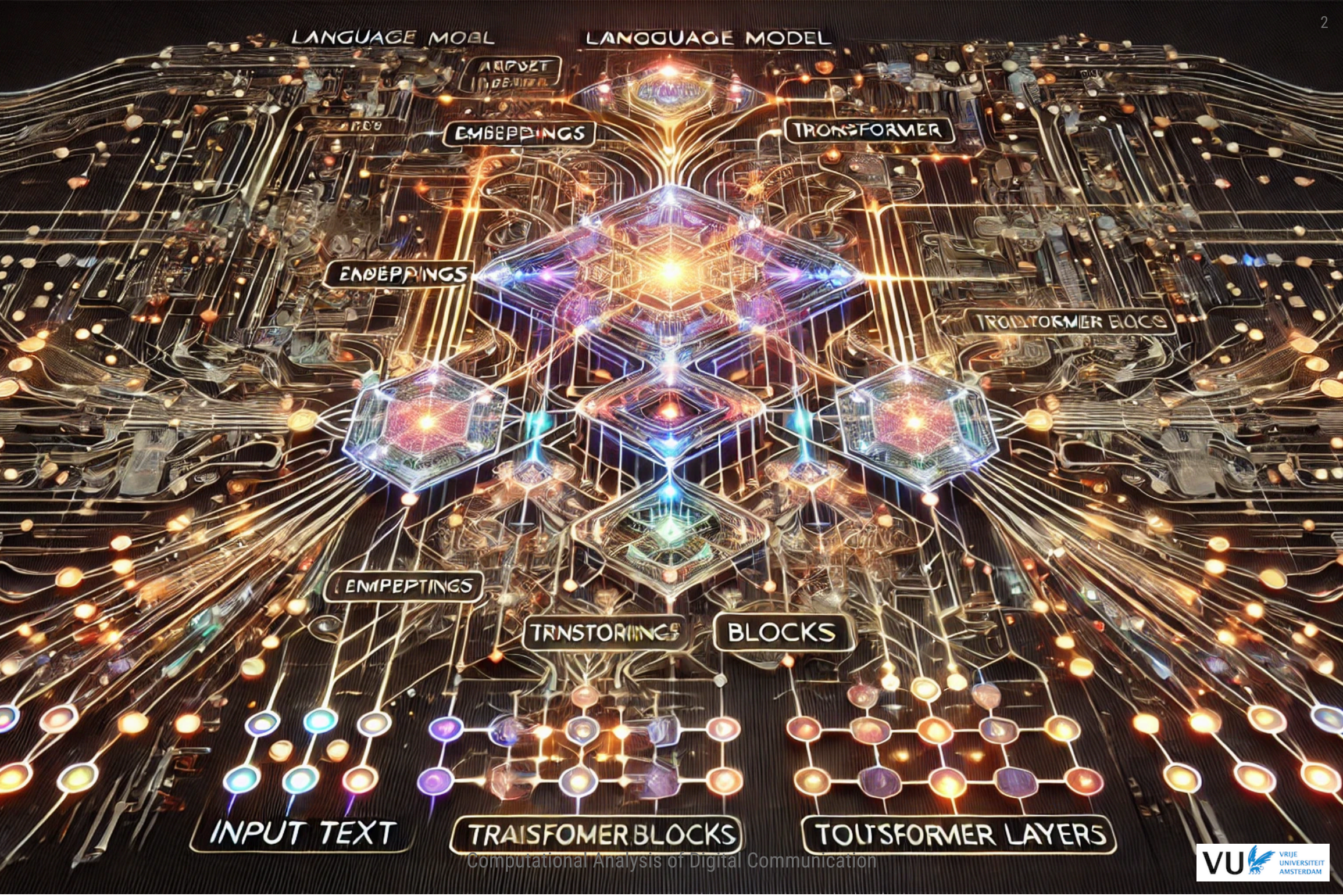
Transformers and Large Language Models

Week 4: Bert, Llama, GPT, and Co

prof. dr. Wouter van Atteveldt & dr. Philipp K. Masur


LANGUAGE MODEL
LANGUAGE MODEL
EMBEDDINGS
TRANSFORMER
EMBEDDINGS
BLOCKS
INPUT TEXT
TRANSFORMER BLOCKS
TRANSFORMER LAYERS

A LOOK BACK AT THE CHRONOLOGY OF NLP

1763

BAYES' THEOREM

- Reverend Thomas Bayes's work is published two years after his death
- It presents major underpinnings of Bayes theorem that is still used in Machine Learning (and Bayesian Statistics) today

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Naïve Bayes

1842

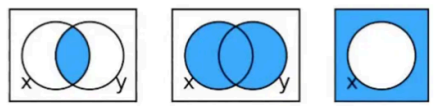
FIRST ALGORITHM?

- Ada Lovelace develops the first algorithm "Note G"
- She is widely seen as the first programmer
- She suggested that a machine could take advantage of computation in order to deal with the realms of music, graphics, and language

1854

BOOLEAN LOGIC

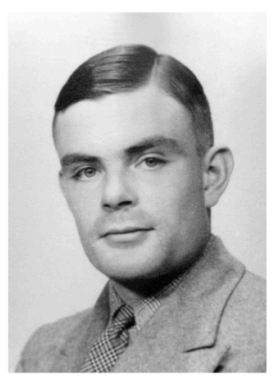
- George Boole invents the Boolean Algebra
- It has been fundamental in the development of digital electronics, and is provided for in all modern programming languages



1934

MACHINES THAT LEARN?

- Alan Turing proposes a machine that can learn
- The Universal Turing Machine is an abstract computing machine consisting of a limitless memory and a scanner that moves back and forth through the memory, symbol by symbol, reading what it finds and writing further symbols



A LOOK BACK AT THE CHRONOLOGY OF NLP

1949 — WEAVER'S MEMORANDUM

- most influential publication in the early days of machine translation
- formulated goals and methods before most people had any idea of what computers might be capable of
- direct stimulus for the beginnings of research into natural language processing

1957 — GRAMMAR THEORIES

- early work in MT was very simple: dictionary-lookup of appropriate words for translation
- Noam Chomsky developed a rule based-system of syntactic structures
- brought insight into how mainstream linguistics could help machine translation

Dictionary Approaches

1960s — ELIZA, ALPAC REPORT & AI WINTER

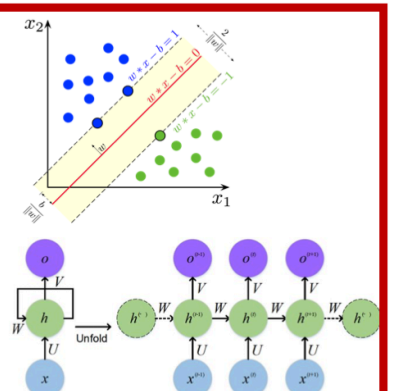
- The most famous prototype is perhaps ELIZA, designed to replicate the conversation between a psychologist and a patient
- The report by the Automatic Language Processing Advisory Committee (ALPAC), a committee established in 1964 by the United States, led to significant cuts in funding

1970s-80s — CONCEPTUAL ONTOLOGIES

- The 1970s brought new ideas into NLP
- Development of conceptual ontologies, i.e. sets of concepts and categories that shows their properties and the relations between them
- structured real-world information into computer-understandable data

1990 — STATISTICAL MODELS

- statistical models came as a revolution in NLP (SVM, regression-based approaches)
- replacing most natural language processing systems based on complex sets of hand-written rules
- Recurrent neural networks and long short-term memory networks



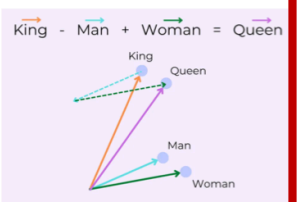
Classic Machine Learning

WHAT WE FOCUS ON TODAY…

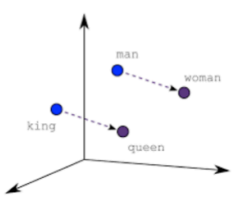
2000s

LANGUAGE MODELING, WORD EMBEDDINGS

- neural networks are used for learning the distribution of language words and predicting the next words in a text given the previous ones
- Start of representing words with dense vectors of numbers
- 2012: First unsupervised neural network by Google



2013



WORD2VEC

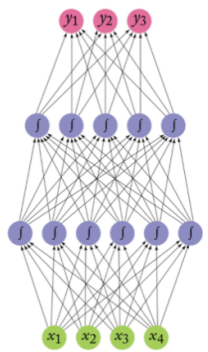
- first algorithm capable of learning word embedding efficiently
- possible to do mathematical operations with them that represent semantic operations
- using pre-trained embeddings as features for ML models improves performance in downstream tasks

Word-Embeddings

2014-2016

LSTM RNNs, CNNs, ENCODER-DECODER

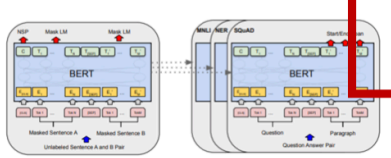
- Most use models are long short-term memory recurrent neural networks, and convolutional neural networks
- sequence-to-sequence learning, a general end-to-end approach for mapping one sequence to another using a neural network
- the first encodes the entire input into a vector, the second decodes the output from the vector



2017

ATTENTION AND TRANSFORMERS

- Introduction of the principle of attention
- Rise of Transformers, which are still used heavily today
- First generative pre-trained transformer by OpenAI in 2018!



Transformer

2020s

Large Language Models

PRE-TRAINED MODELS AND LARGE LANGUAGE MODELS

- Transformers were trained in a self-supervised way on large data sets to become "language models"
- These models can then be reused for different task after only small adaption traning (fine-tuning)

2022

GPT, BART and Co

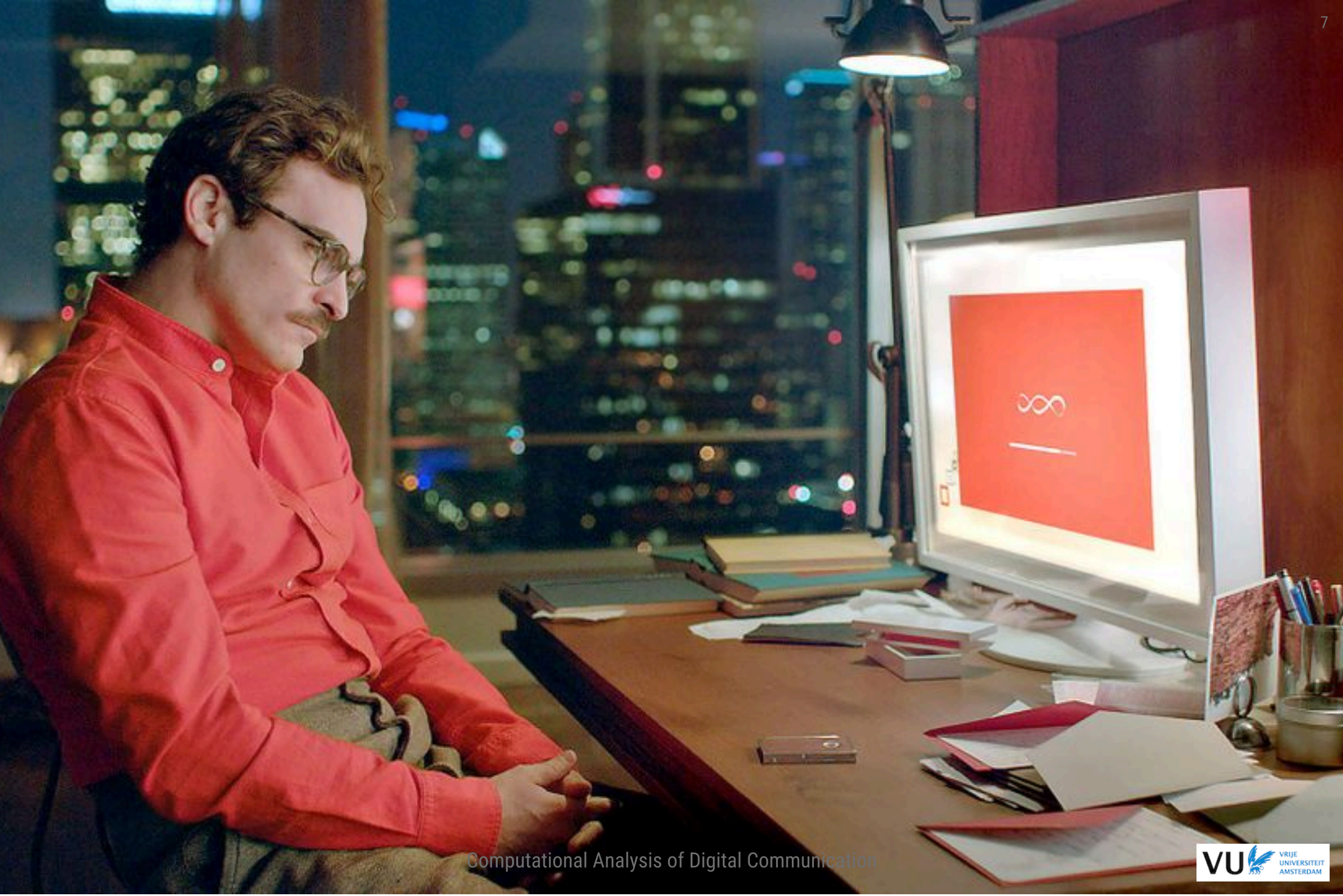
- Increasingly larger language models that are trained on the "Internet"
- November 2022: Launch of chatGPT and wide-spread use of LLMs
- AI revolution?

ChatGPT

THE CLASSIC MACHINE LEARNING APPROACH

- A lot of different steps…
- Training takes a lot of data and time… and performance was still not fantastic

```r
# Get data
science_data <- read_csv("data/science.csv") |>  filter(label != "biology" & label != "finance" & label != "mathematics")

# Create test and train data sets
split <- initial_split(science_data, prop = .50)

# Feature engineering
rec <- recipe(sentiment ~ lemmata, data = science_data) |>
  step_tokenize(lemmata) |>
  step_tf(all_predictors())  |>
  step_normalize(all_predictors())

# Setup algorithm/model
mlp_spec <- mlp(epochs = 600, hidden_units = c(6),
                penalty = 0.01, learn_rate = 0.2) |>
  set_engine("brulee") |>
  set_mode("classification")

# Create workflow
mlp_workflow <- workflow() |>
  add_recipe(rec) |>
  add_model(mlp_spec)

# Fit model
m_mlp <- fit(mlp_workflow, data = training(split))
```



WHAT IF THINGS WERE EASIER?

- Wouldn't it be great if we would not have to wrangle with the data, not engage in any text preprocessing, and simply let the "computer" figure this out?
- In fact, aren't we living in times were we can simply ask the computer, similar as Theodore in the movie "Her"?

```r
library(tidyllm)
library(glue)

glue('Classify this scientific abstract: {abstract}

     Pick one of the following fields.
     Provide only the name of the field:

     Physics
     Computer Science
     Statistics',
     abstract = science_data$text[2]) |>
  llm_message() |>
  chat(ollama(.model = "llama3", .temperature = 0))
```

```
Message History:
system:
You are a helpful assistant
----------------------------------------------------------------
user:
Classify this scientific abstract: Rotation Invariance
Neural Network Rotation invariance and translation
invariance have great values in image
recognition tasks. In this paper, we bring a new
architecture in convolutional
neural network (CNN) named cyclic convolutional layer to
achieve rotation
invariance in 2-D symbol recognition. We can also get the
position and
orientation of the 2-D symbol by the network to achieve
detection purpose for
multiple non-overlap target. Last but not least, this
architecture can achieve
one-shot learning in some cases using those invariance.


Pick one of the following fields.
Provide only the name of the field:

Physics
```

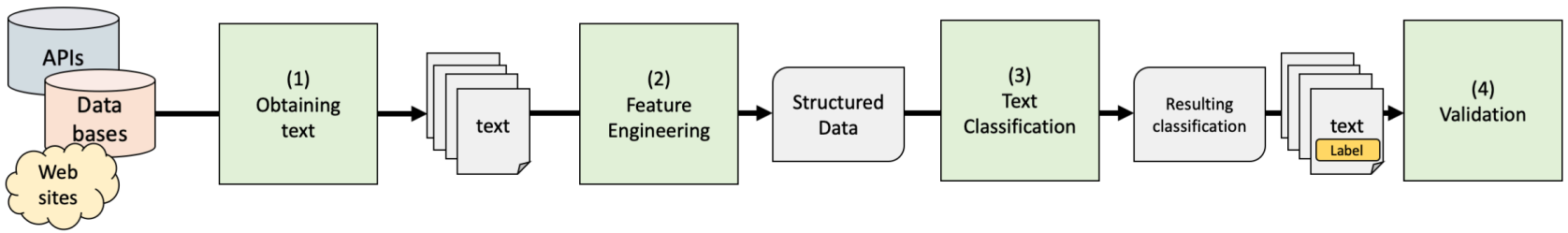
```
Computer Science
Statistics
------------------------------------------------------------
assistant:
```

CONTENT OF THIS LECTURE

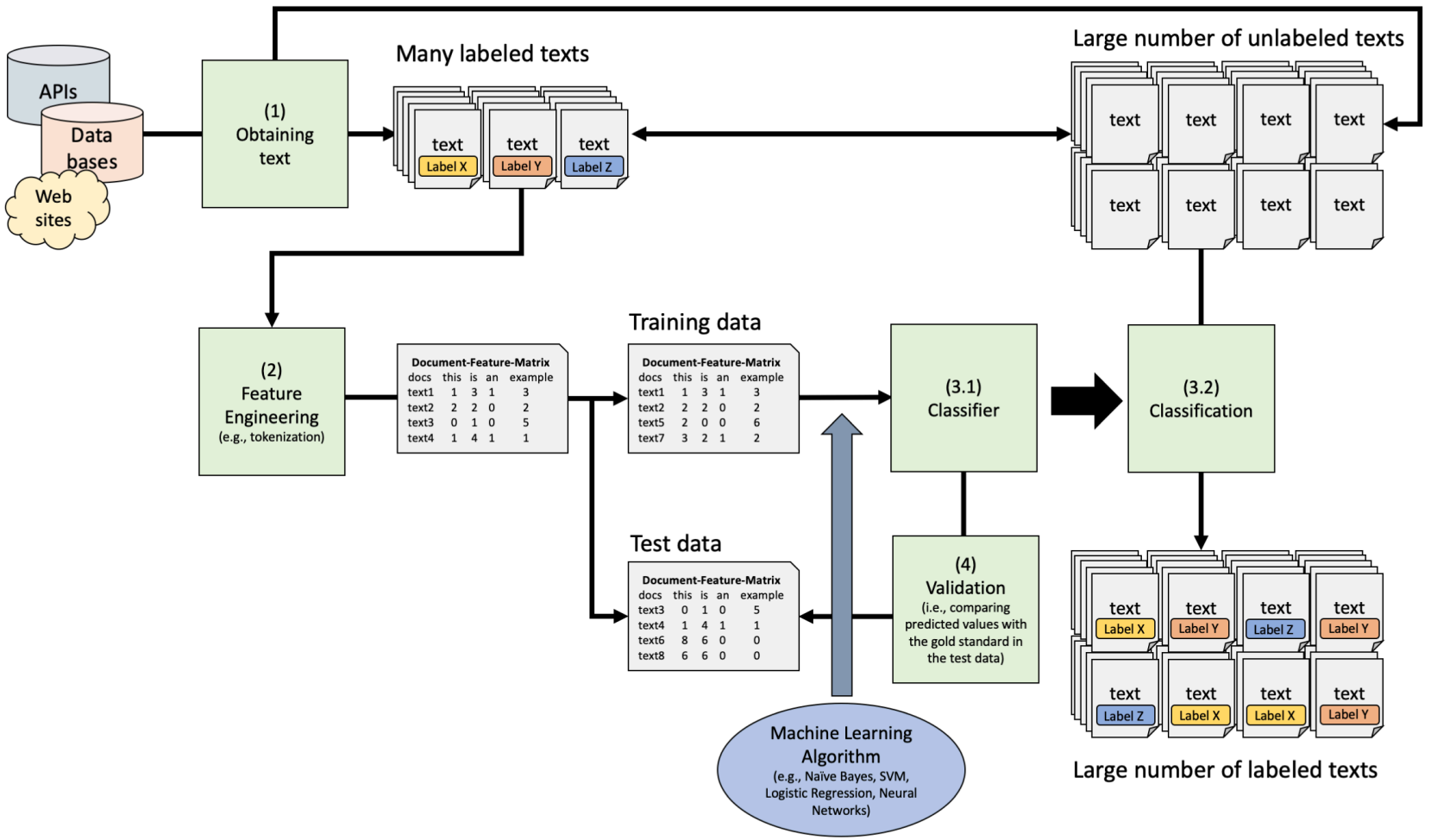
1. Machine Learning vs. Deep Learning
   1.1. Reminder: Text Classification Pipeline
   1.2. How did the Field move on?
2. The Rise of Transformers and Transfer Learning
   3.1. Overview
   3.2. Transfer Learning
   3.3. Architecture of the Transformer Model
   3.4. The Transformer Text Classification Pipeline Using BERT
3. Large Language Models: BERT, Llama, GPT and Co
   3.1. What are Large Language Models and Generative AI?
   3.2. General Idea: Next-Token-Prediction
   3.3. A Peek into the Architecture of GPT
4. Using LLMs for Text Classification
   4.1. Zero-Shot, One-Short, and Few-Shot Classification
   4.2. Text Classification Using Llama and GPT
   4.3. Validation, validation, validation!
   4.4. Examples in the Literature
5. Summary and conclusion
   5.1. State-of-the-Art in Classification
   5.2. Ethical considerations
   5.3. Conclusion

