Language Processing with Deep Learning	https://www.youtube.com/playlist?list=PLoROMvodv4rMFqRtEuo6SGjY4XbRIVRd4
	Data Science: Transformers for Natural Language Processing	https://www.udemy.com/course/data-science-transformers-nlp/learn/lecture/32714220
	Awesome-LLM-KG	https://github.com/RManLuo/Awesome-LLM-KG
	Transformers in Action: Attention Is All You Need	https://towardsdatascience.com/transformers-in-action-attention-is-all-you-need-ac10338a023a
	Attention Is All You Need	https://arxiv.org/abs/1706.03762
	BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding	https://arxiv.org/abs/1810.04805
	What exactly happens when we fine-tune BERT?	https://towardsdatascience.com/what-exactly-happens-when-we-fine-tune-bert-f5dc32885d76
	Sentiment Analysis in 10 Minutes with BERT and TensorFlow	https://towardsdatascience.com/sentiment-analysis-in-10-minutes-with-bert-and-hugging-face-294e8a04b671

DATA SCIENCE.
INSTITUTE
by Fabian Rappert

About me

Dr. Harald Stein

- Data Scientist ~ 8 years experience
- Algotrader ~ 4 years experience
- Ph.D. in Economics, Game Theory

- LinkedIn: https://www.linkedin.com/in/harald-stein-phd-1648b51a
- ResearchGate: https://www.researchgate.net/profile/Harald-Stein