Machine Learning vs. Deep Learning

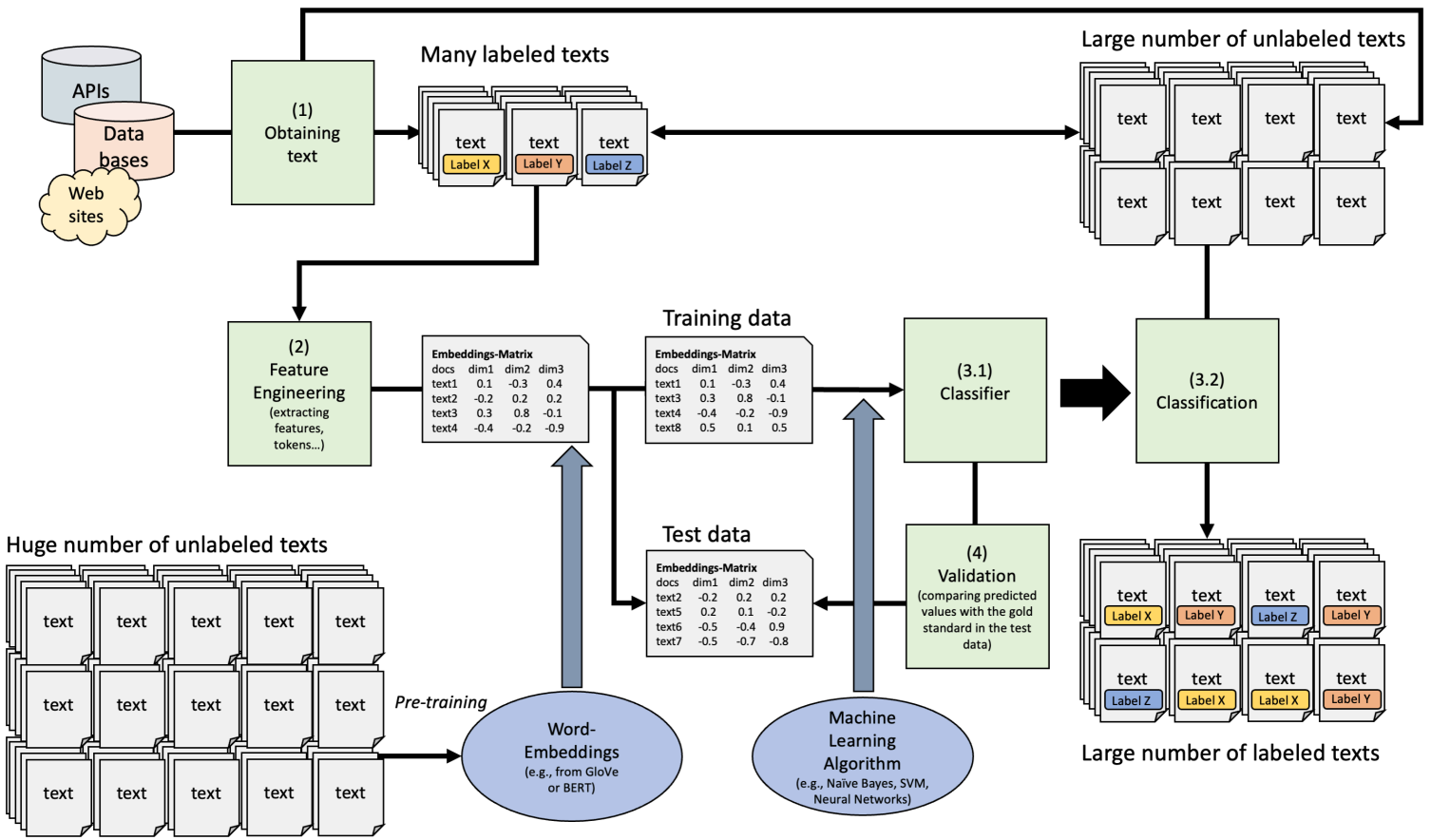
OUR TEXT CLASSIFICATION PIPELINE



CLASSIC MACHINE LEARNING (1990-2013)



WORD-EMBEDDINGS (2013-2020)



BUT MASSIVE ADVANCEMENTS IN RECENT YEARS

1. Massive advancement in how text can be represented at numbers
    - From simple word counts to word embeddings
    - From Static to contextual word embeddings
    - Increasingly better embedding of meaning
2. Pretraining and transfer learning
    - Word embeddings can be trained on large scale corpus
    - Pretrained word embeddings can fine-tuned (less training data) and then used for downstream tasks
3. Transformers and Generative AI
    - Larger and larger “language models”
    - New mechanisms for better embedding (e.g., Attention)
    - Conversational frameworks and Generative AI

The Rise of Transformers and Transfer Learning

ORIGIN OF THE TRANSFORMER

- Until 2017, the state-of-the-art for natural language processing was using a deep neural network (e.g., recurrent neural networks, long short-term memory and gated recurrent neural networks)
- In a preprint called “Attention is all you need”, published in 2017 and cited more than 95,000 times, the team of Google Brain introduced the so-called **Transformer**
- It represents a neural network-type architecture that learns context and thus meaning by tracking relationships in sequential data like the words in this sentence

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

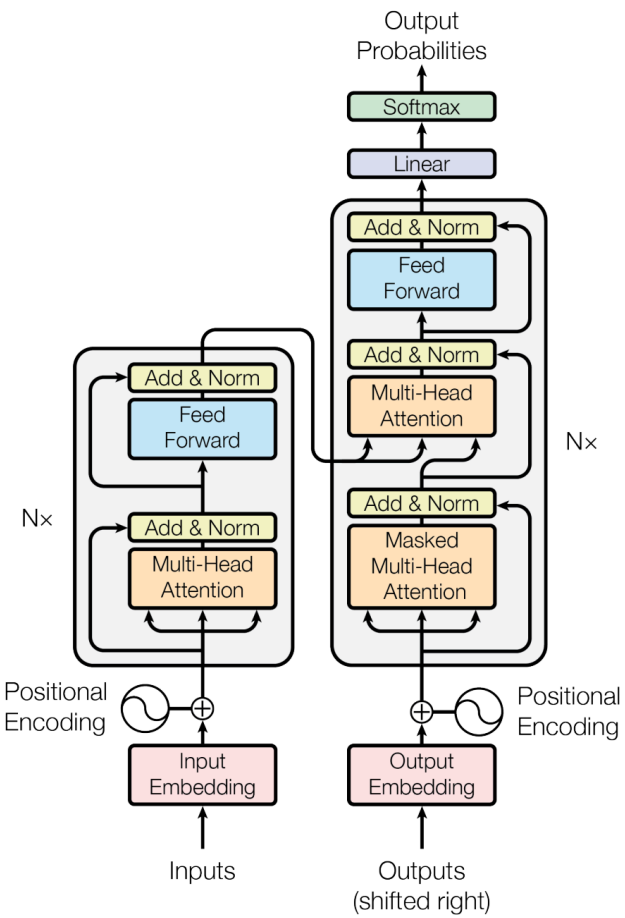
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions

WHAT WAS SO SPECIAL ABOUT TRANSFORMERS?

- Transformer models apply an evolving set of mathematical techniques, called **attention** or **self-attention**, to detect subtle ways even distant data elements in a series influence and depend on each other.
- The proposed network structure had notable characteristics:
  - No need for recurrent or convolutional network structures
  - Based solely on attention mechanism (stacked on top of one another)
  - Required less training time (can be parallelized)
- It thereby outperformed prior state-of-the-art models in a variety of tasks
- The transformer architecture is the back-bone of all current large language models and so far drives the entire “AI revolution”

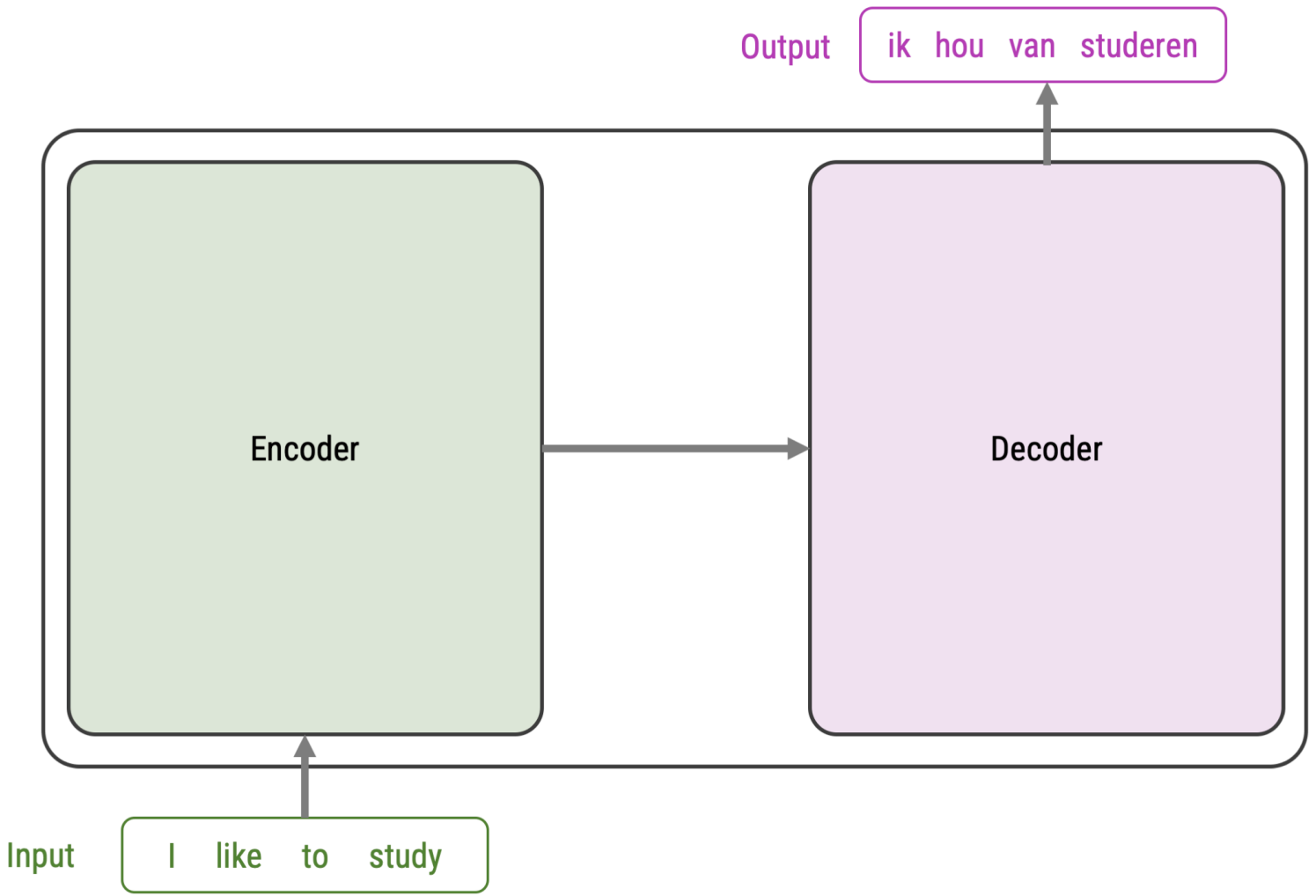
OVERVIEW OF THE ARCHITECTURE

- The figure on the right represent an abstract overview of a transformer's architecture
- It can be used for next-token-predictions
  - classic example is translation: e.g., english-to-dutch
  - but also: question-to-answer, text-to-summary, sentence-to-next-word…
  - an thus also for text classification: text-to-label
- Although models can differ, they generally include:
  - An encoder-decoder framework
  - Word embeddings + positional embedding
  - Attention and self-attention modules

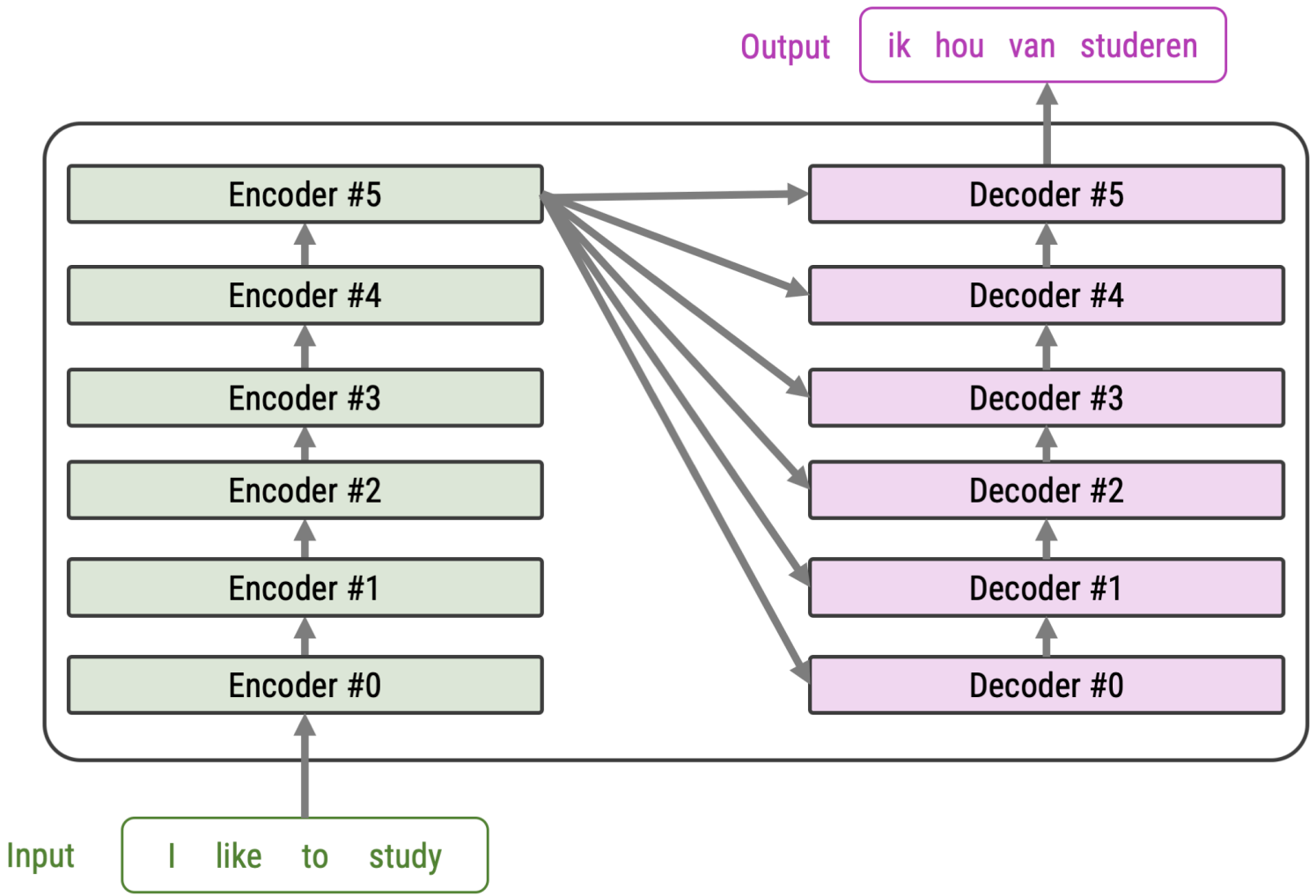


Vaswani et al. 2017

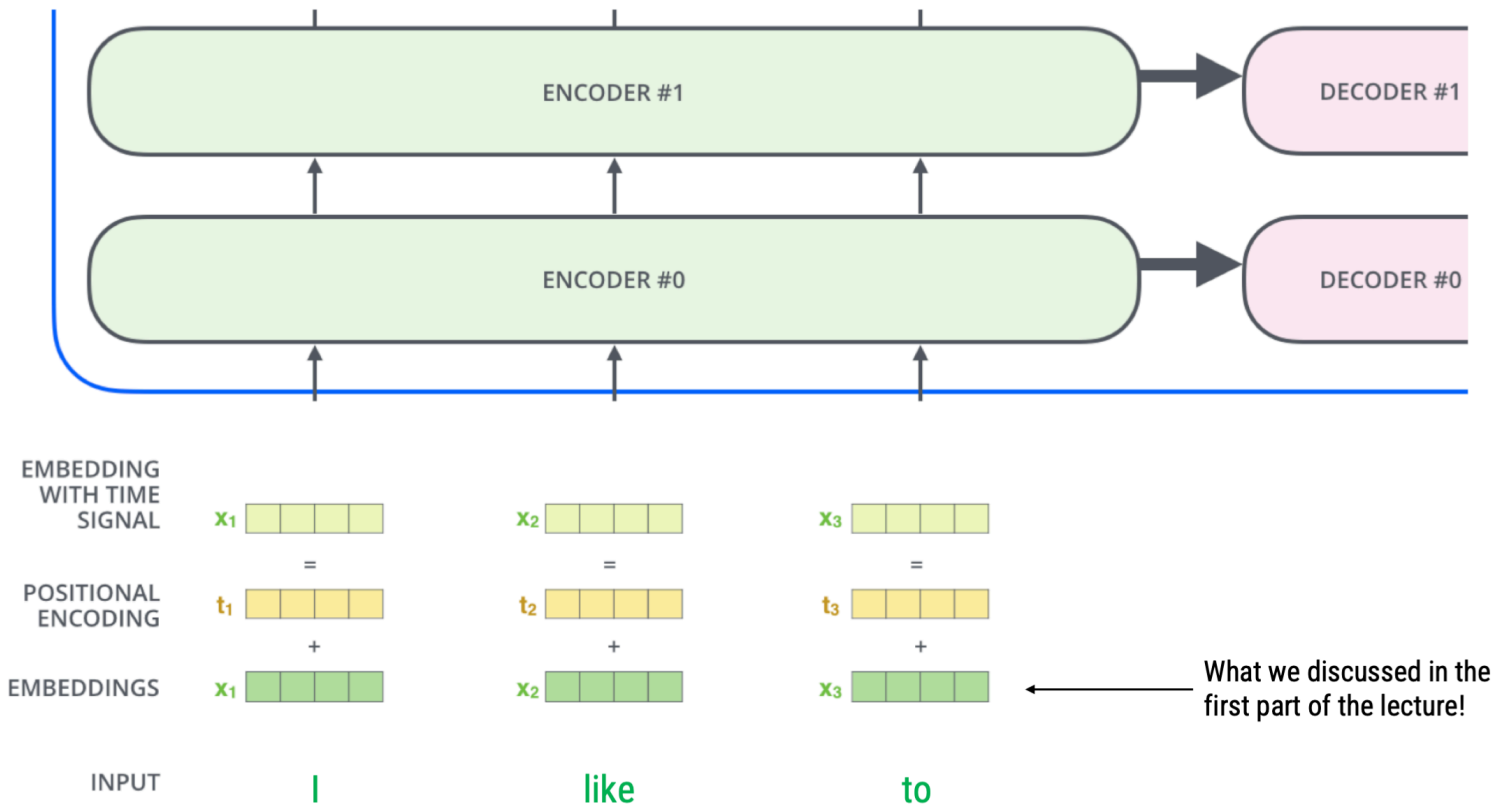
BASIC ENCODER-DECODER FRAMEWORK (FOR TRANSLATION)



STACKED ENCODERS AND DECODERS



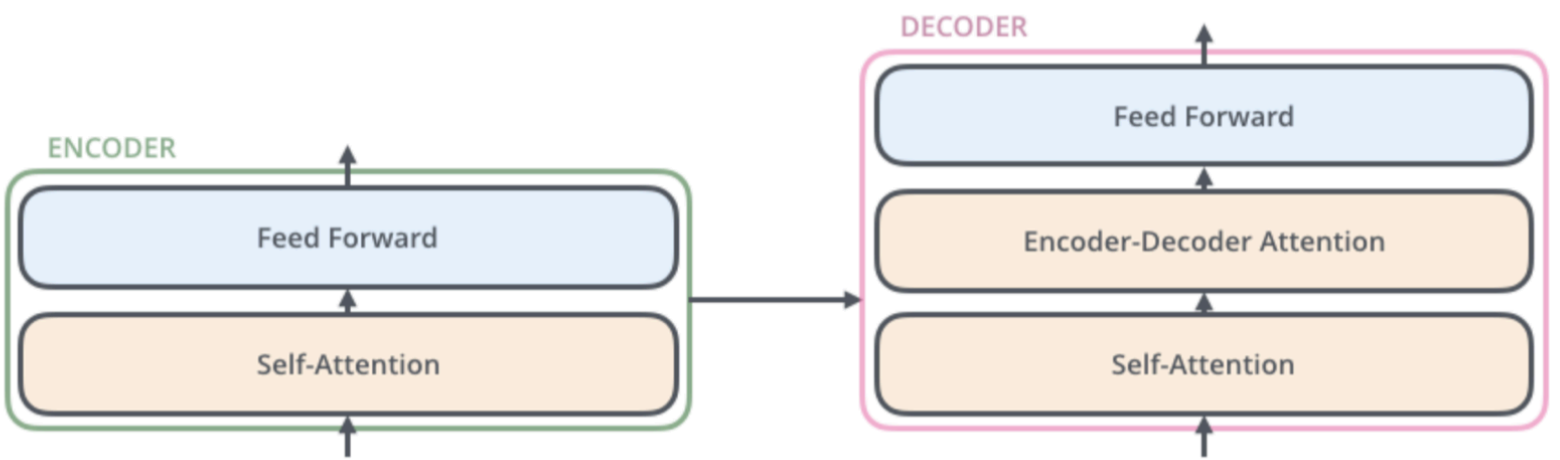
MORE ELABORATE ENCODING OF WORDS



Source: Alammar, 2018

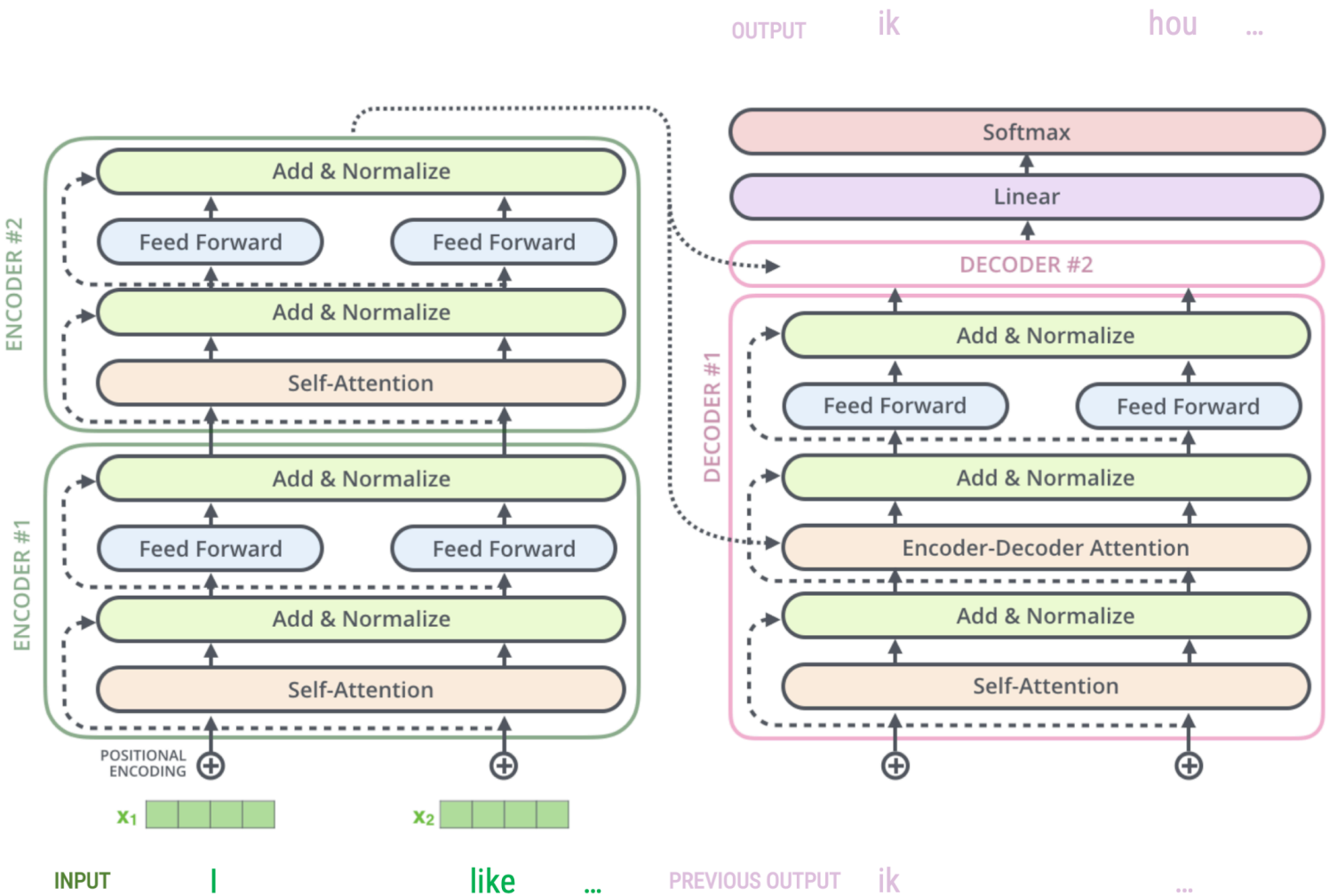
INSIDE OF AN ENCODER AND A DECODER

- The word, position, and time signal embeddings are passed to the first encoder
- Here, they flow through a self-attention layer, which further refines the encoding by “looking at other words” as it encodes a specific word
- The outputs of the self-attention layer are fed to a feed-forward neural network.
- The decoder likewise has both layers as well, but also an extra attention layer that helps to focus on different parts of the input (e.g., the encoders outputs)



Source: Alammar, 2018

PUTTING IT ALL TOGETHER

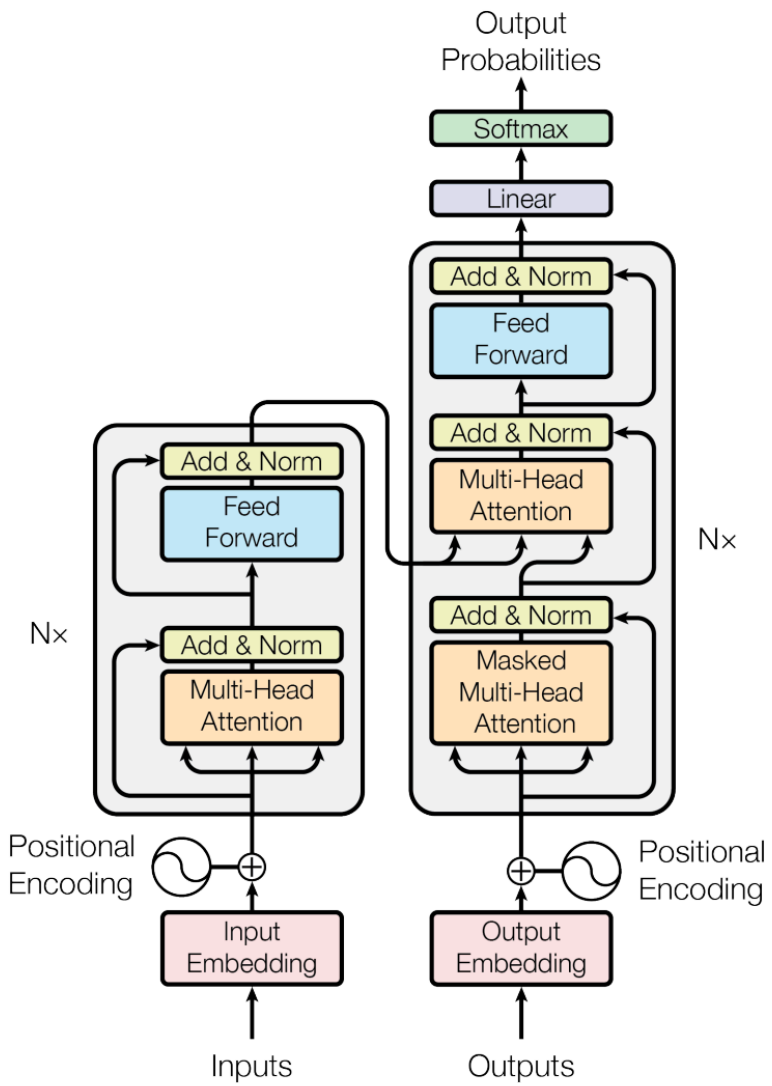


Source: Alammar, 2018

DIFFERENT TYPES OF MODELS FOR DIFFERENT TASKS

BERT

Encoder



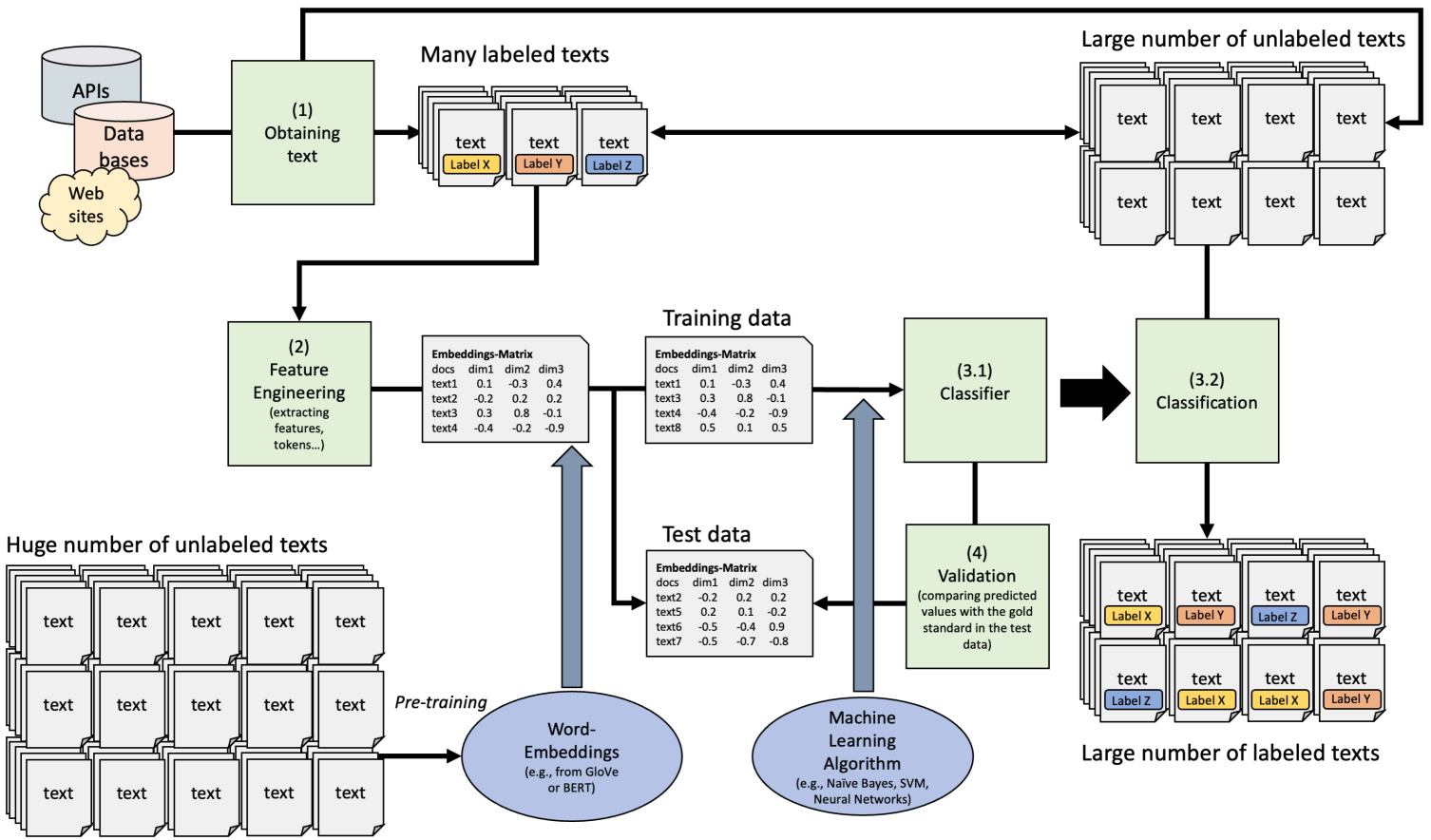
GPT

Decoder

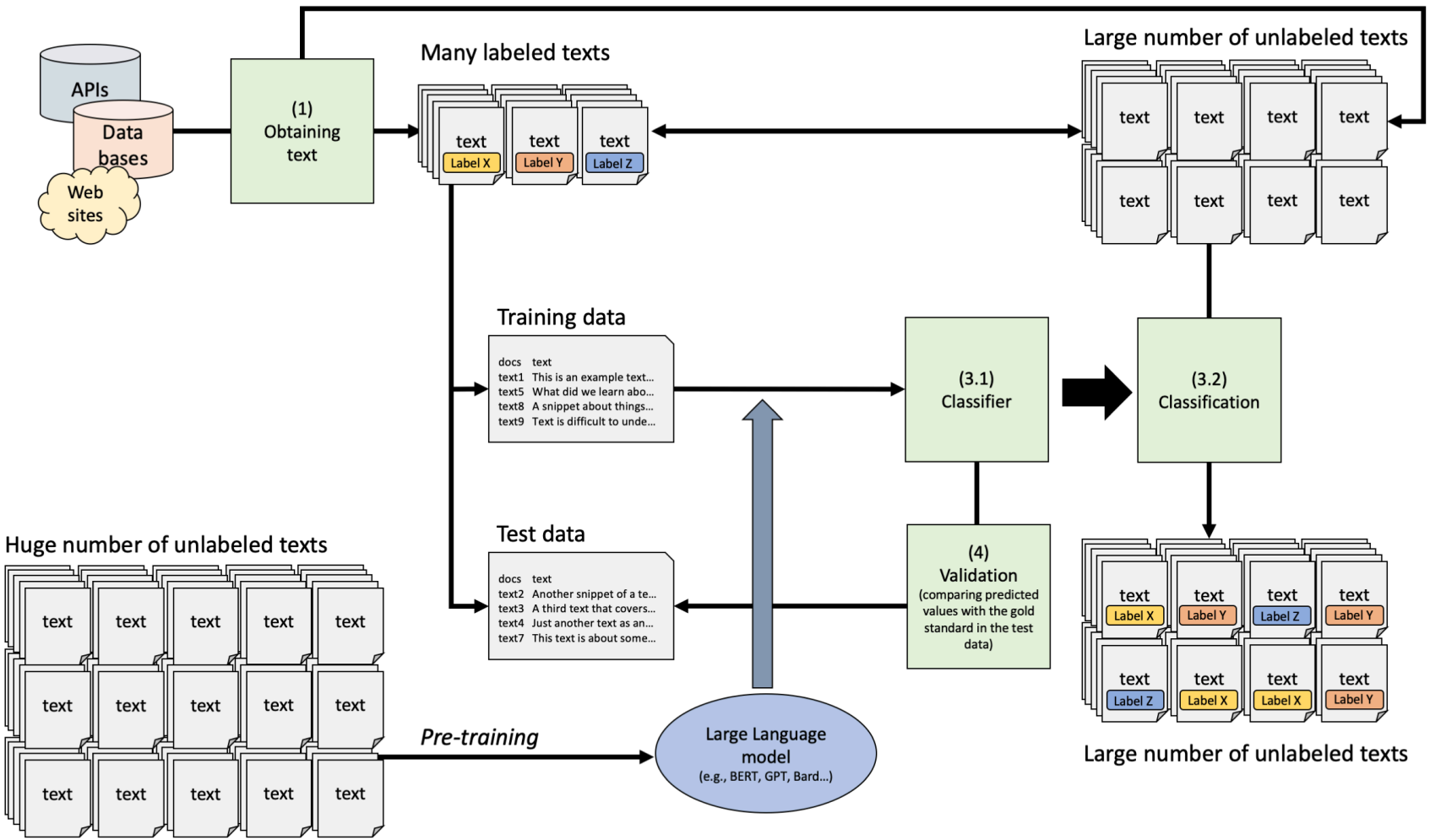
BERT VS. GPT (OR LLAMA)

- Encoder-Decoder Transformers:
  - BART (Lewis et al., 2019): tanslation, but also text generation ,…
- Encoder-Only Transformer
  - BERT (Devlin et al., 2019): Embedding-only, then down-stream tasks…
- Decoder-Only Transformer:
  - GPT-series (OpenAI): Text generation, …
  - Llama (Meta): Text generation…

TEXT CLASSIFICATION WITH TRANSFORMERS, ENCODER-ONLY



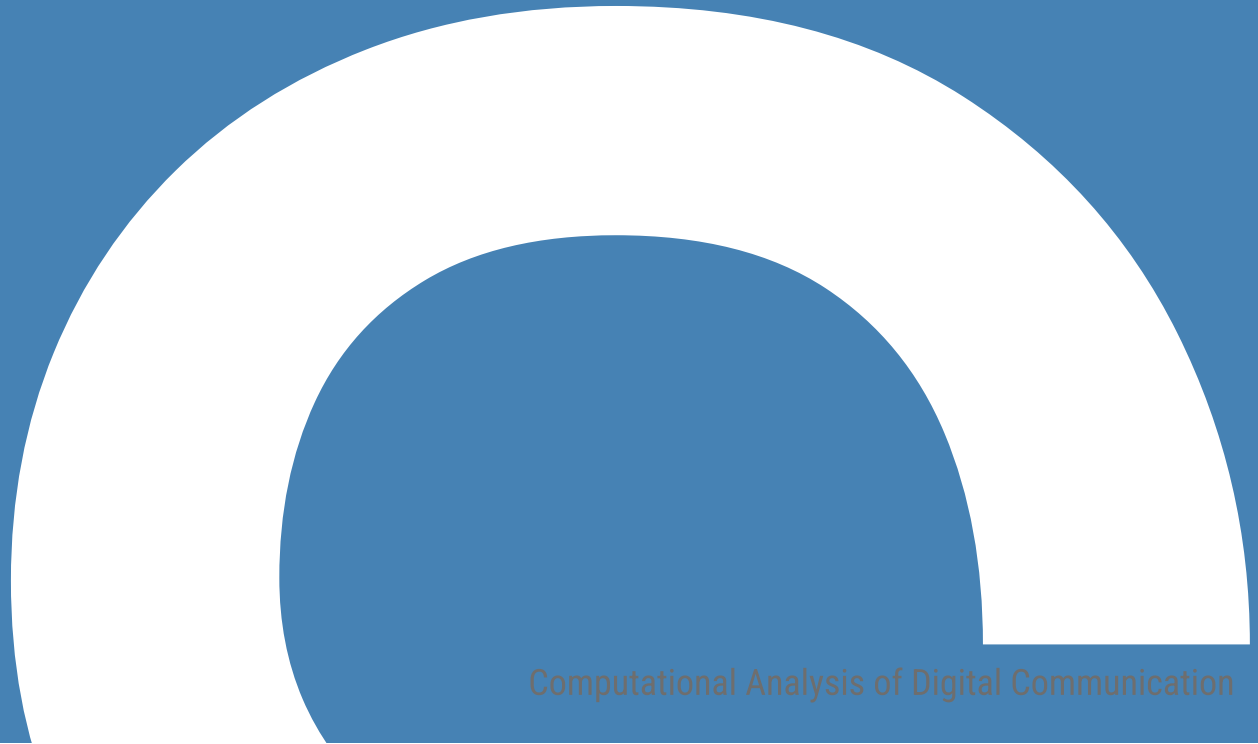
TEXT CLASSIFICATION WITH TRANSFORMERS, DECODER-ONLY

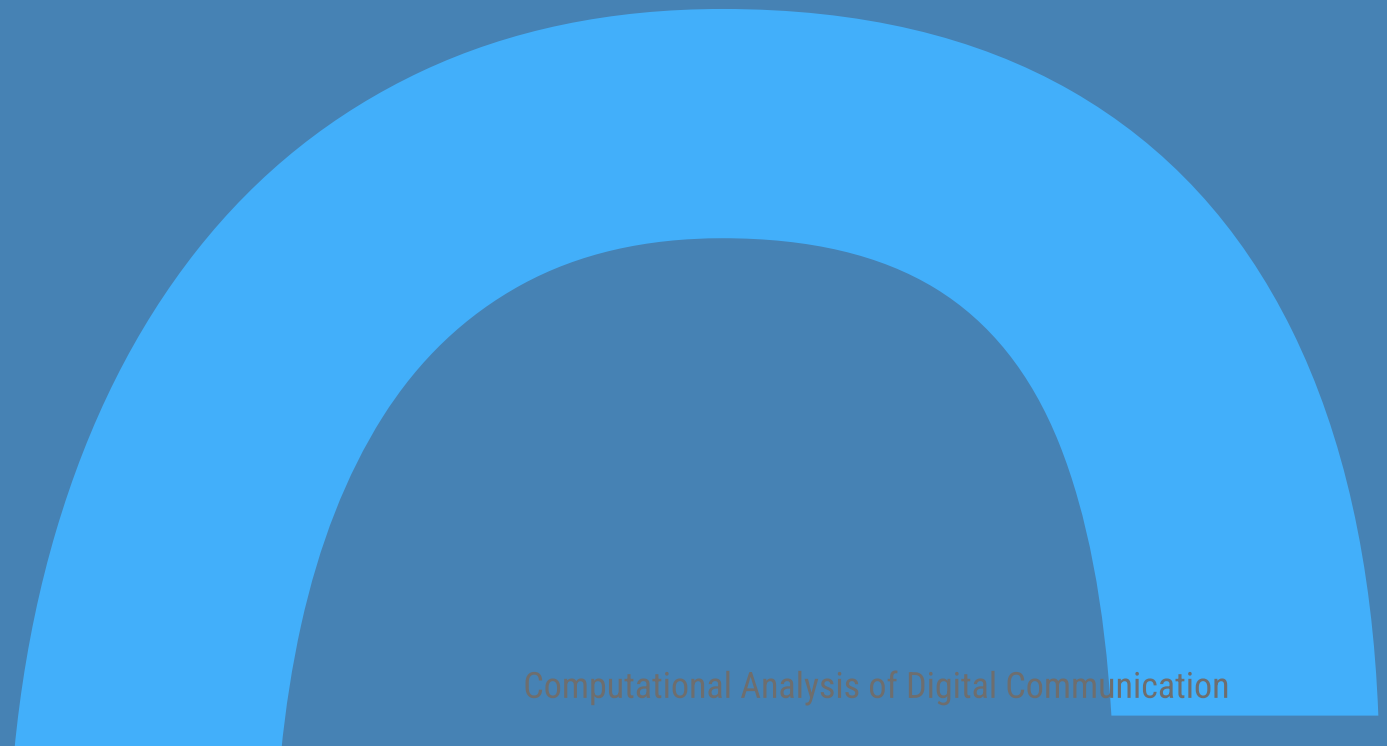


PRE-TRAINING, FINE-TUNING, AND TRANSFER LEARNING

- Generally, transformer models are pre-trained using specific natural language processing tasks
  - Mask Language Modelling (LM): simply mask some percentage of the input tokens at random, and then predict those masked tokens
  - Next sentence prediction (NSP): predict sentence B from sentence A to model relationships between sentences
- The general idea was to use a pre-trained model and then "fine-tune" it on the specific tasks it is supposed to perform (e.g., annotating text with topics or sentiment)
- Although the transformer's architecture has made training more efficient (due to the ability to parallelize), it nonetheless requires significant computing power to fine-tune a model
- As pre-training often involves tasks that are different than what we want the model to do, this is often denoted as "transfer learning", thus a type of learning that transfers to other task as well
- As transformer-based models become larger and larger, the need for fine-tuning decreases as they already do well on down-stream tasks (e.g., using only prompt-engineering)

Very Large Language Models: Llama and GPT





Computational Analysis of Digital Communication

VU VRIJE UNIVERSITEIT AMSTERDAM

WHAT ARE LARGE LANGUAGE MODELS

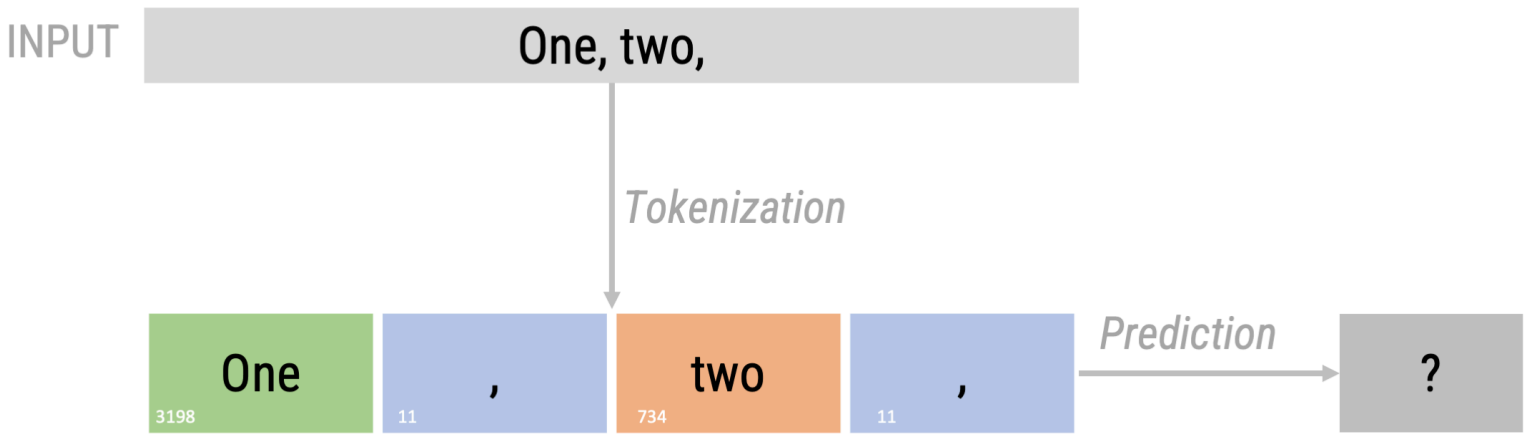
- A large language model (LLM) is a type of language model notable for its ability to achieve general-purpose language understanding and generation
- LLMs acquire these abilities by using massive amounts of data to learn billions of parameters during training and consuming large computational resources during their training and operation
- LLMs are still just a type of artificial neural networks (mainly transformers!) and are pretrained using self-supervised learning and semi-supervised learning (e.g., Mask Language Modelling)
- As so-called autoregressive language models, they take an **input text** and repeatedly **predicting the next token** or word

CURRENT MODELS

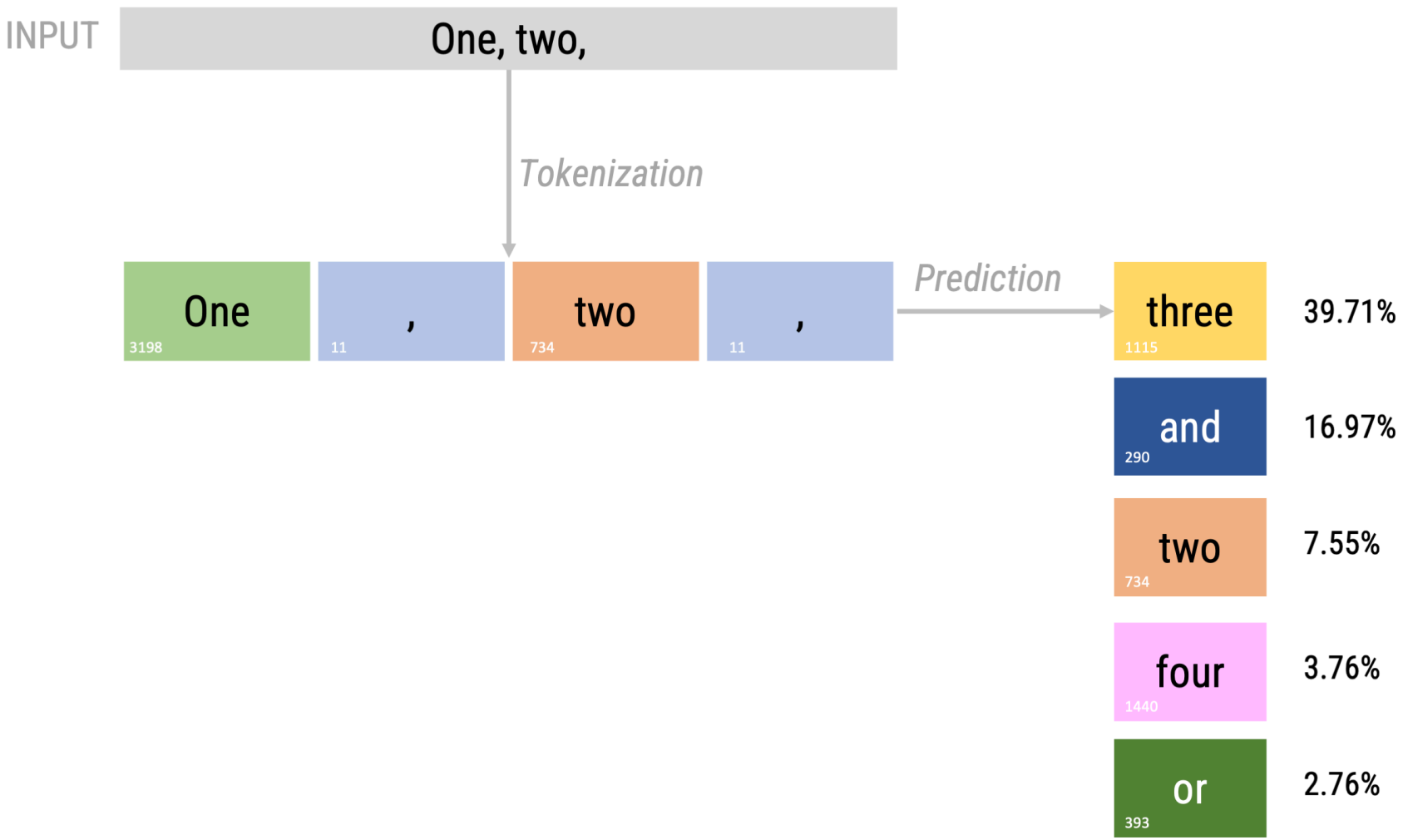
Text-to-Text	Text-to-Image	Image-to-Text	Image-to-3D	Image or Video-to-3D
- ChatGPT - Bard - LLaMa (Meta) - PaLM 2 - Claude - ...many more	- Midjourney - DALL-E 3 - Stable Diffusion - Muse - Imagen - Bard	- ChatGPT - Flamingo - Visualart	- Dream Fusion - Magic3D	- CSM AI

Text-to-Audio	Text-to-Code	Image-to-Science	Text-to-video	Audio-to-text
- AutoLM - Jukebox	- Codex - Alphacode	- Galatica - Minerva	- Runway - Cuebric - Phenaki - Sora	- Whisper

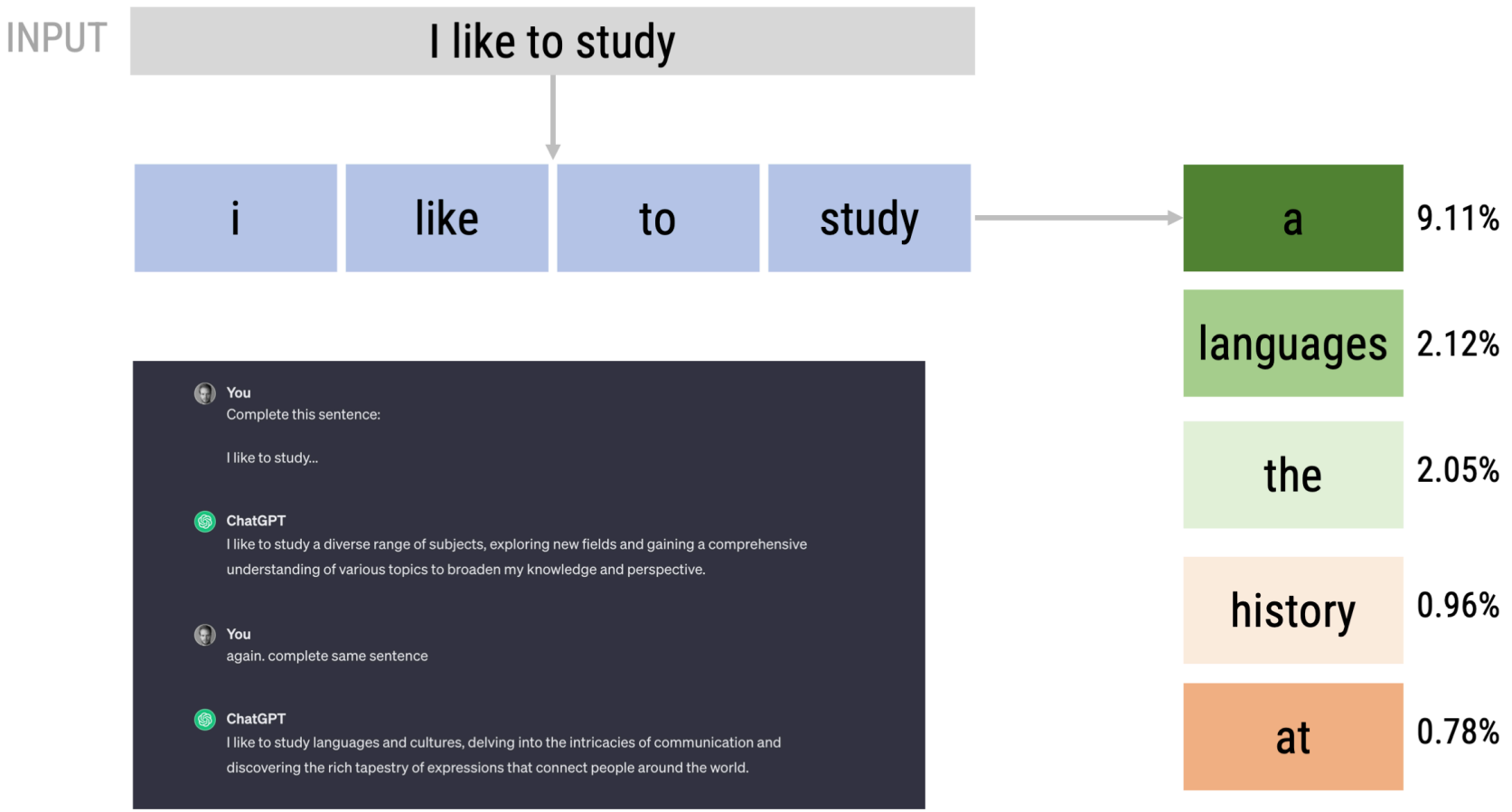
NEXT TOKEN PREDICTION (AS IN GPT-2)



NEXT TOKEN PREDICTION (AS IN GPT-2)



NEXT TOKEN PREDICTION



GPT-SERIES BY OPENAI

- Generative Pre-trained Transformer (GPT), is a set of state-of-the-art large language model developed by OpenAI.
- Particularly GPT-3, released publicly in November 2022 together with a chat interface, caused a lot of public attention
- Millions of users in a very short amount of time (faster than Facebook, Instagram, TikTok, etc…), now 1.5 Billion users

ChatGPT

OVERVIEW OF THE GPT-SERIES BY OPENAI

OpenAI's "GPT-n" series

Model	Architecture	Parameter count	Training data	Release date	Training cost
GPT-1	12-level, 12-headed Transformer decoder (no encoder), followed by linear-softmax.	117 million	BookCorpus:[27] 4.5 GB of text, from 7000 unpublished books of various genres.	June 11, 2018[8]	30 days on 8 P600 GPUs, or 1 petaFLOP/s-day.[8]
GPT-2	GPT-1, but with modified normalization	1.5 billion	WebText: 40 GB of text, 8 million documents, from 45 million webpages upvoted on Reddit.	February 14, 2019 (initial/limited version) and November 5, 2019 (full version)[28]	"tens of petaflop/s-day",[29] or 1.5e21 FLOP.[30]
GPT-3	GPT-2, but with modification to allow larger scaling	175 billion[31]	499 billion tokens consisting of CommonCrawl (570 GB), WebText, English Wikipedia, and two books corpora (Books1 and Books2).	May 28, 2020[29]	3640 petaflop/s-day (Table D.1 [29]), or 3.1e23 FLOP.[30]
GPT-3.5	Undisclosed	175 billion[31]	Undisclosed	March 15, 2022	Undisclosed
GPT-4	Also trained with both text prediction and RLHF; accepts both text and images as input. Further details are not public.[26]	Undisclosed. Estimated 1.7 trillion[32]	Undisclosed	March 14, 2023	Undisclosed. Estimated 2.1e25 FLOP.[30]

Source: Wikipedia

NOW WHAT DOES THAT ACTUALLY MEAN…?

Source: Wikipedia

OVERVIEW OF THE GPT-SERIES BY OPENAI

OpenAI's "GPT-n" series

Model	Architecture	Parameter count	Training data	Release date	Training cost
GPT-1	12-level, 12-headed Transformer decoder (no encoder), followed by linear-softmax.	117 million	BookCorpus:[27] 4.5 GB of text, from 7000 unpublished books of various genres.	June 11, 2018[8]	30 days on 8 P600 GPUs, or 1 petaFLOP/s-day.[8]
GPT-2	GPT-1, but with modified normalization	1.5 billion	WebText: 40 GB of text, 8 million documents, from 45 million webpages upvoted on Reddit.	February 14, 2019 (initial/limited version) and November 5, 2019 (full version)[28]	"tens of petaflop/s-day",[29] or 1.5e21 FLOP.[30]
GPT-3	GPT-2, but with modification to allow larger scaling	175 billion[31]	499 billion tokens consisting of CommonCrawl (570 GB), WebText, English Wikipedia, and two books corpora (Books1 and Books2).	May 28, 2020[29]	3640 petaflop/s-day (Table D.1 [29]), or 3.1e23 FLOP.[30]
GPT-3.5	Undisclosed	175 billion[31]	Undisclosed	March 15, 2022	Undisclosed
GPT-4	Also trained with both text prediction and RLHF; accepts both text and images as input. Further details are not public.[26]	Undisclosed. Estimated 1.7 trillion[32]	Undisclosed	March 14, 2023	Undisclosed. Estimated 2.1e25 FLOP.[30]

4.6 minutes

1.3 hours

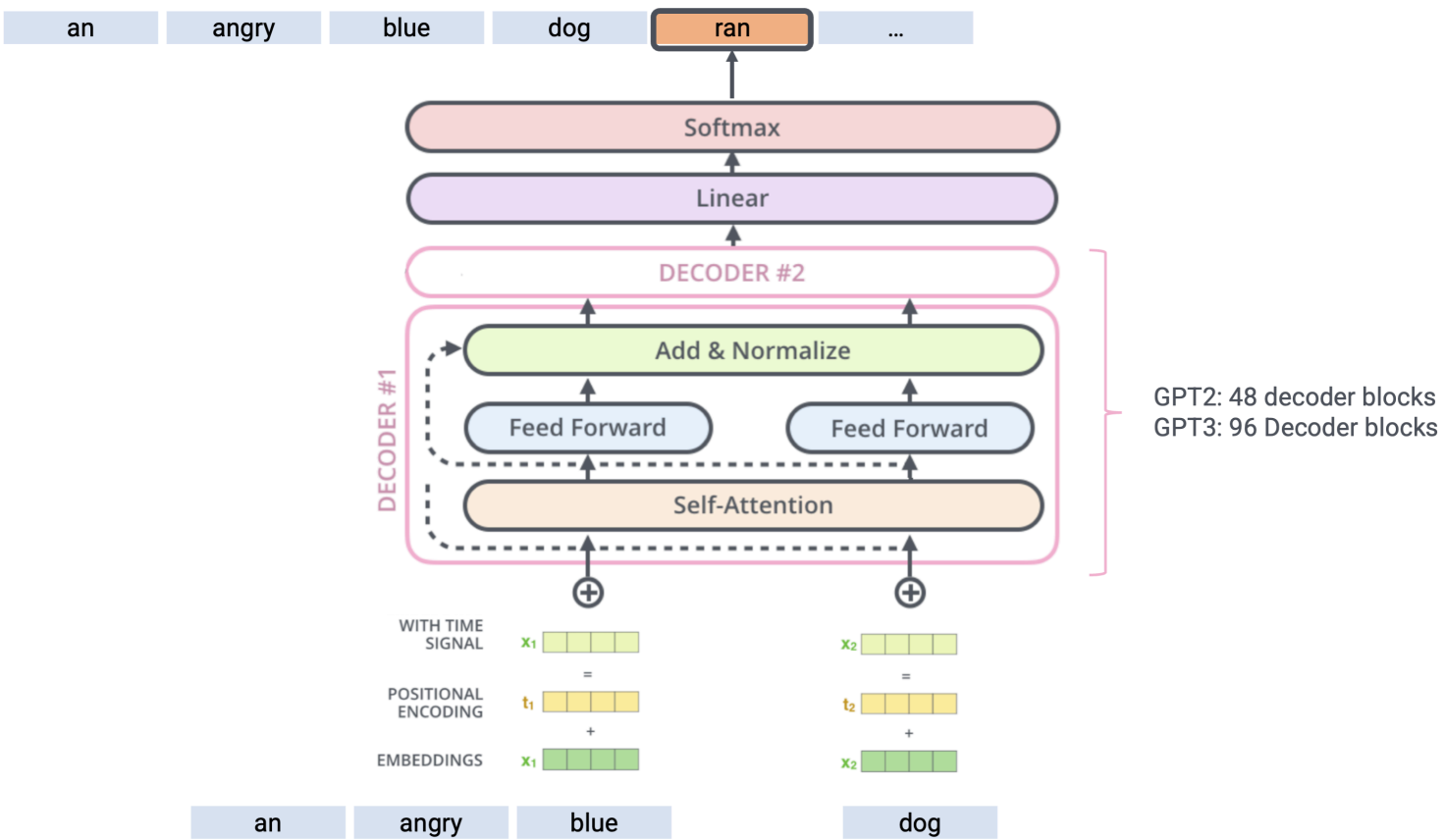
11.6 days

2,152 years

Source: Wikipedia

A Peek into the Architecture of GPT

NEXT-TOKEN-PREDICTION BASED ON INPUT TEXT



Source: Adapted from Alammar, 2018

INTRICATE MEANING OF WORDS

Harry stood at the edge of the Forbidden Forest, his wand gripped tightly in his hand.

Prince Harry delivered a heartfelt speech at a star-studded charity gala last night.

INTRICATE MEANING OF WORDS

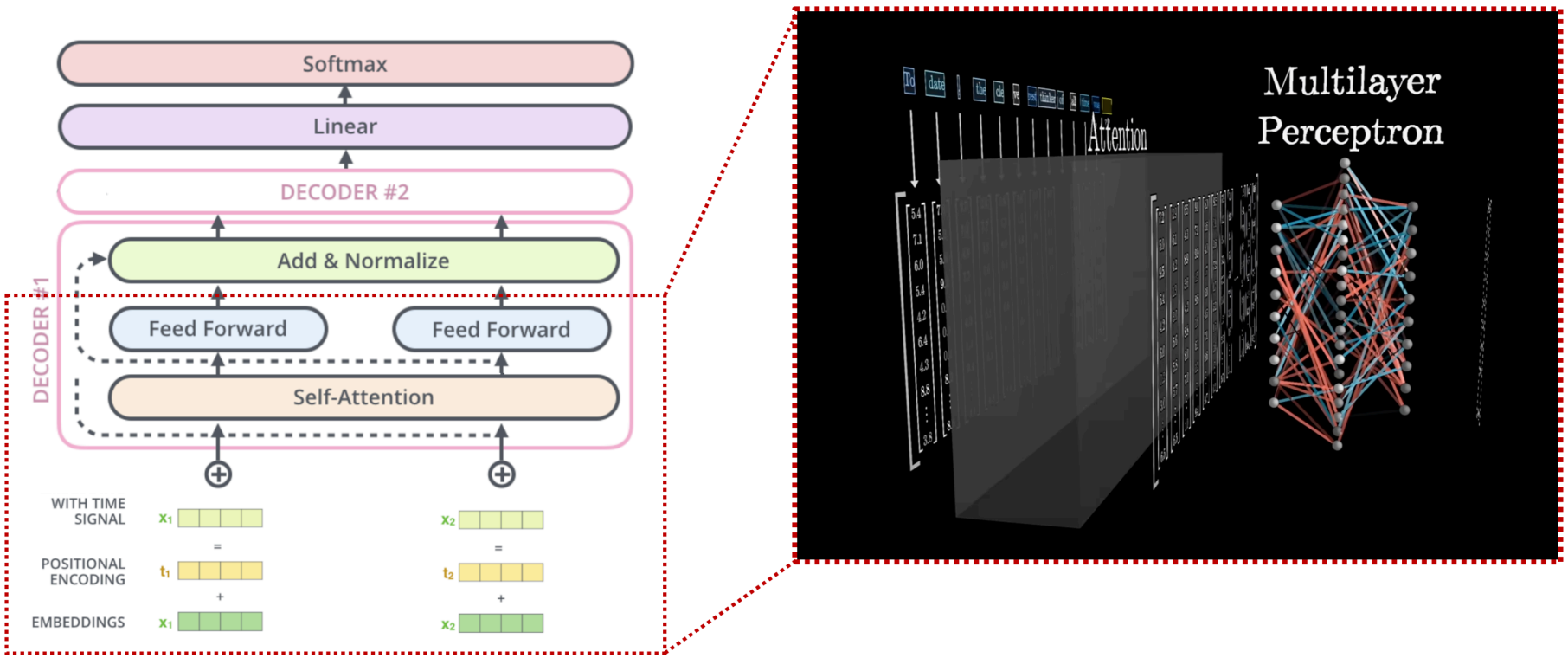
Harry stood at the edge of the Forbidden Forest, his wand gripped tightly in his hand.

Prince **Harry** delivered a heartfelt speech at a star-studded charity gala last night.

CORE IDEA: BETTER ENCODING

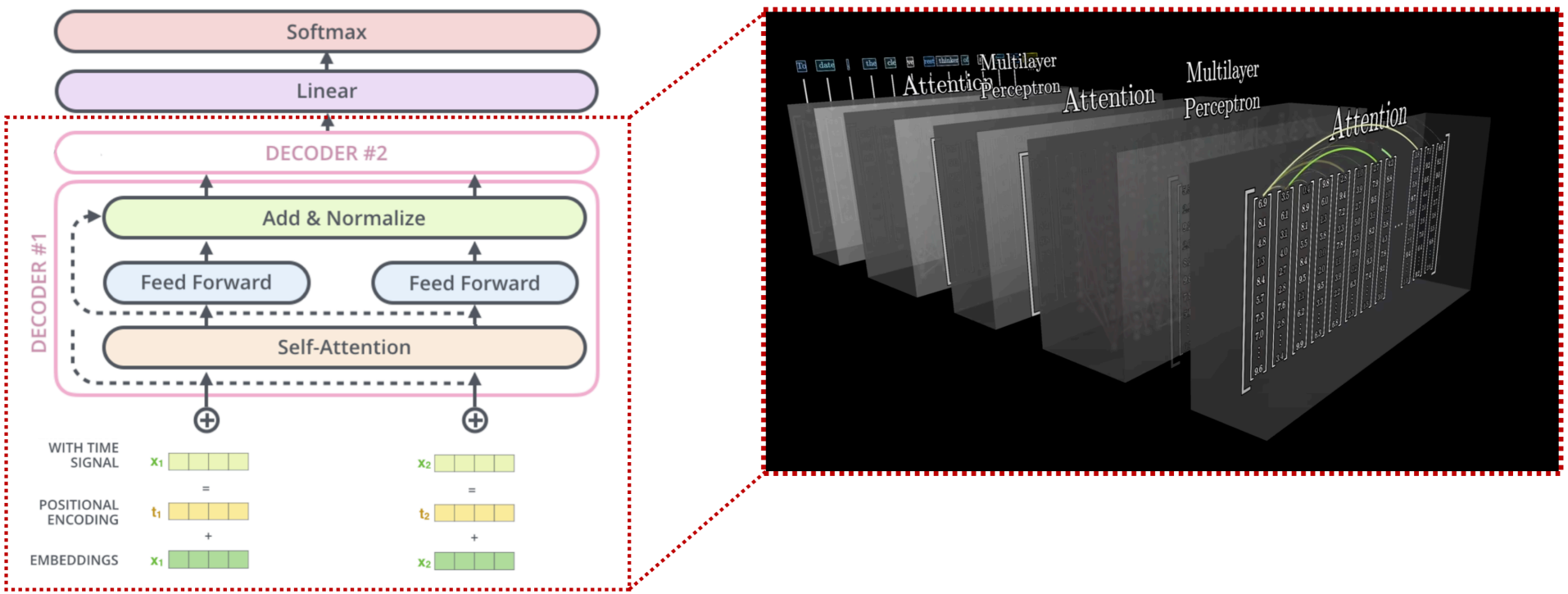
- Based on static word-embeddings (that we discussed in the last lecture), the word “Harry” would get the same embeddings vector, even though we clearly see that they refer to different persons
- The same is true for words like “mole” (which can refer to an animal or a little skin spot) or “model” (e.g., a fashion model vs. a computer model)
- LLMs like GPT work so well, because their architecture allows them to encode additional information into a token’s embedding vector and take surrounding tokens into account
- This way, they learn which name (e.g., Harry) refers to which person or which word (e.g., model) refers to which meaning

A LARGE NEURAL NETWORK



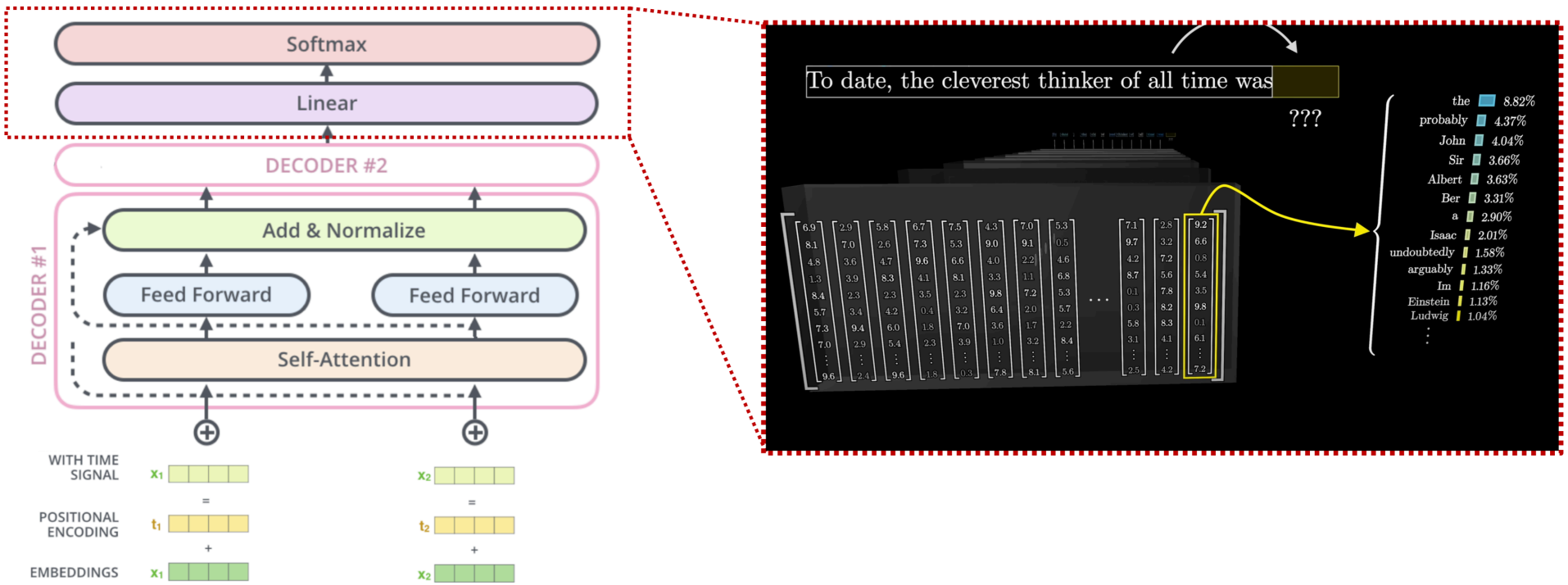
Source: Alammar, 2018 and 3Blue1Brown, 2024

MANY LAYERS OF ATTENTION



Source: Alammar, 2018 and 3Blue1Brown, 2024

MANY LAYERS OF ATTENTION



Source: Alammar, 2018 and 3Blue1Brown, 2024

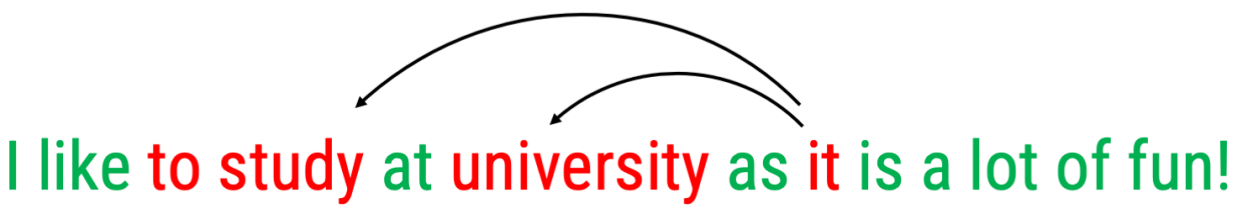
GENERAL IDEA BEHIND ATTENTION

- In general terms, self-attention works encodes how similar each word is to all the words in the sentence, including itself
- Once the similarities are calculated, they are used to determine how the transformers should update the embeddings of each word

I like to study at university as it is a lot of fun!

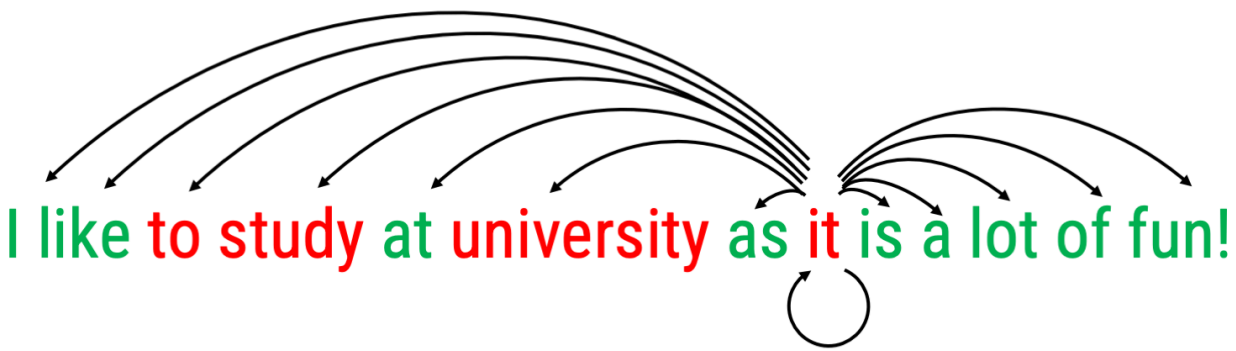
GENERAL IDEA BEHIND ATTENTION

- In general terms, self-attention works encodes how similar each word is to all the words in the sentence, including itself.
- Once the similarities are calculated, they are used to determine how the transformers should update the embeddings of each word



GENERAL IDEA BEHIND ATTENTION

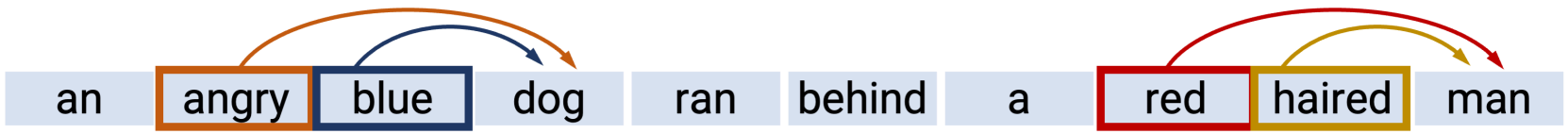
- In general terms, self-attention works encodes how similar each word is to all the words in the sentence, including itself.
- Once the similarities are calculated, they are used to determine how the transformers should update the embeddings of each word



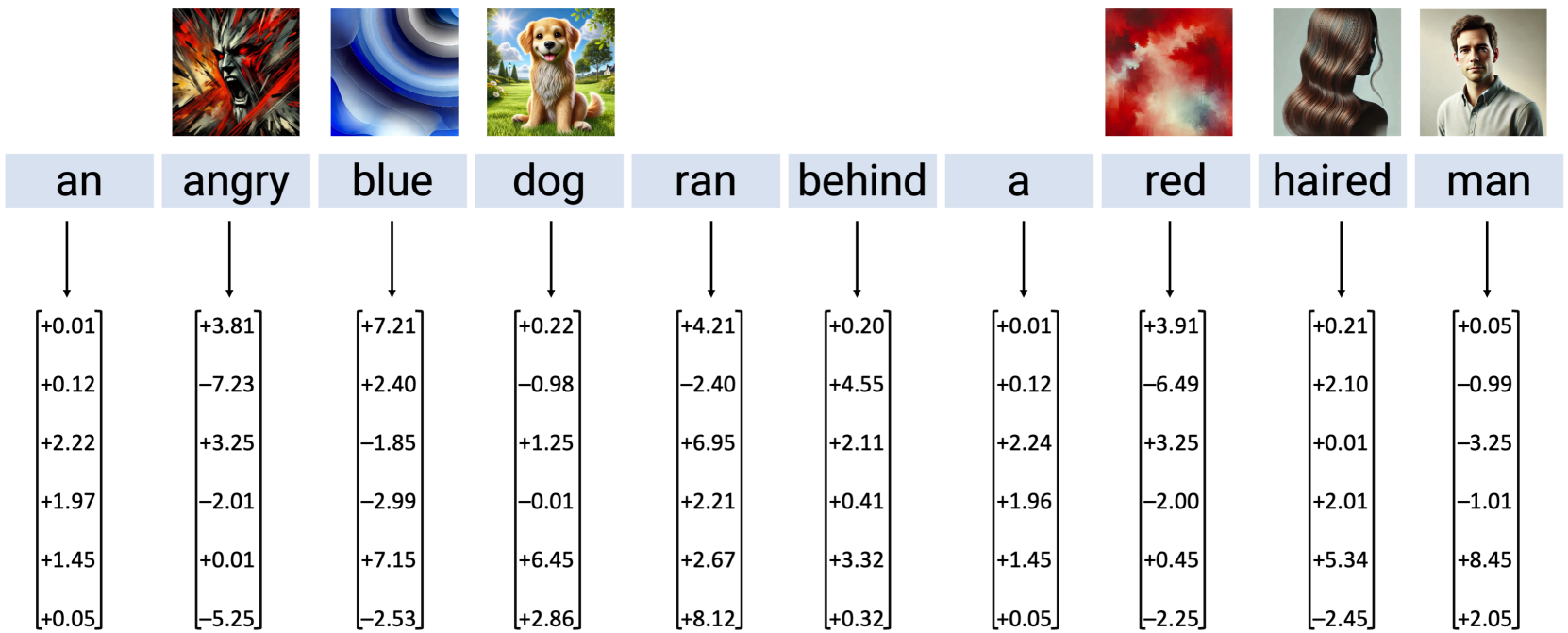
ATTENTION IN DETAIL

an	angry	blue	dog	ran	behind	a	red	haired	man

ATTENTION IN DETAIL

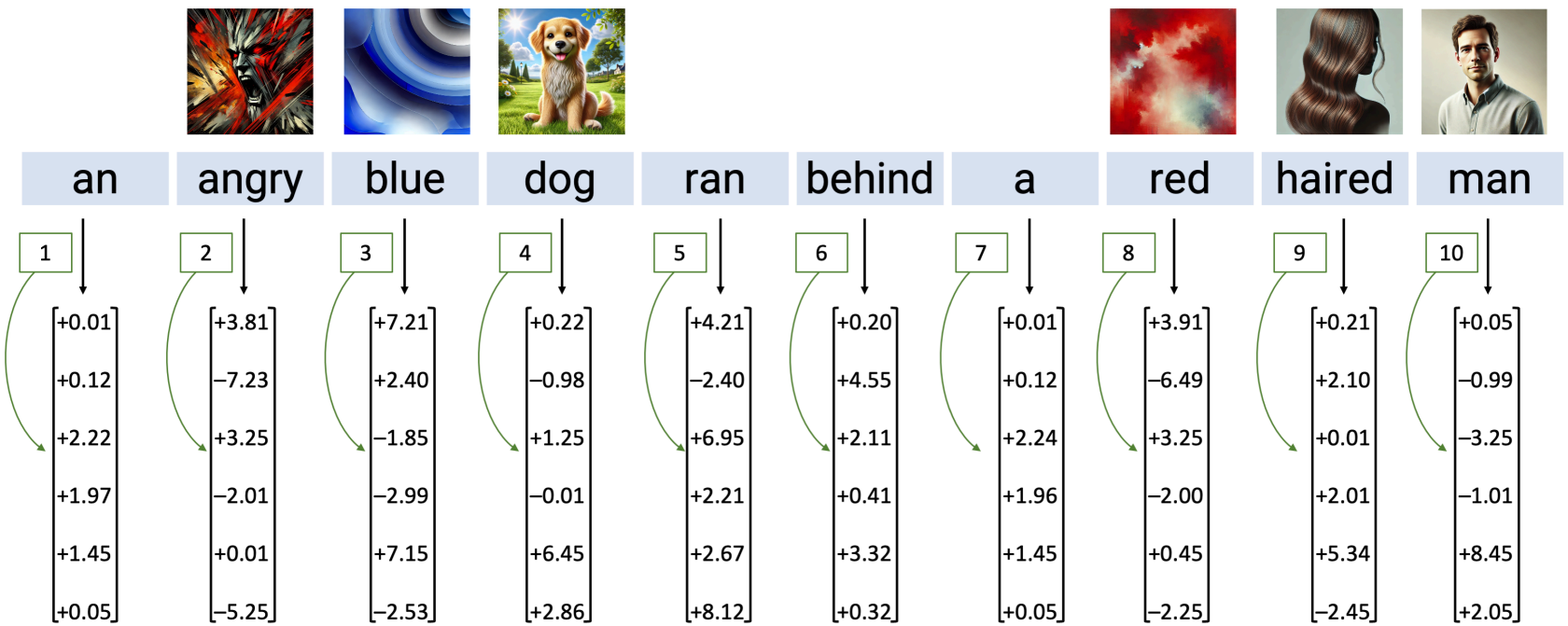


ATTENTION IN DETAIL: STATIC EMBEDDINGS



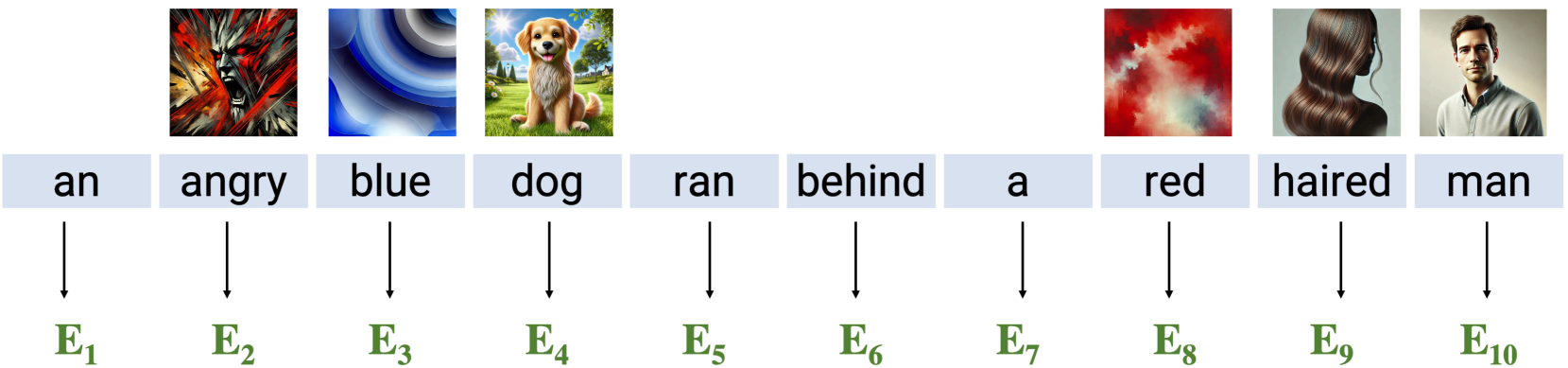
Inspiration: 3Blue1Brown, 2024

ATTENTION IN DETAIL: POSITIONAL ENCODING



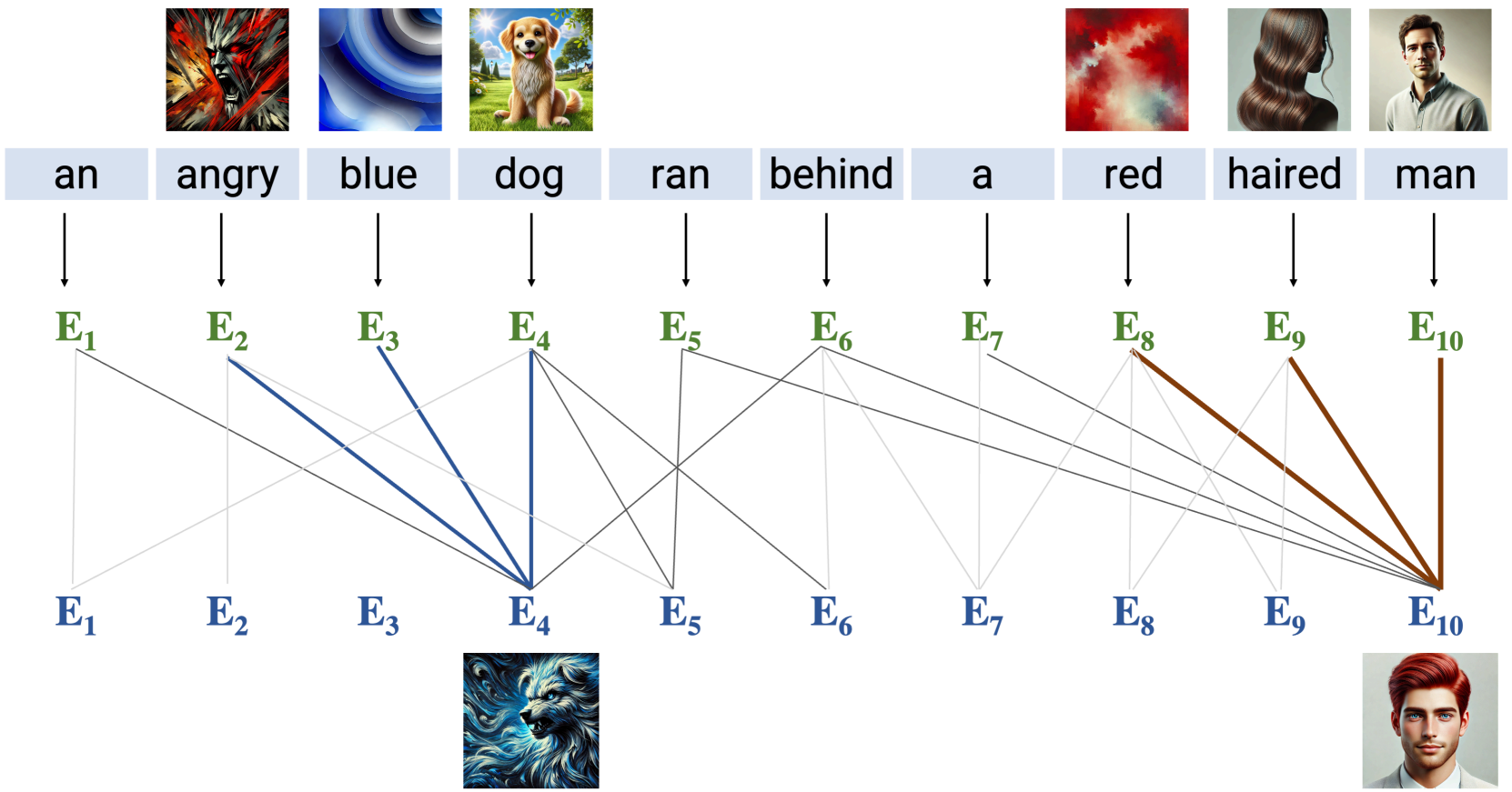
Inspiration: 3Blue1Brown, 2024

ATTENTION IN DETAIL: MECHANISM



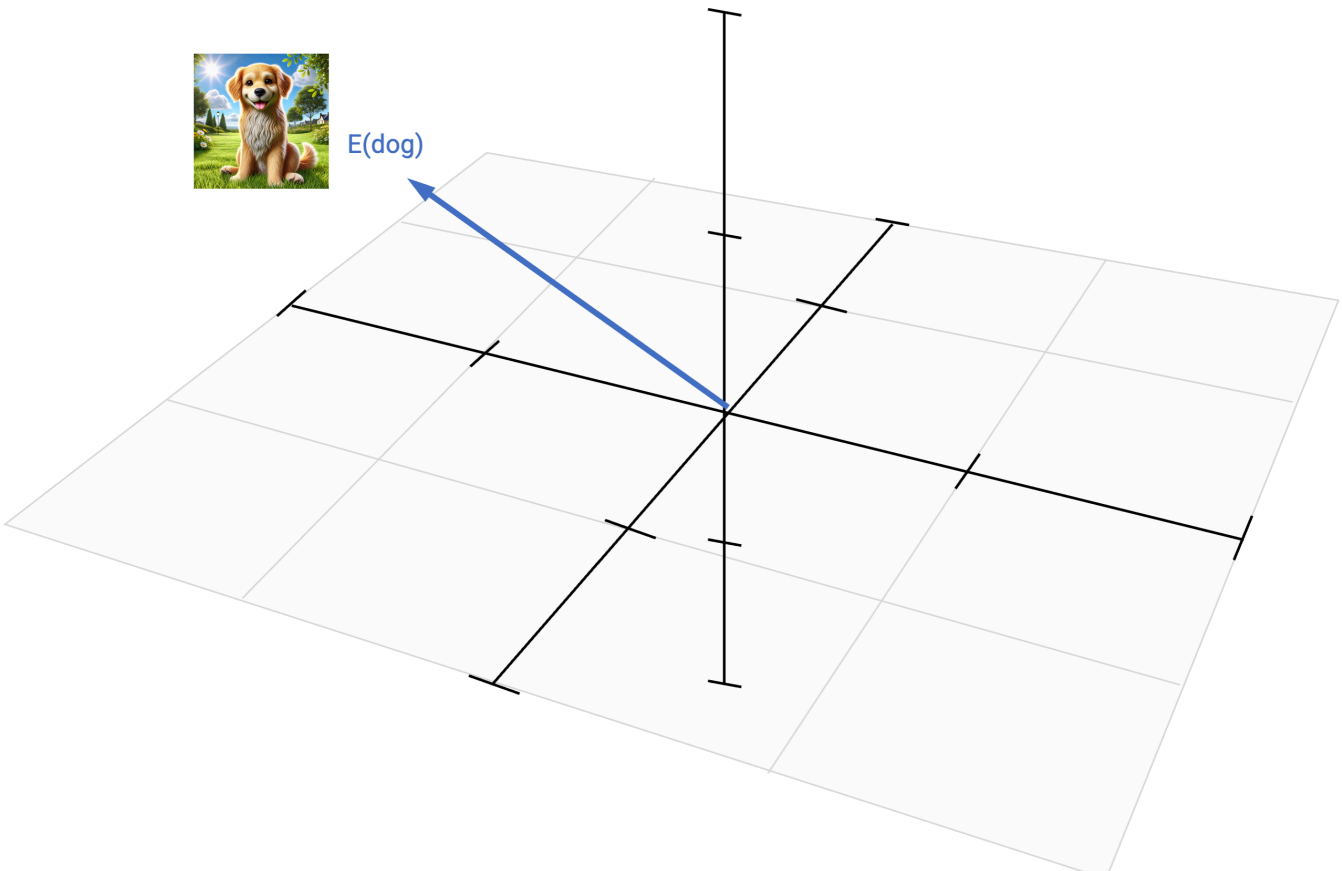
Inspiration: 3Blue1Brown, 2024

ATTENTION IN DETAIL: MECHANISM

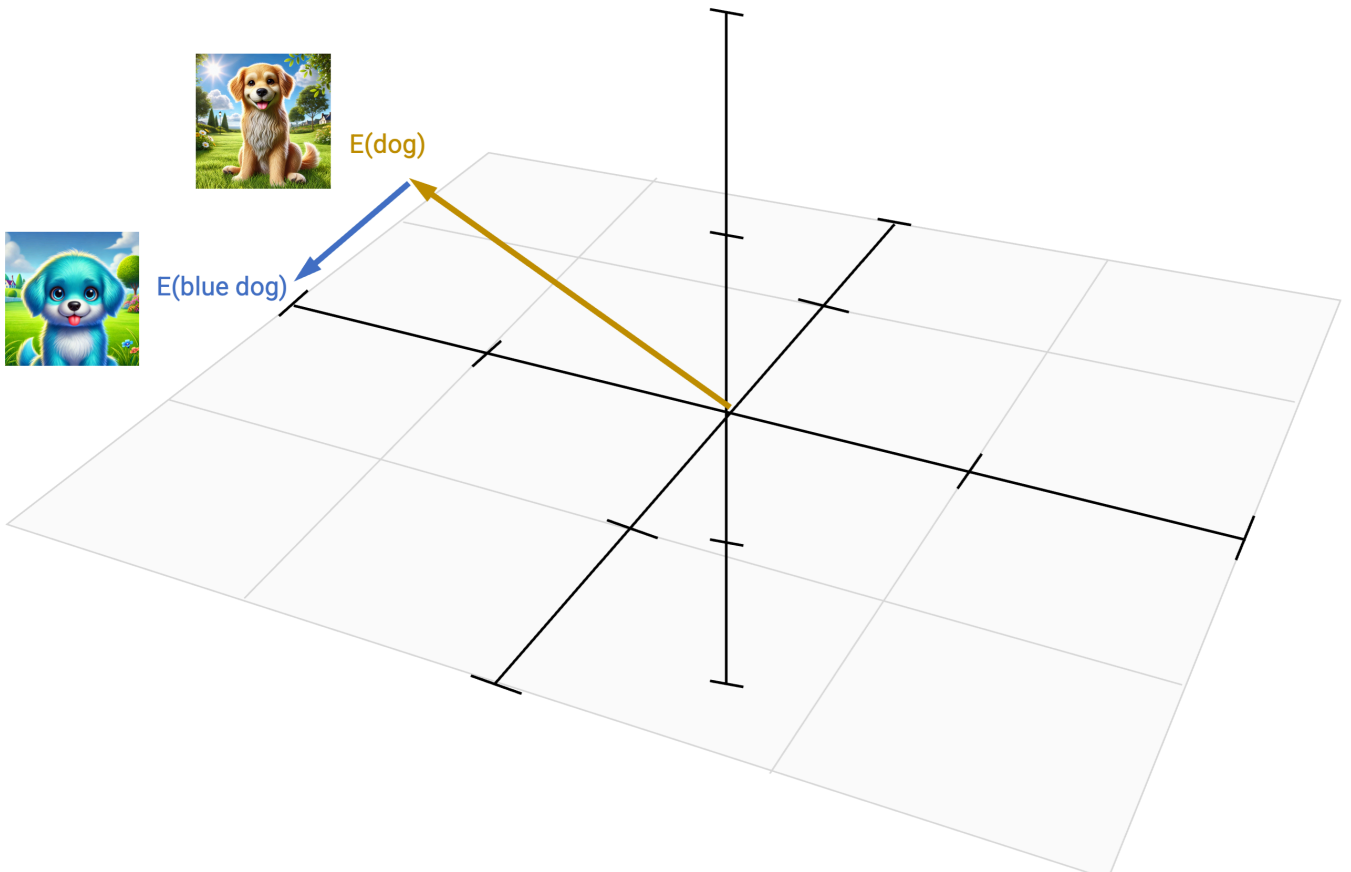


Inspiration: 3Blue1Brown, 2024

UPDATING WORD EMBEDDINGS IN THE K-DIMENSIONAL SPACE

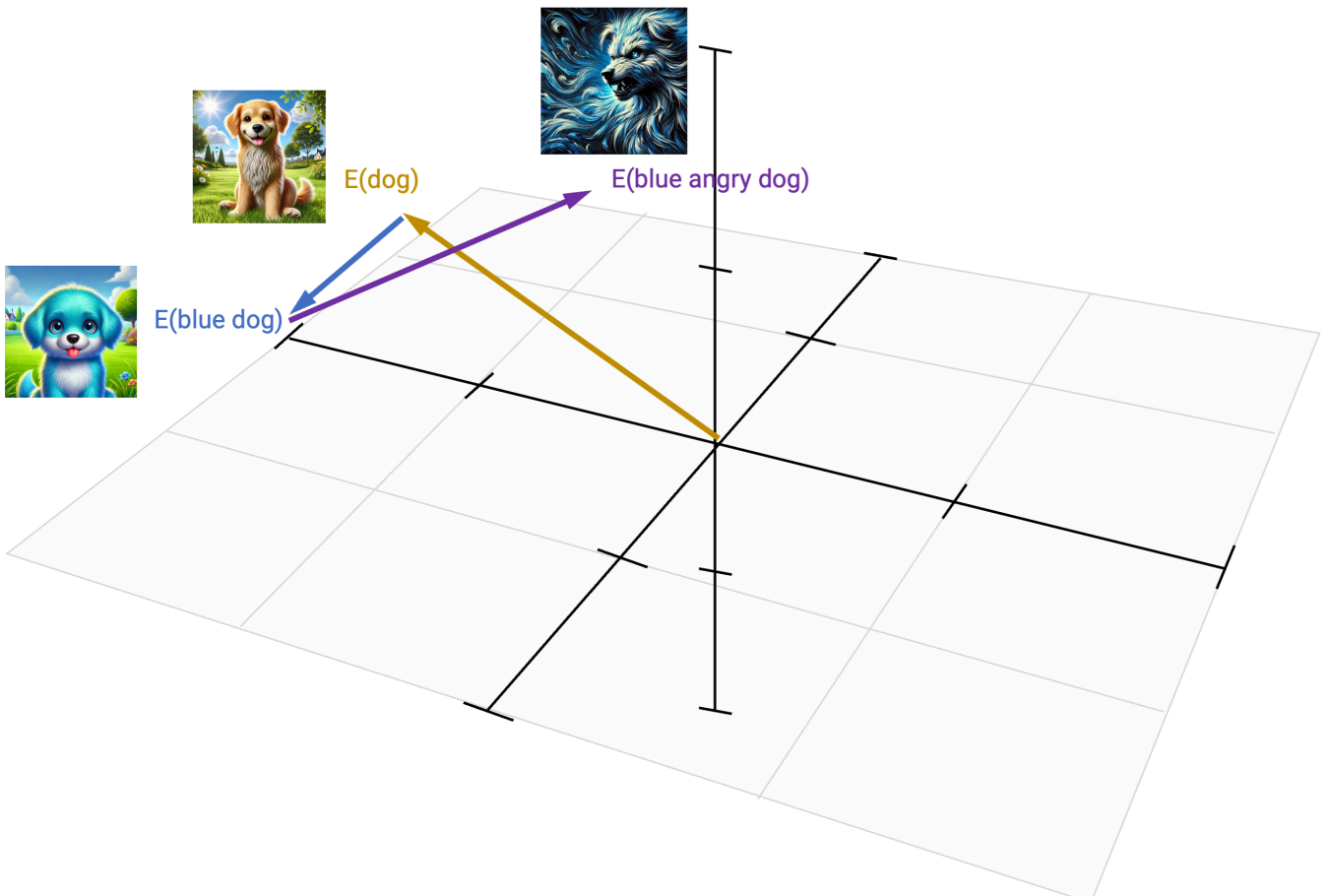


UPDATING WORD EMBEDDINGS IN THE K-DIMENSIONAL SPACE



Inspiration: 3Blue1Brown, 2024

UPDATING WORD EMBEDDINGS IN THE K-DIMENSIONAL SPACE



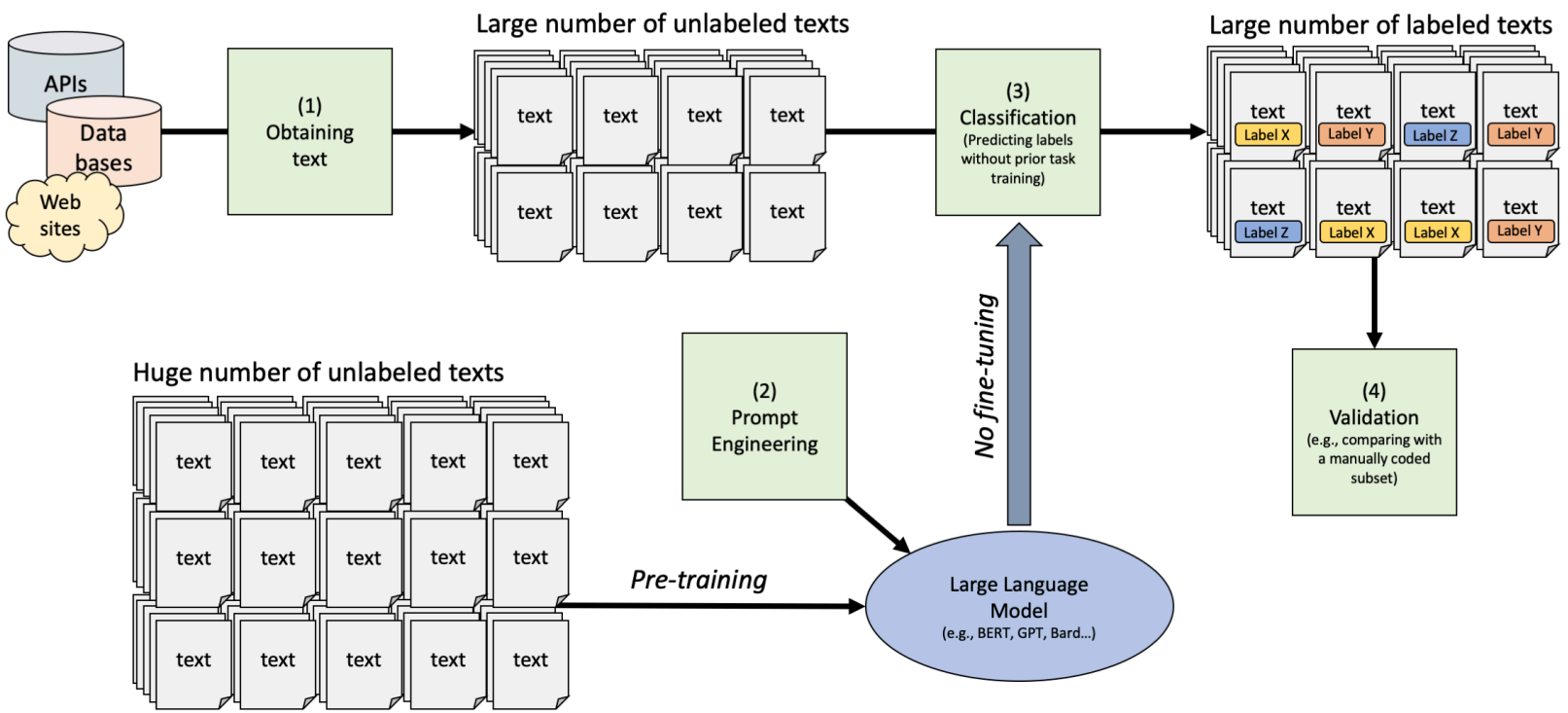
Inspiration: 3Blue1Brown, 2024

SUMMARY OF THE GPT ARCHITECTURE

- This was only a very short peek into the architecture of GPT (and to degree also into large language models generally)
- Decoder-only model designed for text generation
- Based on static word-embedding that are updated via postional encoding and many attention layers that can even encode distant relationships between words and sentences
- In principle, just a lot of matrix algebra: We are constantly updating a multitude of vectors that represent words and their meaning in the context of a sentence and the entire text
- Now, we can ask how we can use these models for text classification tasks

Using LLMs for Text Classification

TEXT CLASSIFICATION WITH LARGE LANGUAGE MODELS



ZERO-SHOT, ONE-SHOT, AND FEW-SHOT CLASSIFICATION

- Classic machine learning models (last lecture) are typically trained on a specific set of classes, and their performance is evaluated on the same set of classes during testing
- LLMs have the ability to perform a task or make predictions on a set of classes that it has never seen or been explicitly trained on
- In other words, the model can generalize its pre-trained knowledge to new, unseen tasks without training for those tasks
- Depending on the type of **prompt engineering**, i.e., how we describe the task for the LLM, we differentiate three types of classifications:
  - Zero-Shot: No examples are given
  - One-Shot: One example is given
  - Few-Shot: More than one example is given
- It is not straight-forward which strategy works best for which task.
- At times, zero-shot classification works well as the model is not too tied to the examples

OVERVIEW OF CLASSIFICATION WITHOUT TRAINING

	Zero-Shot Classification	One-Shot Classification	Few-Shot Classification
Definition	Providing no examples	Providing one example	Providing a few examples
Example prompt	In the following social media posts, look for instances where people offer constructive feedback. 1 = Constructive Feedback present 0 = Constructive Feedback absent	In the following social media posts, look for instances where people offer constructive feedback. 1 = Constructive Feedback present 0 = Constructive Feedback absent ***Example:*** You should always keep two offers on the table before accepting an offer." ***Classification:*** 1	In the following social media posts, look for instances where people offer constructive feedback. 1 = Constructive Feedback present 0 = Constructive Feedback absent ***Example:*** You should always keep two offers on the table before accepting an offer." ***Classification:*** 1 ***Example:*** Haha, that's life." ***Classification:*** 0
Strengths	The model is free in how it interprets the prompt, which can mean it better choses from the range of possibilities	The model has an example to draw from, but because it is just one, it usually doesn't limit its flexibility	The model has clear-cut rules with regard to how to code the texts. Helps in streamlining the codes
Weaknesses	Can lead to unwanted codes or completely unreliable coding	Less open than zero-shot	Too many examples can constrain the models ability to generalize beyond the examples

HOW TO WORK WITH LLMS

- There are generally three ways in which we can work with LLMs:
  - Assess open source models via the hugging face API (only smaller rates per minute, but still useful)
  - Assess models such as GPT via their respective API (limited rates per minute, and costs per token)
  - Download and use models via “ollama” on our own computer (requires some space and can be computationally intensive)
- In this course, we are going to “play around” with GPT-3.5 and GPT-4 via the OpenAI API and (if possible) via download small models via ollama
  - The package `tidyllm` provides a straightforward workflow that intersects with the tidy-style analyses that we already now
  - API for openAI will be provided by teachers
  - A tutorial on how to install ollama will be provided

A SMALL EXAMPLE IN `tidyllm`

- In the package `tidyllm`, we can create a prompt using the function `llm_message()``
- We then pass this prompt to a chat and specify the model we want to use (this can be a local model, like llama, or a model on a server, like GPT)

```r
library(tidyverse)
library(tidymodels)
library(glue)
library(tidyllm)

llm_message("What is this sentence about: To be, or not to be: that is the question.") |>
  chat(ollama(.model = "llama3", .temperature = 0))
```

```
Message History:
system:
You are a helpful assistant
-------------------------------------------------------------
user:
What is this sentence about: To be, or not to be: that is
the question.
-------------------------------------------------------------
assistant:
A classic!

This sentence is from the opening of William Shakespeare's
play "Hamlet", Act 3, Scene 1. It is a famous soliloquy
spoken by Prince Hamlet himself.

The sentence is about the existential crisis and
philosophical dilemma that Hamlet is grappling with. He
is contemplating whether to take action against his uncle
Claudius, who has murdered his father (the king) and taken
the throne for himself. Hamlet is torn between two options:
```

```
1. To be: to exist, to live, to take action, and potentially
face the consequences of doing so.
2. Not to be: to not exist, to die, to avoid the troubles
and uncertainties of life.

The sentence sets the tone for the rest of the play, which
explores themes of mortality, morality, and the human
condition.
```

ZERO-SHOT CLASSIFICATION: PROMPT ENGINEERING

- In a zero-shot classification framework, we simply provide the task.
- Additionally, we can give potential answer options, which help streamlining the code (otherwise, we get full text answers due to the next-token-prediction logic of LLMs!)

```
set.seed(42)

# We create a small test data set
test_data <- science_data |>
  sample_n(size = 100)

# We create a codebook that includes all texts.
codebook <- glue("Identify the discipline based on this abstract: {abstract}

                  Pick one of the following numerical codes from this list.
                  Respond only with the code!

                  1 = Computer Science
                  2 = Physics
                  3 = Statistics",
                  abstract = test_data$text)

# Create a list of llm-message prompts
classification_task <- map(codebook, llm_message)
```

- We create a list that contains already all prompts with all texts

ZERO-SHOT CLASSIFICATION: SEQUENTIAL PROMPTING

- Next, we create a function that sends the message via the API to the relevant model
- Then, we use the function `pmap_dfr()` to "map" this function across our list of prompts

```
# Create a function that to map across the prompts
classify_sequential_llama <- function(texts, message){
    raw_code <- message |>
        chat(ollama(.model = "llama3", .temperature = .0)) |>
        get_reply()
    tibble(text = texts, label = raw_code)
}

# Run the classification by sequentially prompting LLama3
results_llama <- tibble(texts = test_data$text,
                        message = classification_task) |>
  pmap_dfr(classify_sequential_llama, .progress = T)
```

As a result, we get a data set with the relevant code.

```
head(results_llama)
```

```
# A tibble: 6 × 2
  text                                                                label
  <chr>                                                               <chr>
1 "Optimal stopping via reinforced regression In this note we propose a n… 1
2 "Traces of surfactants can severely limit the drag reduction of superhy… 2
3 "A Unified Strouhal-Reynolds Number Relationship for Laminar Vortex Str… 2
4 "Spatio-Temporal Backpropagation for Training High-performance Spiking …1
5 "Well quasi-orders and the functional interpretation The purpose of thi… 1
6 "Concentration of Multilinear Functions of the Ising Model with Applica… 2
```

VALIDATION OF OUR CLASSIFICATION WITH LLAMA3

- We can use the same functions from the `tidymodels` package to get our performance scores
- Llama3 actually does amazingly well on this task without any prior training!

```r
# Create predict table
predict_llama <- test_data |>
  bind_cols(results_llama |> select(predicted = label)) |>
  mutate(truth = factor(label),
         predicted = factor(case_when(predicted == 1 ~ "computer science",
                                      predicted == 2 ~ "physics",
                                      predicted == 3 ~ "statistics")))


# Define performance scores
class_metrics <- metric_set(accuracy, precision, recall, f_meas)

# Check performance
predict_llama |>
  class_metrics(truth = truth, estimate = predicted)
```

```
# A tibble: 4 × 3
  .metric   .estimator .estimate
  <chr>     <chr>          <dbl>
1 accuracy  multiclass     0.89
2 precision macro          0.798
3 recall    macro          0.809
4 f_meas    macro          0.802
```

TESTING WITH OTHER MODELS

Our neural network from last week:

```
# Create recipe
library(tidymodels)

load("results/m_ann.Rdata")

# Create predict table
predict_nn <- predict(m_ann, test_data) |>
   bind_cols(test_data) |>
   mutate(truth = factor(label),
          predicted = .pred_class)
```

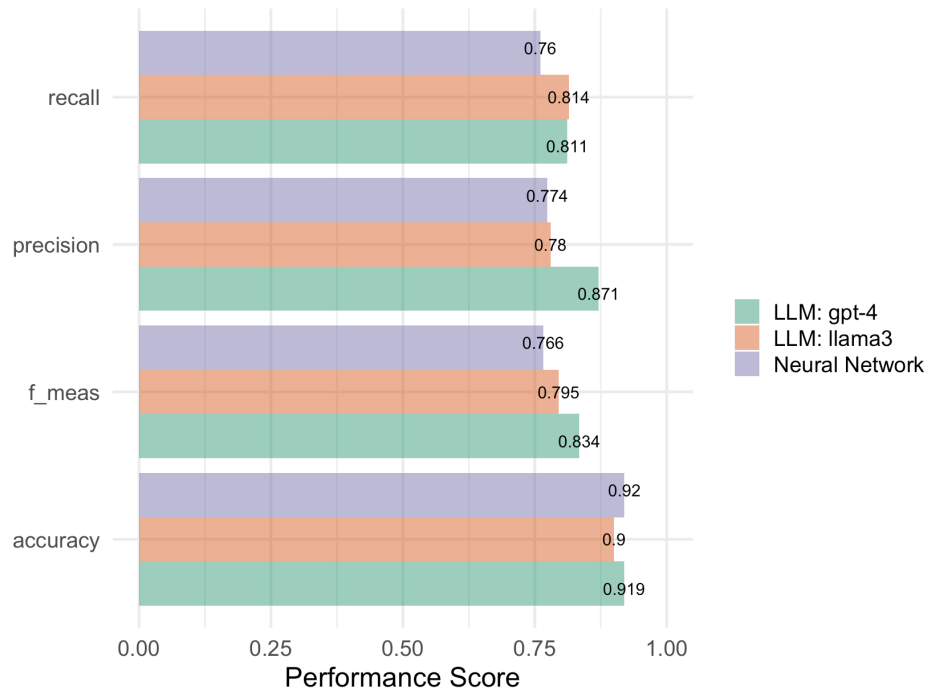
The same classification with GPT-4:

```
# Create a function that to map across the prompts
classify_sequential_gpt4 <- function(texts,message){
    raw_code <- message |>
        chat(openai(.model = "gpt-4",
                    .temperature = .0)) |>
        get_reply()
    tibble(text = texts, label = raw_code)
}

# Run commands
results_gpt <- tibble(texts = test_data$text,
                      message = classification_task) |>
  pmap_dfr(classify_sequential_gpt4, .progress = T)

# Create predict table
predict_gpt <- test_data |>
  bind_cols(results_gpt |>
            select(predicted = label)) |>
  mutate(truth = factor(label),
         predicted = parse_number(predicted),
         predicted = factor(case_when(predicted == 1 ~ "c
                                      predicted== 2 ~ "phy
                                      predicted == 3 ~ "s
```

COMPARISON

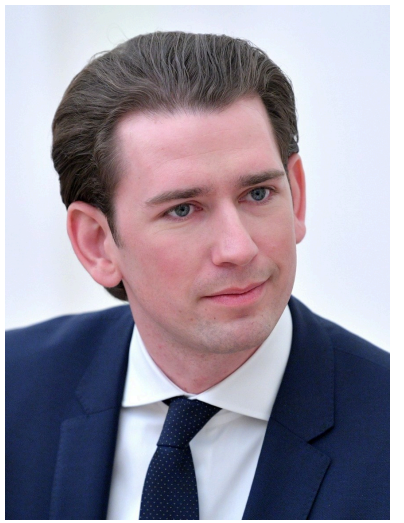
As we can see, both LLMs perform almost as good as out neural network, GPT-4 does even better, despite not having been trained on any of the data!

```
bind_rows(
   predict_llama |>
   class_metrics(truth = truth,
                 estimate = predicted) |>
   mutate(model = "LLM: llama3"),
   predict_nn |>
   class_metrics(truth = truth,
                 estimate = predicted) |>
   mutate(model = "Neural Network"),
   predict_gpt |>
   class_metrics(truth = truth,
                 estimate = predicted) |>
   mutate(model = "LLM: gpt-4")
) |>
  ggplot(aes(x = .metric, y = .estimate,
             fill = model)) +
  geom_col(position = position_dodge(), alpha = .5) +
  geom_text(aes(label = round(.estimate, 3)),
            position = position_dodge(width = 1)) +
  ylim(0, 1) +
  coord_flip() +
  scale_fill_brewer(palette = "Dark2") +
  theme_minimal(base_size = 18) +
  labs(y = "Performance Score", x = "", fill = "")
```



EXAMPLES IN THE LITERATURE

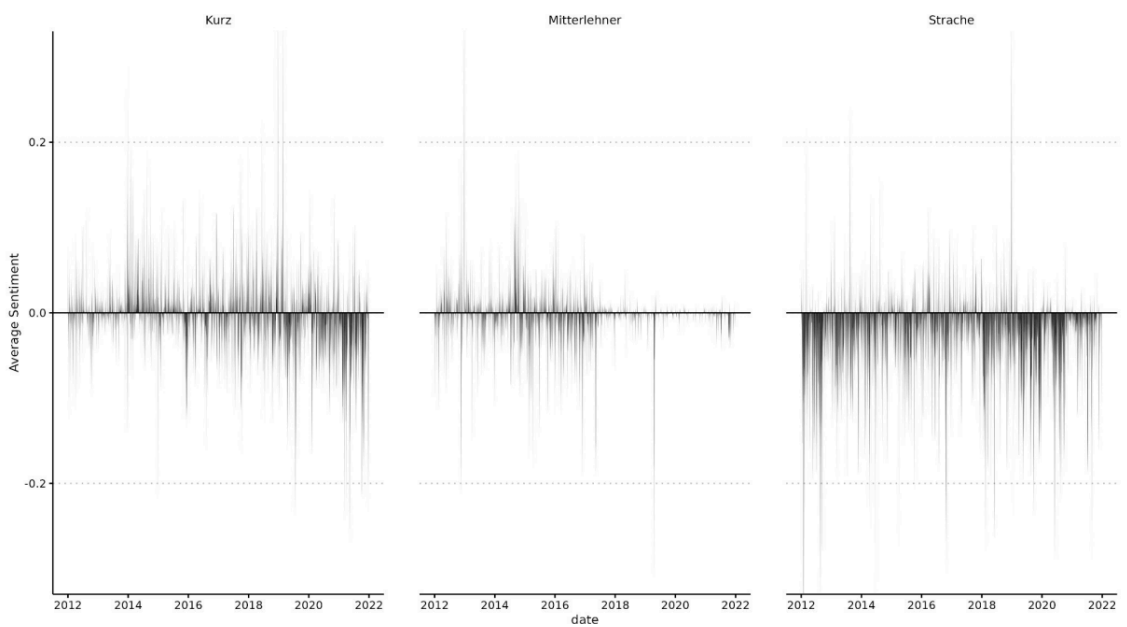
- Baluff et al. (2023) investigated a recent case of media capture, a mutually corrupting relationship between political actors and media organizations.
- This case involves former Austrian chancellor who allegedly colluded with a tabloid newspaper to receive better news coverage in exchange for increased ad placements by government institutions.
- They implemented automated content analysis (using BERT) of political news articles from six prominent Austrian news outlets spanning 2012 to 2021 (n = 188,203) and adopted a difference-in-differences approach to scrutinize political actors' visibility and favorability in news coverage for patterns indicative of the alleged serious breach of professional political and journalistic norms.



METHODS

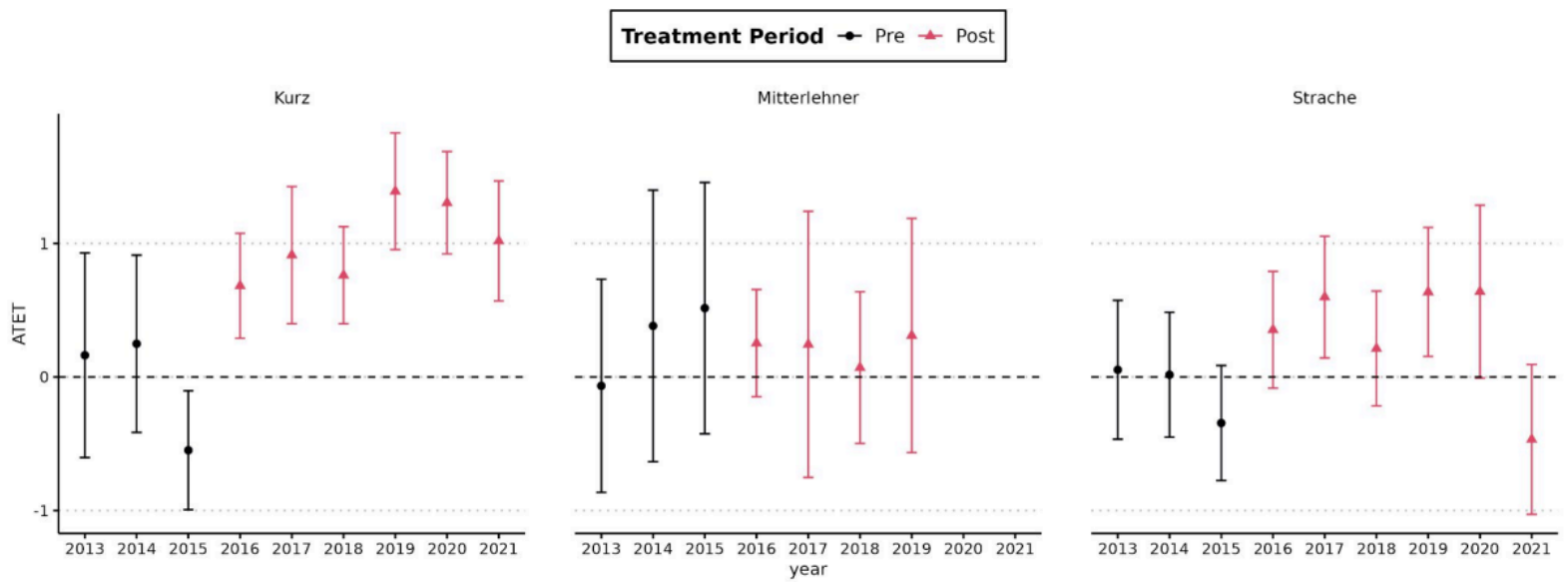
- Used a German-language GottBERT model (Scheible et al., 2020) that they further fine-tuned for the task using publicly available data from the AUTNES Manual Content Analysis of the Media Coverage 2017 and 2019 (Galyga et al., 2022; Litvyak et al., 2022c)
- Comparatively difficult task, but were able to reach a satisfactory F1-Score of 0.77 (precision = 0.77, recall = 0.77).

Figure 2. Results of sentiment analysis per politician. Aggregation of sentiment across all outlets on a daily level.



FINDINGS

- The findings indicate a substantial increase in the news coverage of the former Austrian chancellor within the news outlet that is alleged to have received bribes.
- In contrast, several other political actors did not experience similar shifts in visibility nor are similar patterns identified in other media outlets.



Notes: The plot shows the point estimates alongside their confidence intervals. The estimations for each year show the effect in comparison to the reference year. In the pre-treatment period, the point estimate refers to the previous year. In the post-treatment period, the point estimates show the effect compared to 2015. An effect of approximately 1 corresponds to an increase of approximately 100% in coverage.

Summary and conclusion

A LOOK BACK AT THE CHRONOLOGY OF NLP

1763

BAYES' THEOREM

- Reverend Thomas Bayes's work is published two years after his death
- It presents major underpinnings of Bayes theorem that is still used in Machine Learning (and Bayesian Statistics) today

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Naïve Bayes

1842

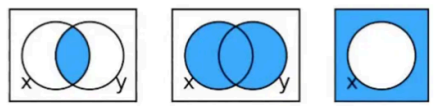
FIRST ALGORITHM?

- Ada Lovelace develops the first algorithm "Note G"
- She is widely seen as the first programmer
- She suggested that a machine could take advantage of computation in order to deal with the realms of music, graphics, and language

1854

BOOLEAN LOGIC

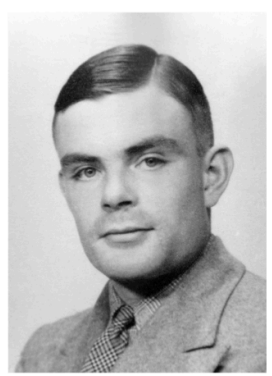
- George Boole invents the Boolean Algebra
- It has been fundamental in the development of digital electronics, and is provided for in all modern programming languages



1934

MACHINES THAT LEARN?

- Alan Turing proposes a machine that can learn
- The Universal Turing Machine is an abstract computing machine consisting of a limitless memory and a scanner that moves back and forth through the memory, symbol by symbol, reading what it finds and writing further symbols



A LOOK BACK AT THE CHRONOLOGY OF NLP

1949

WEAVER'S MEMORANDUM

- most influential publication in the early days of machine translation
- formulated goals and methods before most people had any idea of what computers might be capable of
- direct stimulus for the beginnings of research into natural language processing

1957

GRAMMAR THEORIES

- early work in MT was very simple: dictionary-lookup of appropriate words for translation
- Noam Chomsky developed a rule based-system of syntactic structures
- brought insight into how mainstream linguistics could help machine translation

Dictionary Approaches

1960s

ELIZA, ALPAC REPORT & AI WINTER

- The most famous prototype is perhaps ELIZA, designed to replicate the conversation between a psychologist and a patient
- The report by the Automatic Language Processing Advisory Committee (ALPAC), a committee established in 1964 by the United States, led to significant cuts in funding

1970s-80s

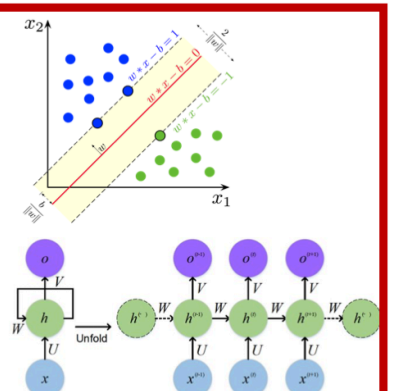
CONCEPTUAL ONTOLOGIES

- The 1970s brought new ideas into NLP
- Development of conceptual ontologies, i.e. sets of concepts and categories that shows their properties and the relations between them
- structured real-world information into computer-understandable data

1990

STATISTICAL MODELS

- statistical models came as a revolution in NLP (SVM, regression-based approaches)
- replacing most natural language processing systems based on complex sets of hand-written rules
- Recurrent neural networks and long short-term memory networks



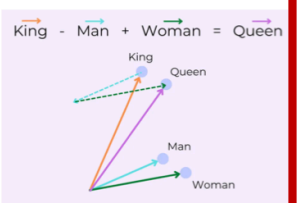
Classic Machine Learning

A LOOK BACK AT THE CHRONOLOGY OF NLP

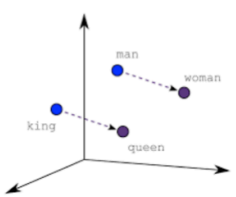
2000s

LANGUAGE MODELING, WORD EMBEDDINGS

- neural networks are used for learning the distribution of language words and predicting the next words in a text given the previous ones
- Start of representing words with dense vectors of numbers
- 2012: First unsupervised neural network by Google



2013



WORD2VEC

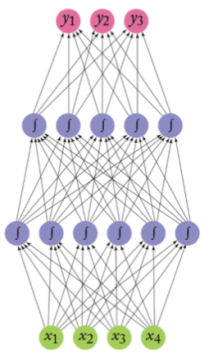
- first algorithm capable of learning word embedding efficiently
- possible to do mathematical operations with them that represent semantic operations
- using pre-trained embeddings as features for ML models improves performance in downstream tasks

Word-Embeddings

2014-2016

LSTM RNNs, CNNs, ENCODER-DECODER

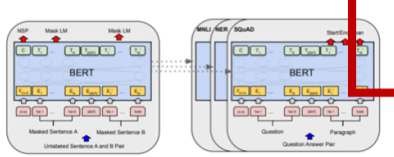
- Most use models are long short-term memory recurrent neural networks, and convolutional neural networks
- sequence-to-sequence learning, a general end-to-end approach for mapping one sequence to another using a neural network
- the first encodes the entire input into a vector, the second decodes the output from the vector



2017

ATTENTION AND TRANSFORMERS

- Introduction of the principle of attention
- Rise of Transformers, which are still used heavily today
- First generative pre-trained transformer by OpenAI in 2018!



Transformer

2020s

Large Language Models

PRE-TRAINED MODELS AND LARGE LANGUAGE MODELS

- Transformers were trained in a self-supervised way on large data sets to become "language models"
- These models can then be reused for different task after only small adaption traning (fine-tuning)

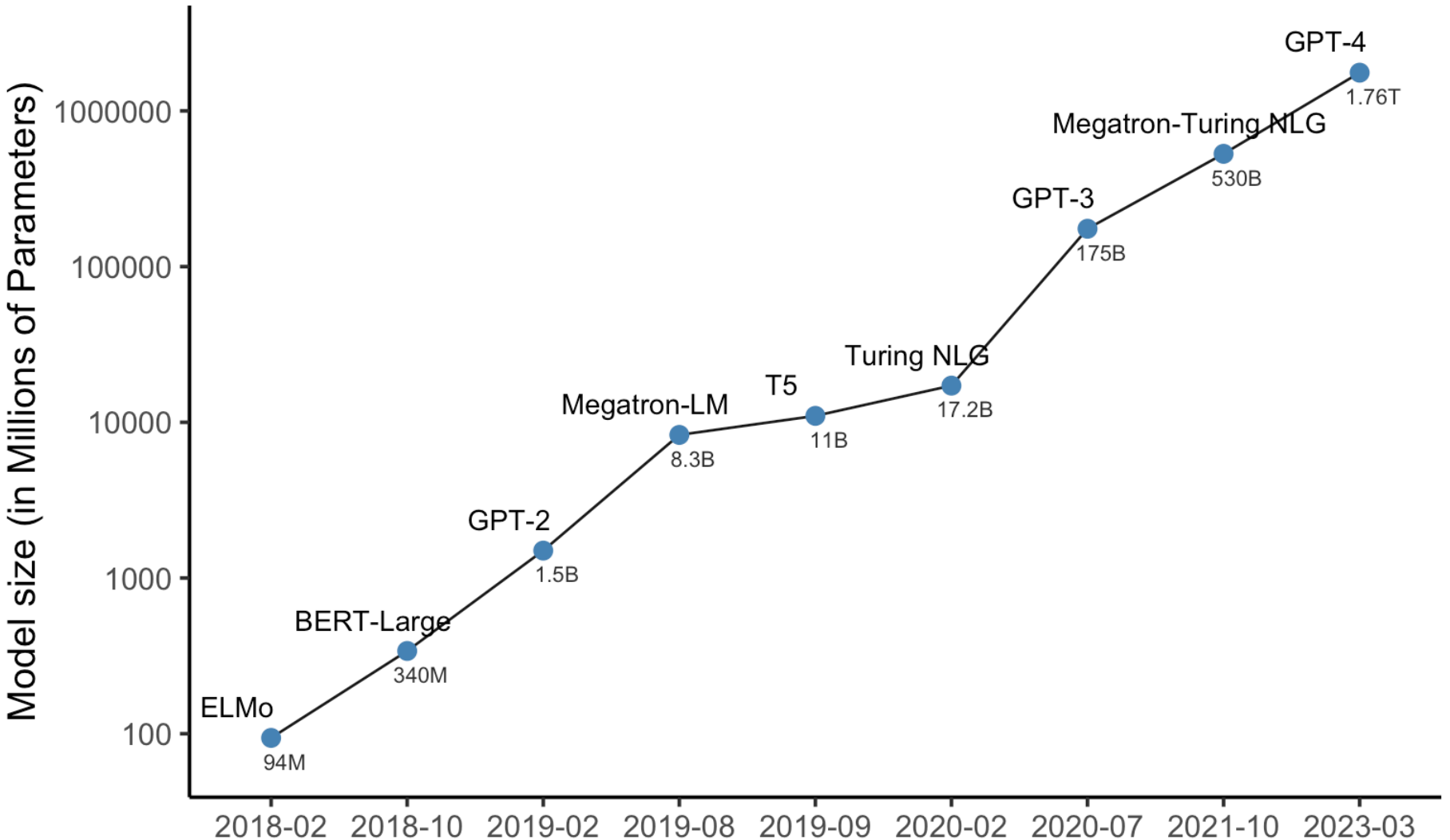
2022

GPT, BART and Co

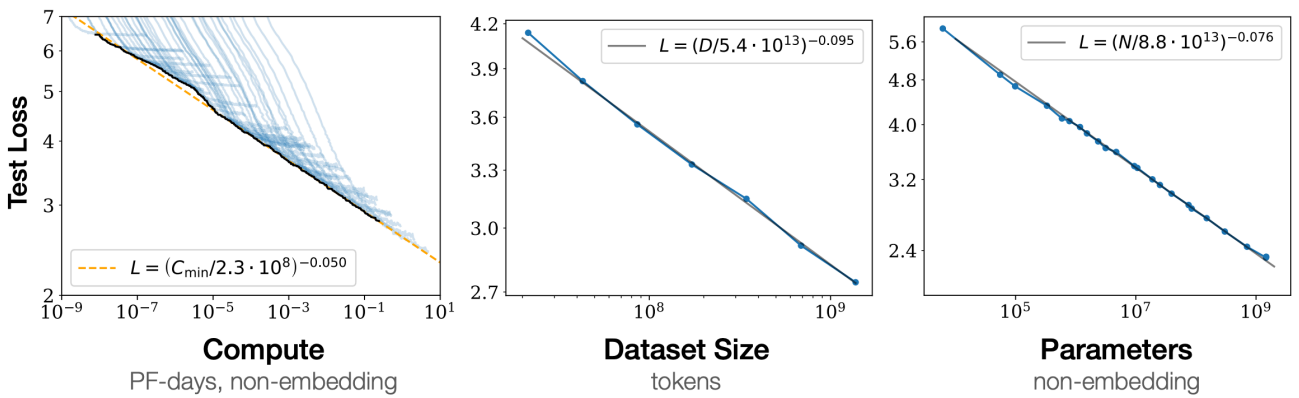
- Increasingly larger language models that are trained on the "Internet"
- November 2022: Launch of chatGPT and wide-spread use of LLMs
- AI revolution?

ChatGPT

EXPLOSION IN MODEL SIZE?



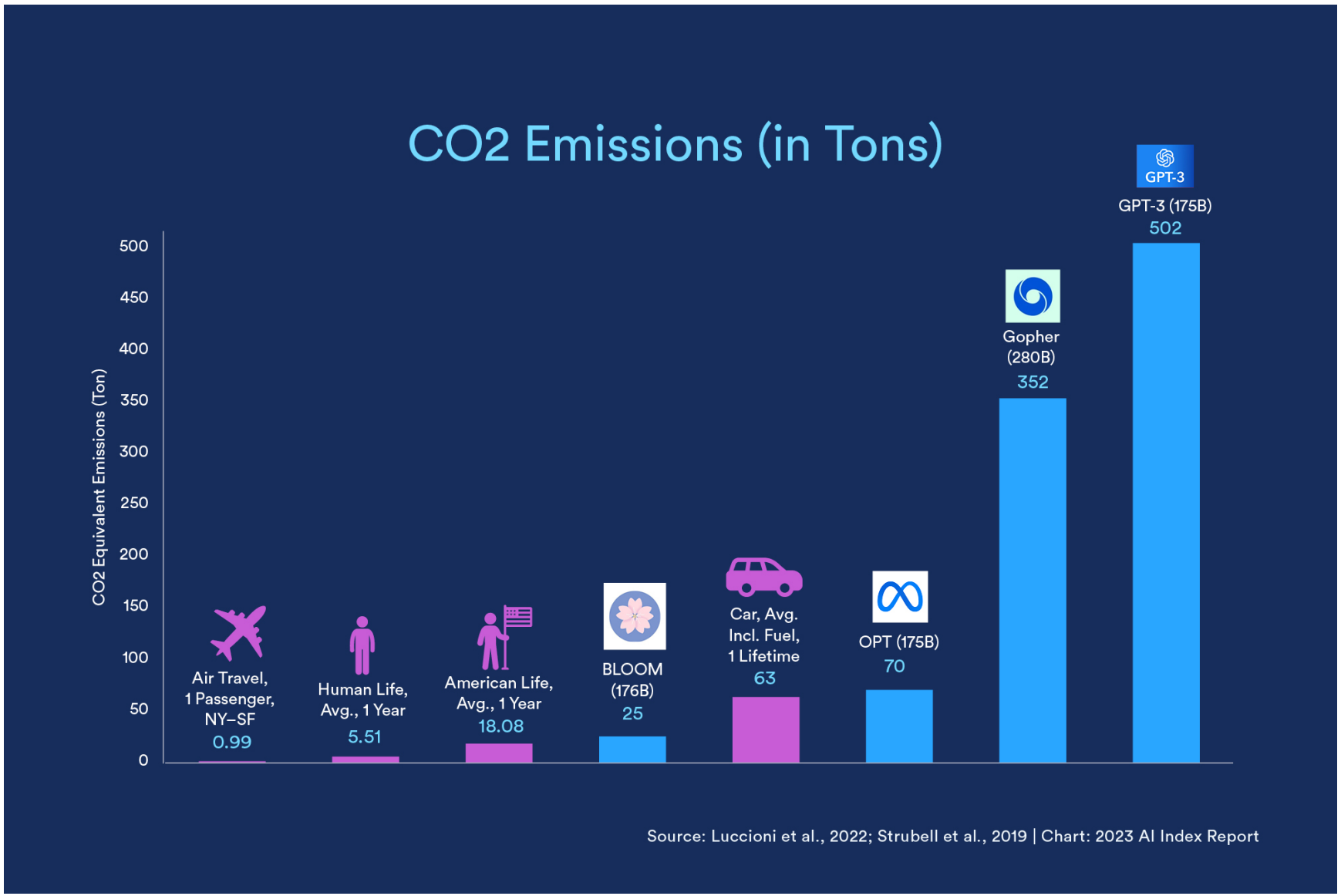
PERFORMANCE DEPENDS STRONGLY ON SCALE



Kaplan et al, 2020

- Performance depends strongly on scale, weakly on model shape
- Model performance is mostly related toscale, which consists of three factors:
  - the number of model parameters (excluding embeddings)
  - the size of the dataset
  - the amount of compute used for training
- Within reasonable limits, performance depends very weakly on other architectural hyperparameters such as depth vs. width

ENVIRONMENTAL IMPACT



ETHICAL CONSIDERATIONS

- Training large language models requires significant computational resources, contributing to a substantial carbon footprint.
- LLMs can inherit and perpetuate biases present in their training data, which can result in the generation of biased or unfair content, reflecting and potentially amplifying societal biases and stereotypes.
- Developers and users must be aware of the potential for bias and take steps to mitigate it during model training and deployment.
- The fact that some LLMs are developed, trained, and employed behind closed doors causes yet another ethical dilemma in using them!

REMINDER: GUIDELINES

#	Principle	Important aspects
1.	**Ensure privacy and data security**	• Prioritize informed consent and data privacy, if possible obtain permission for data use. • Most importantly, ensure the protection of individuals' sensitive information via strict data management and protection rules • When using proprietary software and algorithms, consider risks of sharing subject data with the companies operating them (e.g., OpenAI, Google, etc.)
2.	**Prevent bias and ensure fairness**	• Ensure that text classification models are designed to be fair, with strategies in place to identify and mitigate biases in the data and algorithms. • Regularly assess the performance of text classification models and be committed to refining them to enhance accuracy and fairness. Overall, put a strong emphasis on validation!
3.	**Strive for transparency**	• Use transparent and interpretable text classification models if possible, allowing users to understand how decisions are made and to address concerns regarding model opacity. • Share more complex models for re-use and evaluation.
4.	**Be mindful about conducting social experiments**	• If possible, obtain informed consent and ensure thorough debriefing • If that is not possible or undesired, thoroughly assess potential negative consequences of the study. Only if they are acceptable or not different from everyday use of technology, the study may be ethical to conduct (think about psychological consequences, discrimination, political implications…)
5.	**Reduce environmental impact**	• A lot of task do NOT require the use of computationally extensive approaches (e.g., the fine-tuning of large language models); test performance on smaller subsets before wasting energy on using an overly complex approach for a simple task • Assess the impact of your research: Is it worth using this much energy in light of expected impact?
6.	**Ensure cross-cultural sensitivity**	• Acknowledge and respect cultural differences in language and context, adapting text classification models to be sensitive to various cultural norms and nuances.
7.	**Engage with peers, subjects, community and stakeholders**	• Engage with relevant communities and stakeholders to gather feedback, understand concerns, and involve them in shaping the development and application of text classification methods.

CONCLUSION

- Advancement in NLP and AI are fast-paced; difficult to keep up
- LLMs promise immense potential for communication research
- Yet, large language models can contain biases or even hallucinate!
  - Validation, validation, validation!
- Also: We see already that more and more content online is AI-based. What does it mean if in the future, LLMs are trained on their own content?

AND IT DOESN'T STOP HERE…

- Large language models like "llava" can also identify and describe images…


```
llm_message("Describe this picture? Can you guess where it was made?",
                                  .imagefile = "img/england.jpeg") |>
  chat(ollama(.model = "llava", .temperature = 0))
```

```
Message History:
system:
You are a helpful assistant
--------------------------------------------------------------
user:
Describe this picture? Can you guess where it was made?
 -> Attached Media Files:  england.jpeg
--------------------------------------------------------------
assistant:
The image shows a scene from London, England. There is a
blue taxi cab driving on the left side of the road, which
is typical for the UK. In the background, there's a famous
landmark known as Big Ben, which is part of the Palace of
Westminster and the Elizabeth Tower, located in the heart
of London. The sign "LOOK RIGHT" indicates that drivers
should look to their right before proceeding, which is
consistent with driving on the left side of the road in the
UK. The presence of a crowd of people suggests this might
be a popular tourist area or a busy time of day. The overall
scene is iconic and instantly recognizable as being from
London.
--------------------------------------------------------------
```

Thank you for your attention!

REQUIRED READING

Kroon, A., Welbers, K., Trilling, D., & van Atteveldt, W. (2023). Advancing Automated Content Analysis for a New Era of Media Effects Research: The Key Role of Transfer Learning. Communication Methods and Measures, 1-21

(available on Canvas)

REFERENCE

- Alammar, J. (2018). The illustrated Transformer. Retrieved from: https://jalammar.github.io/illustrated-transformer/
- Andrich, A., Bachl, M., & Domahidi, E. (2023). Goodbye, Gender Stereotypes? Trait Attributions to Politicians in 11 Years of News Coverage. Journalism & Mass Communication Quarterly, 100(3), 473-497. https://doi-org.vu-nl.idm.oclc.org/10.1177/10776990221142248
- Balluff, P., Eberl, J., Oberhänsli, S. J., Bernhard, J., Boomgaarden, H. G., Fahr, A., & Huber, M. (2023, September 15). The Austrian Political Advertisement Scandal: Searching for Patterns of “Journalism for Sale”. https://doi.org/10.31235/osf.io/m5qx4
- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
- Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., … & Amodei, D. (2020). Scaling laws for neural language models. arXiv preprint arXiv:2001.08361.
- Kroon, A., Welbers, K., Trilling, D., & van Atteveldt, W. (2023). Advancing Automated Content Analysis for a New Era of Media Effects Research: The Key Role of Transfer Learning. Communication Methods and Measures, 1-21
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M. A., Lacroix, T., … & Lample, G. (2023). Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

EXAMPLE EXAM QUESTION (MULTIPLE CHOICE)

How are word embeddings learned?

A. By assigning random numerical values to each word

B. By analyzing the pronunciation of words

C. By scanning the context of each word in a large corpus of documents

D. By counting the frequency of words in a given text

EXAMPLE EXAM QUESTION (MULTIPLE CHOICE)

How are word embeddings learned?

A. By assigning random numerical values to each word

B. By analyzing the pronunciation of words

C. By scanning the context of each word in a large corpus of documents

D. By counting the frequency of words in a given text

EXAMPLE EXAM QUESTION (OPEN FORMAT)

What does zero-shot learning refer to in the context of large language models?

In the context of large language models, zero-shot learning refers to the ability of a model to perform a task or make predictions on a set of classes or concepts that it has never seen or been explicitly trained on. Essentially, the model can generalize its knowledge to new, unseen tasks without specific examples or training data for those tasks.

In traditional machine learning, models are typically trained on a specific set of classes, and their performance is evaluated on the same set of classes during testing. Zero-shot learning extends this capability by allowing the model to handle tasks or categories that were not part of its training set.

In the case of large language models like GPT-3, which is trained on a diverse range of internet text, zero-shot learning means the model can understand and generate relevant responses for queries or prompts related to concepts it hasn't been explicitly trained on. This is achieved through the model's ability to capture and generalize information from the vast and varied data it has been exposed to during training.